Real-time Deep Neural Networks for internet-enabled arc-fault detection

Citation: Siegel, Joshua E. et al. "Real-time Deep Neural Networks for internet-enabled arc-fault detection." Engineering Applications of Artificial Intelligence 74 (September 2018): 35-42 © 2018 Elsevier

As Published: https://doi.org/10.1016/j.engappai.2018.05.009

Publisher: Elsevier

Persistent URL: https://hdl.handle.net/1721.1/121372

Version: Original manuscript: author's manuscript prior to formal peer review

Real-Time Deep Neural Networks For Internet-Enabled Arc-Fault Detection

Joshua E. Siegel[a,*], Shane Pratt, Yongbin Sun, Sanjay E. Sarma

[a] *Field Intelligence Laboratory, Massachusetts Institute of Technology*
77 Massachusetts Avenue, Room 35-205, Cambridge MA USA 02139

Abstract

We examine methods for detecting and disrupting electronic arc faults, proposing an approach leveraging Internet of Things connectivity, artificial intelligence, and adaptive learning. We develop Deep Neural Networks (DNNs) taking Fourier coefficients, Mel-Frequency Cepstrum data, and Wavelet features as input for differentiating normal from malignant current measurements. We further discuss how hardware-accelerated signal capture facilitates real-time classification, enabling our classifier to reach 99.95% accuracy for binary classification and 95.61% for multi-device classification, with trigger-to-trip latency under 220ms. Finally, we discuss how IoT supports aggregate and user-specific risk models and suggest how future versions of this system might effectively supervise multiple circuits.

Keywords: Emerging applications and technology, intelligent infrastructure, ambient intelligence, embedded intelligence, distributed sensing, arc fault detection, real-time

1. Arc Detection Matters

Electrical circuits harbor silent and serious risks. Conductors flex, break, and oxidize; insulation abrades, and interconnects, switches and terminals degrade. Wires are routed in hard-to-inspect areas between walls, ignored until problems

*Corresponding author
Email address: j_siegel@mit.edu (Joshua E. Siegel)

manifest. One such problem is arcing, an unintended, luminous and sustained discharge of electricity in conductive, ionized gas between two regions of varied electrical potential.

Arcs may be series or parallel. Series arcs occur when a conductor is unintentionally broken, *e.g.* from a loose connector, a poorly-made splice, or an accidental nick or cut. Parallel arcs occur between hot and neutral or ground, or neutral and ground. Though parallel arcs burn hotter, series arcs have the potential to burn between $5,000$ and $15,000°$C[1], expelling molten liquid capable of starting fires.

Since 1998, specialized devices called Arc Fault Circuit Interrupters (AFCIs)[2] have helped mitigate fire risks. These systems interrupt faulty circuits, but err on the side of over-sensitivity, disconnecting benign devices like vacuums or computers. We propose leveraging advances in sensing, connectivity, inference and action in order to build an intelligent, cost-effective Internet-of-Things enabled arc-fault detector capable of learning new definitions, similar to a virus scanner.

To prove this concept's feasibility, we examine low-power, AC series arcs, which provide worst-case training data. AC faults are difficult to classify because the circuit's connected load limits the arc's maximum current[3, 1], reducing the signal-to-noise ratio. Series faults pose a high likelihood of confusion with benign appliances such as DC motors, and by testing with low-power circuits, ($<$ 15A @ 120VAC), our algorithms will readily extend to higher-power arcs.

This paper proposes fault detection using an adaptive deep neural network trained using real data. Such a system provides a future-proof and scalable system for arc classification, maintaining sensitivity while reducing unintended interruptions. Connectivity allows device and fault "definitions" to be aggregated at scale, while on-board computation and connectivity enables operating characteristic measurements and remote control. More than describing a smart AFCI's implementation, however, this paper highlights the opportunity latent in bringing AI and connectivity into "mundane" devices, such as those found in infrastructure.

2. Existing Arc Detectors

Contemporary arc detectors leave much to be desired from the perspective of cost, immunity to false positives (unnecessary interruption), response time, and upgradability.

AFCIs may rely upon analog circuits, application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), or optical and electromagnetic techniques to detect arcs. Device sensitivities and reaction times (25-250ms) vary[4, 5].

Low-cost analog detection is most prevalent, but it suffers from high false positive rates[3, 6]. Mechanical approaches are initially more reliable, but costly and degrade over time[6].

Algorithmic detectors face different challenges. Arcs are dynamic, and detection efficacy varies as the cathode erodes[2]. Appliances may share characteristics with arcs, including current shoulders, a change in amplitude, or an increased rate of current rise[2], leading to misclassification.

Some arc detectors utilize machine learning to improve classification accuracy. For example, researchers have applied neural networks to identify abnormal operation without a priori arc models. These approaches yield between 95% and 99% accuracy using small feature vectors for training and testing, though it is unclear how resilient these approaches are to nuisance detection[7, 8, 9].

More generally, algorithmic arc detection is a form of dynamic process transient fault detection. Roverso (2002) describes one approach to dynamic fault detection using bagged recurrent neural networks, windowed wavelet feature generation, and task (fault) decomposition[10]. However, this approach might require costly hardware to deploy in real-time.

Hidden Markov Models (HMMs) learn time-dependent spatial and temporal patterns to identify state transitions from normal to abnormal plant operation. [11] Similar HMMs identify individual appliances from combined electrical loads, but require supervised model creation and long inter-state transitions not conducive to the fast ($>$ 1Hz) realtime operation required to protect against

arc-related fires[12].

Computer-controlled AFCI's running these algorithms may rely upon features including Fourier coefficients, wavelets, and use techniques such as band-pass filtering to eliminate harmonics and baseline current from measurements[6, 13, 14]. Other approaches derive features by correlating multiple information sources, for example by relating differential current ($\frac{di}{dt}$) to absolute current ($|i|$), which improves separability of nuisance tripping from fault tripping. These approaches may detect early arcing with up to 98% accuracy[15].

