

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

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

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

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

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

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

21 To prove this concept’s feasibility, we examine low-power, AC series arcs,
22 which provide worst-case training data. AC faults are difficult to classify because
23 the circuit’s connected load limits the arc’s maximum current[3, 1], reducing the
24 signal-to-noise ratio. Series faults pose a high likelihood of confusion with benign
25 appliances such as DC motors, and by testing with low-power circuits, (< 15A
26 @ 120VAC), our algorithms will readily extend to higher-power arcs.

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

36 2. Existing Arc Detectors

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

40 AFCIs may rely upon analog circuits, application-specific integrated circuits
41 (ASICs), field-programmable gate arrays (FPGAs), or optical and electromag-
42 netic techniques to detect arcs. Device sensitivities and reaction times (25-
43 250ms) vary[4, 5].

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

47 Algorithmic detectors face different challenges. Arcs are dynamic, and de-
48 tection efficacy varies as the cathode erodes[2]. Appliances may share charac-
49 teristics with arcs, including current shoulders, a change in amplitude, or an
50 increased rate of current rise[2], leading to misclassification.

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

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

61 Hidden Markov Models (HMMs) learn time-dependent spatial and temporal
62 patterns to identify state transitions from normal to abnormal plant opera-
63 tion. [11] Similar HMMs identify individual appliances from combined electri-
64 cal loads, but require supervised model creation and long inter-state transitions
65 not conducive to the fast ($> 1\text{Hz}$) realtime operation required to protect against

66 arc-related fires[12].

67 Computer-controlled AFCI's running these algorithms may rely upon fea-
68 tures including Fourier coefficients, wavelets, and use techniques such as band-
69 pass filtering to eliminate harmonics and baseline current from measurements[6,
70 13, 14]. Other approaches derive features by correlating multiple information
71 sources, for example by relating differential current ($\frac{di}{dt}$) to absolute current ($|i|$),
72 which improves separability of nuisance tripping from fault tripping. These ap-
73 proaches may detect early arcing with up to 98% accuracy[15].

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

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

84 **3. Hypothesis**

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

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