Some AFCIs create additional value to drive adoption. Ming (2009) developed a system using Controller Area Network (CAN) to connect sensors a single host computer for classification[16]. Koziy (2013) proposed integrating detectors into smart meters[17].

Most AFCIs rely on predefined and immutable arc definitions, leading to nuisance interruption. Developing an AFCI with adaptive and remotely-updatable definitions would provide additional utility relative to conventional approaches. Such an approach allows for common-Cloud signature aggregation to minimize nuisance disconnects while facilitating new insights (what's plugged in?) and remote control (turning off a stove while vacationing).

3. Hypothesis

Current waveforms differ between arcing and normally-operating circuits. Unlike the current traces from a resistive circuit, arc fault waveforms typically have shoulders because the arc does not flow current until sufficient voltage across the gap returns following a zero current condition (excitation and reignition). [1, 5]. Representative normal and arcing traces from an electronic stovetop and ozone (arc) generator can be seen in Figures 1 and 2.

Listening to these signals as audio, we could differentiate between resistive and arcing signals. We therefore hypothesized that audio processing techniques may be used to classify normal and faulty circuits. Audio-based classification has been successfully applied to the development of automotive diagnostic

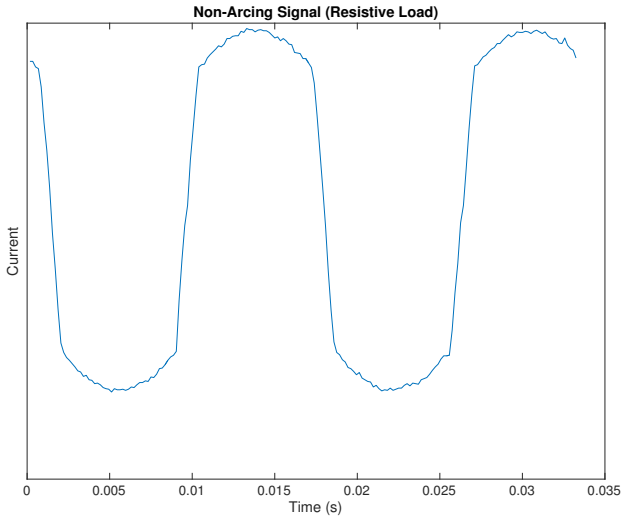

Figure 1: This figure shows a typical smooth and periodic current trace for a resistive electrical load.

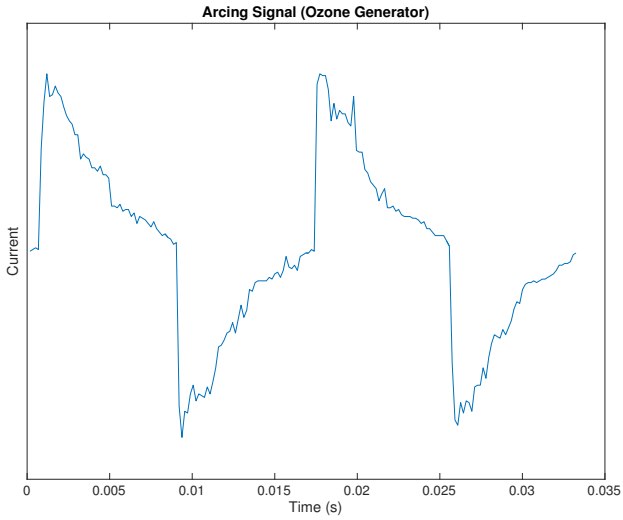

Figure 2: In this arcing current trace, note the shoulders as the electrical potential climbs before creating an arc.

systems[18, 19]. However, these approaches rely on physical or statistical models. Deep learning techniques might instead allow for normal and abnormal

classification without an a-priori hypothesis, and would scale to support the volume of data generated by connected devices.

In the following sections, we test a neural network using audio features to identify circuit operating states.

4. Experimental Setup

Typical arc fault detector training data comprises arcs, nuisance trips, and normal circuits. Arc testing approaches including guillotine, carbonized path, wet arc, and loose terminals. Nuisance trip sources, designed to test false positive rejection, include motor loads, dimmers, and computers [3]. For normal circuits, resistive elements are used.

Earlier papers' training and testing would provide a uniform baseline for evaluation but the data were unavailable. Further, these data sets neglect multi-state classification, which is a key capability of our solution.

We instead generated data from an electric stove-top burner, an iMac computer, a fan, and an ozone generator. The burner simulates an ideal resistive circuit, the iMac switching power supply introduces noise, and the fan's DC motor arcs by design. The ozone generator represents a continuous series fault, as it relies on a high-voltage discharge to cause an arc between two metallic grid plates, resulting in the continuous formation of O_3.

We validated the ozone generator's similarity to real arcs by comparing its current traces to those found in the literature. Visual inspection of the time/current trace in Figure 2 shows a strong correlation to the point-to-point AC series arcing demonstrated in Li (2003) [3].

We collected data from each device under real-world use to ensure that classification results depend not on signal amplitude or periodic features, but rather on invariant waveform "signatures." In the case of the iMac, we rendered video; in the case of the burner, we varied target temperature and applied thermal loads to the heating element.

Current data were recorded on a Raspberry Pi Model 3 microcomputer

via an MCP3008 10-bit analog to digital converter connected to a clamp-on current measurement meter, as show in Figure 3. Current can be measured non-invasively, allowing an airgap between logic electronics and measurement circuitry.

An op-amp amplifies normal and inverted input signals from the clamp transformer. The voltage proportionate to current input is reconstructed using the formula $V_{sig} = V_{ADC0} - V_{ADC1}$. We calibrated the amplifier's gain by loading the circuit to 15A and tuning a trim potentiometer to ensure that the analog input was used across its full range without saturation.

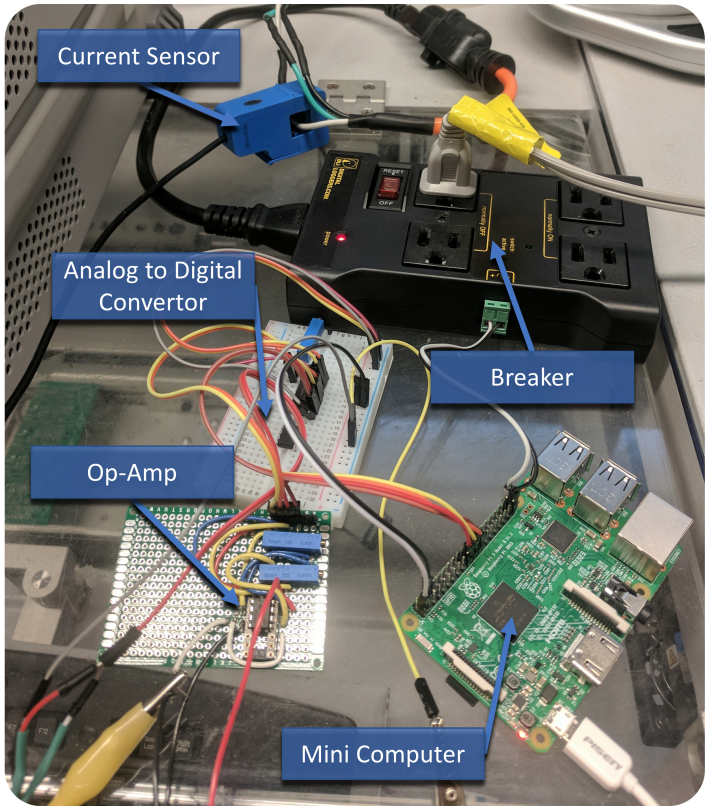

Figure 3: This image shows the experimental setup for data collection (and later, for testing real-time classification).

This setup sampled at 5.865kHz, limited by the Raspberry Pi's SPI band-

width and timing jitter. The number of wire turns on the inductive pickup and the operational amplifier's gain of 6.6x yielded a precision of 14.6mA of current draw per count. Each sample was recorded for at least 60 minutes to ensure sufficient training and testing data.

5. Proof of Concept

We treated classifier development as a supervised learning problem. This section describes feature generation and the design of a Deep Neural Network (DNN) for binary (normal versus arcing) and multi-state (device) classification.

5.1. Feature Generation

Each measurement was assigned a label based on device type, then split into chunks of 0.2 seconds to capture nearly 12 complete 60Hz AC cycles while allowing 50ms for classification and to mechanically switch a relay.

Each chunk was RMS current normalized, and low amplitude signals (noise) were discarded to avoid classifying unloaded circuits.

For each segment, we generated a feature vector in Python using SciPy, SkLearn, PyLab, PythonSpeechFeatures and PyWt. These features included the Discrete Fourier Transform (DFT) with a bin width of 5Hz and a maximum frequency of 1.5kHz to avoid aliasing. The average FT magnitude in each bin comprised one feature, with all such features concatenated into feature set F_{DFT}.

In addition to the binned FT, we used Mel Frequency Cepstral Coefficients (MFCC) to provide a spectral signature of the current. We generated 64 frames with 12 coefficients, stored in F_{MFCC}.

We also created wavelet-based features by conducting a Discrete Wavelet Transform (DWT) using the Daubechies 4 wavelet at decomposition level 5. For each decomposition level, we computed mean, kurtosis, standard deviation and skewness features, concatenating each into a wavelet feature vector, F_{DWT}.

Finally, these features were concatenated to form one vector comprising DFT, MFCC and DWT features ($F_{ALL} = F_{DFT} \| F_{MFCC} \| F_{DWT}$) for each sample current measurement, where $\|$ represents vector concatenation.

The use of FT, DWT, and MFCC was chosen for its efficacy in diagnosing automotive faults[18, 19].

5.2. Initial DNN Model

To test neural networks' viability on constrained hardware, we implemented a DNN classifier. Bringing "intelligence to the edge" allows durable goods and infrastructure devices to adapt in the face of new data, and DNNs do not require excessive engineering, even for multi-state classification which non-linear separability.

We implemented a fully-connected DNN in TensorFlow. The model takes as input the generated 1D vector F_{ALL}, and outputs a probability distribution over predefined classes. In our experiments, we found that a model comprising three hidden layers with the number of neurons being 16, 32 and 16, respectively, works well. The model architecture is visualized in Figure 4.

All the three hidden layers are Fully-Connected (FC) layers: each neuron connects 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 Rectified Linear Unit (ReLU). For the output layer, Sigmoid function is used as the activation function for the binary classification case, and the Softmax function is used for multi-state classification.

This model used the Adam optimizer with a learning rate 0.001.[20] The batch size was 64 signal segments. We calculated the cross entropy between ground truth labels and predicted logits as the training loss and used L1 regularization to further reduce model overfitting. We randomly split data chunks into two sets to avoid time dependence, keeping 80% for training and 20% for testing. We had a large sample size, so did not perform multi-fold cross-validation.

For testing, we deployed the testing script with pre-trained model parameters

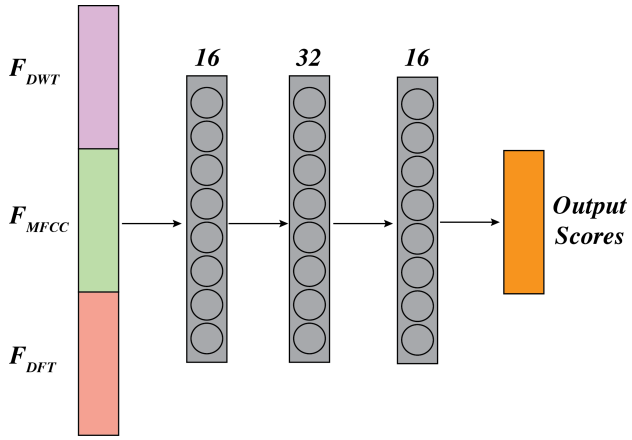

Figure 4: The input to the DNN is a 1D vector comprising $F_{DFT} \| F_{MFCC} \| F_{DWT}$. The model has three hidden layers, with 16, 32 and 16 neurons. The model generates a probability distribution over predefined classes to determine normal versus abnormal circuit operation, as well as identifying specific outlet loads.

on an iMac. We further designed a multithread scheme for operation within a real-time application, where the classification task is implemented on a separate, non-blocking thread. This parallel design helped our prototype system perform with minimal data loss or delay.

6. Early Results and Refinement

This first DNN returned 99.98% classification accuracy on in-sample data using a one-versus-all approach (ozone [abnormal] versus all other input data [normal]). With a 0.2s data segment, these results represent classification on $12,357$ unique events representing a roughly 30/70%, with the confusion matrix shown in Table 1.

The false positive rate (good circuit reported as faulty) was 0.01%, almost eliminating nuisance trips. The false negative rate (bad circuit reported as good) was 0.05%, suggesting the algorithm may need tuning or biasing to minimize fire risk. The sensitivity (when an arc is present, how often does the algorithm detect it) was 99.95%, the specificity (when the circuit is safe, how often does the

Table 1: This confusion matrix shows the DNN's binary (normal/abnormal) classification performance.

	Normal	**Arcing**
Normal	8,674	1
Arcing	2	3,680
	n=	12,357

Table 2: The confusion matrix for a DNN device identification algorithm for multi-device classification.

	Burner	**iMac**	**Fan**	**Ozone**
Burner	1,284	5	11	11
iMac	9	3,553	109	12
Fan	5	135	3,539	2
Ozone	9	5	1	3,667
			n=	12,357

algorithm detect this) was 99.99%, and the precision (when a fault is predicted, how often is it actually faulty) was 99.46%, demonstrating stellar performance. This improves significantly upon the 99% state of the art AFCI performance.

Categorizing by device type, the DNN returned a four-class accuracy of 97.46%. The resulting confusion matrix is shown in Table 2.

From this matrix, one sees that that the in-sample performance for ozone (arc) detection is high, with 15 of $3,682$ samples (0.4%) misidentified, suggesting a high sensitivity relative to other classes that might be more easily improved upon.

We then verified that the model could improve over time by incorporating 15 minutes of additional training data from the same devices. The in-sample binary performance reached 100% accuracy, and multi-state reached 99.98%. We validated that the improved model was not overfit by running the classifier on 10 hours of data from an outsample non-arcing load, with no false positives.

6.1. Definition Updatability

We then tested the ability of the model to dynamically update "definitions" for unseen device configurations by testing the previously-trained model on a circuit powering the fan and ozone generator in parallel. The model failed to detect the arc, so we collected additional training data and retrained the classifiers to include parallel devices.

We retrained the model by starting with the already-learned weights and biases and introducing new data into the training set (a similar approach could be used for Cloud-connected AFCIs, which would learn new weight and bias definitions at a central location and deploy a "delta" over-the-air update).

We changed the learning parameters to minimize overfit, reducing the batch size and learning rate and increasing the maximum step limit to ensure convergence with the smaller batches. In a future implementation, we will consider LeCun et al.'s work on "Efficient BackProp" to start with small mini-batches and increase size as training progresses to reduce model noise and to allow computation on more constrained hardware.[21]

The lower learning rate reduces sensitivity to deep, narrow feature troughs and improves generalizability to outsample data, but requires running longer relative to a higher learning rate to avoid converging to a local minimum.

Despite these changes, the model failed to differentiate particular segments of the burner load from the arc load. This is because arcs tend to be low current, and the burner uses a bang-bang controller to modulate heat. When the load resistor is disengaged to allow the heating element to cool down, the burner becomes very low current, driving only an unloaded power supply and an indicator light. In these cases, the burner's power supply demonstrates low-amplitude noise closely representing arcing.

To address this issue, we reduced the training and realtime classification cutoff threshold from an RMS current value of 146mA to a value of 43.6mA, allowing classification of lower-amplitude signals. We then retrained the model again.

With these new models, the classifier performed reliably using prerecorded, outsample data. We were able to obtain 100% classification accuracy on 15 minutes of samples from each represented state, with no false positives or negatives.

This implement-then-update use case proves the value in having adaptively-learned models for fault classification: new scenarios can be added easily, and classifier performance improves as additional labeled samples are introduced into the training set. The network inherently picks the best-differentiating features, so a network of AFCIs at scale will quickly lead to robust, generalizable load classifiers.

7. Building Towards a Connected AFCI

With batch processing proven, we sought to develop a real-time, IoT-enabled AFCI.

The Internet of Things is key to learning and deploying new classification models, ensuring that real-world edge cases make it into training data. Aggregated fingerprints and centralized model training allows scalable Cloud resources to be leveraged by infrastructure with growing "embedded intelligence." Connectivity and distributed computation allow constrained microcomputers to participate in on-line and adaptive learning, helping crowdsource classifier development and enabling remote control that incentivizes the installation of costly "smart" switches.

The first step towards real-time arc detection was to implement the DNN on a low-cost, power-efficient microcomputer. We began by testing the classifier on the same Raspberry Pi 3 from the experimental setup to capture live data, with the addition of a relay-controlled outlet to interrupt abnormal circuits. This relay had a nominal switching time of 5ms, but could be replaced by a faster (and costlier) solid state relay. The full system block diagram appears in Figure 5, and the schematic appears in Figure 6.

The Raspberry Pi met our objective of being low-cost and power-efficient. At $35 for the microcomputer and slightly more for supporting hardware, a

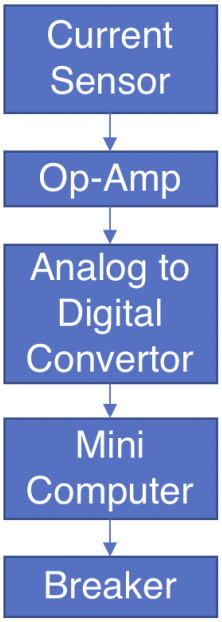

Figure 5: This block diagram shows the elements of the connected AFCI's data collection and outlet control system.

single Raspberry Pi with a four-channel MCP3008 could monitor four outlets if installed near a distribution box. The ongoing operating cost is also low – with a measured 2W draw, at 0.12c/kWh this would cost just $2.10 annually.

The software required more adaptation for real-time use. While feature generation took 15ms, the classification initially took 7.5s for a 0.2s data sample. We worked with TensorFlow's lower-level functions to load the graph to memory once in a persistent session, which sped classification up to 3ms. With a three-thread software implementation (a caller thread, a data collection thread, and a prediction thread), we were able to capture and classify data with zero signal loss. This architecture is shown in Figure 7.

The resulting system took approximately 3ms for classification, 15ms for feature generation, and 10ms to trigger the solid-state power interruption relay. From a computation perspective, with a 200ms data sampling window, the total time from arcing to disconnection was bounded to 240ms or fewer, which

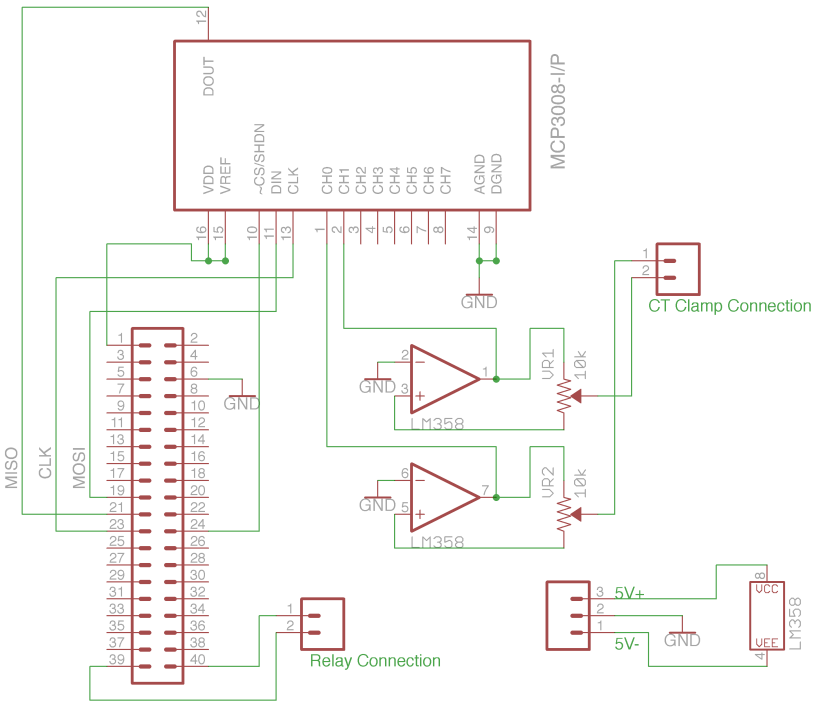

Figure 6: This schematic shows the MCP3008 ADC, a dual LM358 amplification circuit, the CT sensor input, and the Raspberry Pi SPI interface connections.

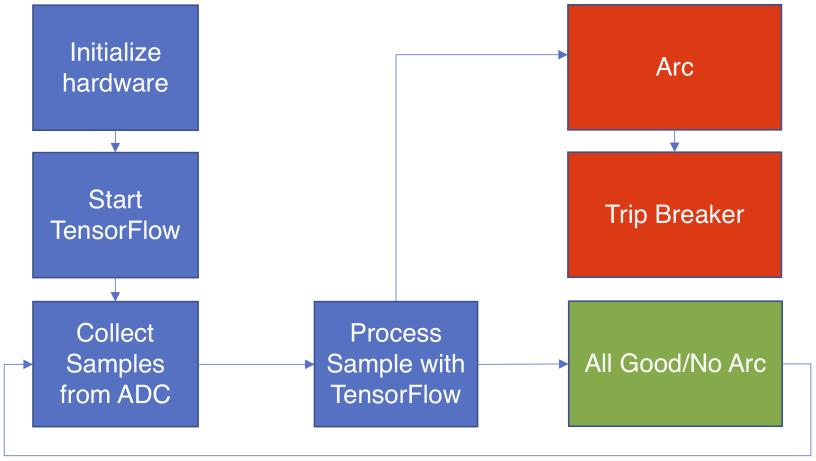

Figure 7: This diagram shows a simplified process flow for arc fault classification.

is competitive today's AFCI's.

However, as we optimized the code we encountered new and difficult-to-address challenges. While the batch-processed model demonstrated 100% reliability on outsample data, realtime classification failed to trigger in a timely manner or at all. In the next section, we consider the differences between classifying real-time and post-processed data to identify the root cause.

7.1. Software Challenges

The Raspberry Pi does not run a realtime operating system, making deterministic timing impossible. We analyzed our "real-time" data and found that for a target frequency of 5865Hz, the system clock stability had a standard deviation of 3%, causing substantial frequency variation (176Hz). As the processor cores loaded with data acquisition, OS functions, and classification, the generated FFT features suffered. This non-uniformity dramatically increased false positives and negatives.

We attempted to address these issues by running a non-preemption OS and bit-banging SPI to reduce overhead. Despite these changes, the system overhead remained at odds with deterministic sampling and processing. GPU acceleration was also not an option, as TensorFlow does not support OpenCL. As a result, in the current implementation, the Raspberry Pi 3 is capable of either collecting data reliably or classifying accurately, but not both.

8. Hardware-Accelerated Classification

Unsatisfied with accurate classification only in post-processing, we developed optimized hardware in an effort to stabilize the data capture clock and to free up CPU cycles for classification. We therefore implemented hardware acceleration, using a USB sound card to capture the input signal from the current sensor. Such devices are produced in volume and low-cost, and ideally suited to the type of data capture needed to classify electronic circuits.

USB sound cards feature integrated clock timing controllers, high-precision analog-to-digital converters, buffered storage, and automated gain compensation to simplify both the hardware and software necessary for data collection. The USB soundcard we used allowed sampling up to 48kHz at 16-bits of resolution, and enabled us to eliminate the ADC hardware, op-amp and dual power supply from the AFCI's bill of materials, reducing cost and simplifying manufacturability.

With the hardware accelerator implemented, we tested classification on our existing models. We collected data at 48kHz and saved every 8th sample to approximate our initial target sampling rate of 6kHz. The use of a USB soundcard significantly reduced CPU load during capture from 60% to 0.2%. These freed cycles made it more likely that the CPU could perform realtime classification. Further, the substantial reduction in CPU load and amount of remaining USB bandwidth mean that a single Raspberry Pi might be able to use several USB soundcards to simultaneously monitor multiple circuits (or, could use the left and right channels from a stereo soundcard to monitor two outlets from a single capture card). The new hardware setup is shown in Figure 8, with a block diagram shown in Figure 9.

Using a dedicated sound card eliminates the need for threading, as the input signal is buffered in hardware. The sound card stores a sliding window buffer of 0.2s worth of data (a single frame) and transmits it to the Raspberry Pi in a batch operation. Because the data transfer, feature generation and classification take under 0.2s, no data are lost and classification can take place in realtime without the need for multithreading.

The data from the sound card had stable timing, so we took the opportunity to update the model parameters by collecting new data at 48kHz to allow for the possibility of richer feature generation,[1] and downsampling the data to 6kHz based on our FFT features' estimated maximum frequency.

[1]The fully-sampled data are available to the community at `https://doi.org/10.7910/DVN/IFDIZ1`.

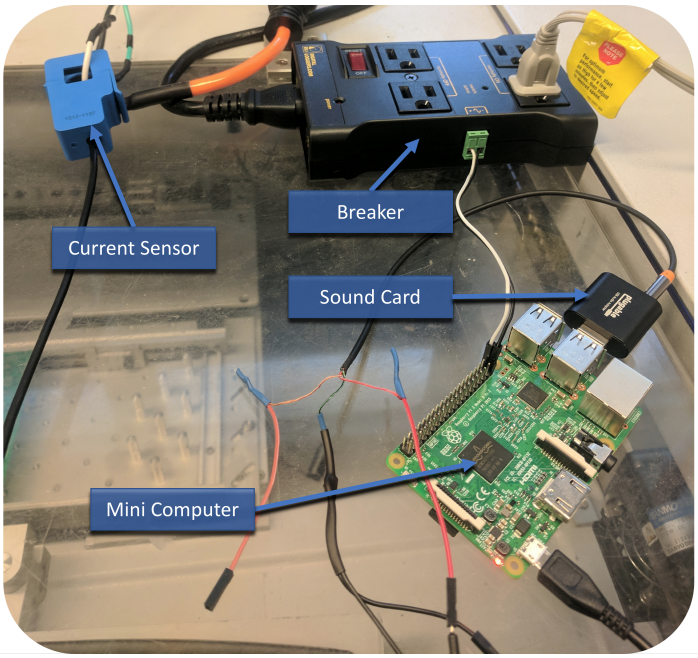

Figure 8: This image shows the experimental setup for data collection using a USB soundcard for hardware acceleration.

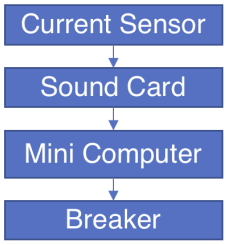

Figure 9: This block diagram shows the elements of the hardware-accelerated connected AFCI's data collection and outlet control system.

We repeated the data collection process described in Section 4 and captured at least an hour and 15 minutes of data for each input type, and captured more data from the ozone generator to keep classes relatively balanced. We also took this opportunity to make changes to the model based on our observations about the impact each hyperparameter had on model performance.

Table 3: This confusion matrix shows the hardware-accelerated classification performance on binary (normal/abnormal) circuit characterization.

	Normal	**Arcing**
Normal	21,599	1
Arcing	15	8,985
	n=	30,600

We kept layer sizes of 16, 32 and 16 fully connected units with a learning rate of 0.001 and added a decay step of $1,000$ with a decay rate of 0.9, significantly expediting the model learning process. Based on the challenges we faced earlier with tuning the model, we also increased the batch size to $2,048$ to allow for improved model generalization.

Using a grid search for optimization, we increased the DFT's upper frequency limit to $2,500$Hz to create additional differentiating features. We did not increase the feature limit to 24kHz to ensure quick computation and avoid sampling near the capture hardware's upper limit, where jitter would have more impact. We used a floor of 40Hz to eliminate low-frequency components. The final feature vector length was $n_{F_{ALL}} = 607$, comprising $n_{F_{DFT}} = 491$, $n_{F_{MFCC}} = 96$, and $n_{F_{DWT}} = 20$.

Finally, we changed early-stopping to look for three consecutive increases in validation loss at points 100 steps apart, with a floor of 10 epochs. This minimized overfit.

From the new model, we obtained the out-sample data confusion matrices for normal-versus-abnormal and multi-state classification appearing in Tables 3 and 4:

This model's binary outsample accuracy is 99.95% with a false positive rate of 0.004% (once per 83 minutes). Device identification accuracy is 95.61% with a false positive rate of 1.8%. The model's primary misclassification mode is between the fan and the burner, a non-malignant mistake. With continued supervision, these models could improve on-line.

Table 4: The confusion matrix for hardware-accelerated device identification.

	Burner	**iMac**	**Fan**	**Ozone**
Burner	3,612	0	867	21
iMac	0	5,400	0	0
Fan	371	0	11,302	27
Ozone	39	0	17	8,944
			n=	30,600

Using hardware acceleration, we attained repeatable periodicity of 200ms for data capture, transfer, computation, and classification, making for an approximate 220ms trip-to-trigger time when using a mechanical interruption relay. The power consumption slightly increased to 2.6W, resulting in a slightly increased annual operating cost of $2.72 at $0.12 per kilowatt-hour. With a different ASIC capturing 6kHz signals instead of 48kHz and Digital Signal Processing (DSP), we believe this can be reduced.

In summary, the hardware-accelerated setup works reliably and efficiently to differentiate normal and abnormal circuits or to identify specific loads.

More generally, we demonstrated the ability to learn new device signatures and to deploy computationally-intensive classification on constrained devices. Shifting artificial intelligence into commodity hardware is a significant step toward developing objects with true "embedded intelligence." We hope this paper helps practitioners bring AI into increasing numbers of connected, low-power and low-cost devices, building a smarter, more interconnected future.

9. Final Model Weights and Neuron Activation

To determine whether our generated features are being used and therefore worth generating, we plotted each feature against its learned weight. Figure 10 shows the first filter layer feature weights for the multi-state classifier. The x-axis indicates the feature, while the y-axis indicates different neurons. The weights are represented using the color bar on the right hand side.

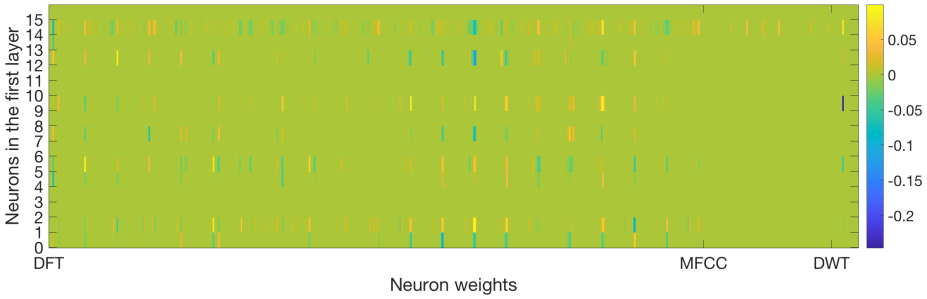

Figure 10: This figure shows the first layer feature weights for the multi-state classifier, plotted by feature vs neuron number. The weights are color coded using the scale on the right side.

Note that while most weights are near 0 as expected, we see significant weighting of certain features. This is particularly true for the DFT features, which are the most utilized, followed by the MFCC features. These DFT features cluster into a few bands, with wider banding and heaver weights towards the higher frequency region (the right-most DFT features). Few DWT features have significant weighting.

In Figure 11, we show the activation results beyond the first layer for randomly selected samples from the burner, fan, iMac, and ozone generator. Note the similarities between the burner (pure resistive load, [a]-[c]) and the fan (DC motor with minimal commutator arcing. [g]-[i]).

10. Conclusion and Outlook

We successfully demonstrated a real-time implementation of a DNN for arc fault detection on constrained hardware. This approach exceeds the accuracy (99.95% vs. 99%) of commercial and academic AFCIs and matches their latency.

In future iterations, we will implement a decaying learning rate to limit model learning computation time, allowing constrained nodes to locally adapt while awaiting fresh "master" signatures from the Cloud. We will also explore whether feature ranking[18] can improve robustness while requiring a subset of features that can be generated more efficiently.[22] In the near term, we will

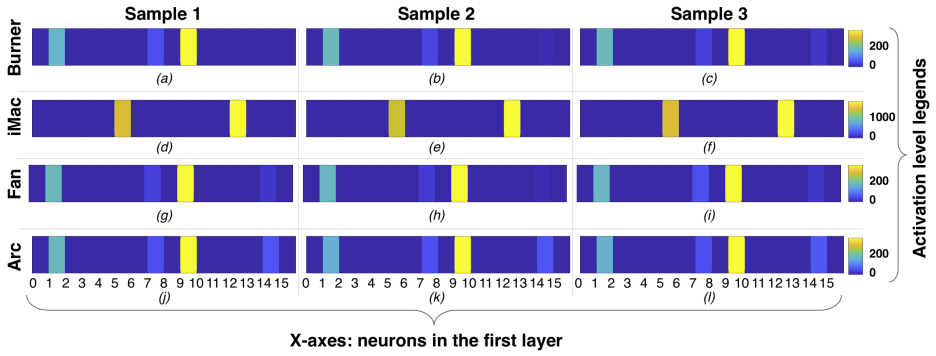

Figure 11: These images show the activation function for three random samples of each device class. (a)-(c) represent a pure resistive load (the burner), (d)-(f) represent the iMac's switching power supply, (g)-(i) represent the fan, and (j)-(l) represent arcing.

examine the classifier's sensitivity to window size, which could improve model accuracy or reduce classification time.

As our next steps, we will modify the algorithm to adaptively learn and to support over-the-air updates on a separate thread, so devices that cause nuisance trips (switch-mode power supplies, DC motors, and similar) can be identified and the likelihood of inadvertent interruption reduced.

There are opportunities to reduce system cost while improving functionality. Adding a second current sensor will allow the same device to detect parallel arcs, as well as to support ground-fault detection. With additional sound cards, multiple circuits may be monitored from a power distribution box. Software tweaks will allow outlets to be controlled remotely, improving homeowner convenience.

Another area for exploration is user-specific risk tolerance models. Even at a high accuracy, the system may generate a false negative sample (reporting "normal" when the system is actually "abnormal.") In this case, the device could trigger on the next-detected "abnormal" event, or, based on the user's preferences, wait for a set number of consecutive abnormal events or events within a timespan to disconnect the circuit. Risk models could be derived from aggregate data and actuarial tables, dynamically updated. A user could be notified of heightened risk through a mobile application when a single fault is

triggered, choosing to override interruption while assuming liability for damage.

Smart AFCIs will become valuable data collection tools, helping precisely identify the signatures of electronic devices while generating house- or city-wide data useful for mapping power consumption, identifying maintenance needs, and more. Our hope is that once end-users realize the value of embedded intelligence, that such techniques will find their ways into all aspects of life, starting from mundane devices.

References

[1] Gregory, G.D., Scott, G.W.. The arc-fault circuit interrupter, an emerging product. In: Industrial and Commercial Power Systems Technical Conference, 1998 IEEE. IEEE; 1998, p. 48–55.

[2] Gregory, G.D., Wong, K., Dvorak, R.. More about arc-fault circuit interrupters. In: Conference Record of the Industry Applications Conference; vol. 2. IEEE; 2003, p. 1306–1313.

[3] Li, C., Dawson, F., Kojori, H., Meyers, C., Yue, E.. Arc fault detection and protection - opportunities and challenges. Tech. Rep.; Society of Automotive Engineers; 2003. URL: https://www.sae.org/publications/technical-papers/content/2003-01-3037/. doi:10.4271/2003-01-3037.

[4] Johnson, J., Gudgel, B., Meares, A., Fresquez, A.. Series and parallel arc-fault circuit interrupter tests. Tech. Rep.; Sandia Natational Laboratories, Albuquerque, NM, USA; 2013. URL: http://energy.sandia.gov/wp-content/gallery/uploads/SAND2013-5916-SeriesAndParallelArc-FaultCircuitInterrupterTests1.pdf.

[5] Li, W.J., Li, Y.C.. Arc fault detection based on wavelet packet. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics; vol. 3. IEEE; 2005, p. 1783–1788.

[6] Liu, G., Cao, Y.N., Liu, Y., Liu, Z.. A survey on arc fault detection and wire fault location for aircraft wiring systems. SAE International Journal of Aerospace 2008;1(2008-01-2870):903–914.

[7] Ma, S., Guan, L.. Arc-fault recognition based on BP neural network. In: Third International Conference on Measuring Technology and Mechatronics Automation; vol. 1. IEEE; 2011, p. 584–586.

[8] Liu, X., Liu, X., Hou, C., Leng, X., Lai, Z.. Arc fault diagnosis and analysis based on wavelet neural network. In: 1st International Conference on Electric Power Equipment-Switching Technology. IEEE; 2011, p. 187–190.

[9] Kai, Y., Rencheng, Z., Jianhong, Y., Jianhua, D., Shouhong, C., Ran, T.. Series arc fault diagnostic method based on fractal dimension and Support Vector Machine. Transactions of China Electrotechnical Society 2016;31(2):70. URL: http://www.ces-transaction.com//EN/abstract/article_3596.shtml.

[10] Roverso, D.. Plant diagnostics by transient classification: The ALADDIN approach. International Journal of Intelligent Systems 2002;17(8):767–790. doi:10.1002/int.10049.

[11] Kwon, K., Kim, J., Seong, P.. Hidden Markov model-based real-time transient identifications in nuclear power plants. International journal of intelligent systems 2002;17(8):791–811.

[12] Zia, T., Bruckner, D., Zaidi, A.. A hidden Markov model based procedure for identifying household electric loads. In: 37th Annual Conference of the IEEE Industrial Electronics Society. 2011, p. 3218–3223. doi:10.1109/IECON.2011.6119826.

[13] Cheng, H., Chen, X., Liu, F., Wang, C.. Series arc fault detection and implementation based on the short-time fourier transform. In: Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific. IEEE; 2010, p. 1–4.

[14] Li, D., Song, Z., Wang, J., Geng, Y., Chen, H., Yu, L., et al. A method for residential series arc fault detection and identification. In: Proceedings of the 55th IEEE Holm Conference on Electrical Contacts. IEEE; 2009, p. 8–14.

[15] Yang, K., Zhang, R., Yang, J., Liu, C., Chen, S., Zhang, F.. A novel arc fault detector for early detection of electrical fires. Sensors 2016;16(4). doi:10.3390/s16040500; basel, Switzerland.

[16] Ming, Z., Tian, Y., Zhang, F.. Design of arc fault detection system based on CAN bus. In: International Conference on Applied Superconductivity and Electromagnetic Devices. IEEE; 2009, p. 308–311.

[17] Koziy, K., Gou, B., Aslakson, J.. A low-cost power-quality meter with series arc-fault detection capability for smart grid. IEEE Transactions on Power Delivery 2013;28(3):1584–1591.

[18] Siegel, J.E., Kumar, S., Ehrenberg, I., Sarma, S.E.. Engine misfire detection with pervasive mobile audio. In: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Cham: Springer International Publishing. ISBN 978-3-319-46131-1; 2016, p. 226–241. doi:10.1007/978-3-319-46131-1_26.

[19] Siegel, J.E., Bhattacharyya, R., Kumar, S., Sarma, S.E.. Air filter particulate loading detection using smartphone audio and optimized ensemble classification. Engineering Applications of Artificial Intelligence 2017;66:104–112. URL: https://www.sciencedirect.com/science/article/abs/pii/S0952197617302294. doi:10.1016/j.engappai.2017.09.015.

[20] Kingma, D.P., Ba, J.. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980 2014;.

[21] LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.. Efficient backprop. In: Neural networks: Tricks of the trade. Springer; 1998, p. 9–50.

[22] Goertzel, G.. An algorithm for the evaluation of finite trigonometric series. The American Mathematical Monthly 1958;65(1):34–35.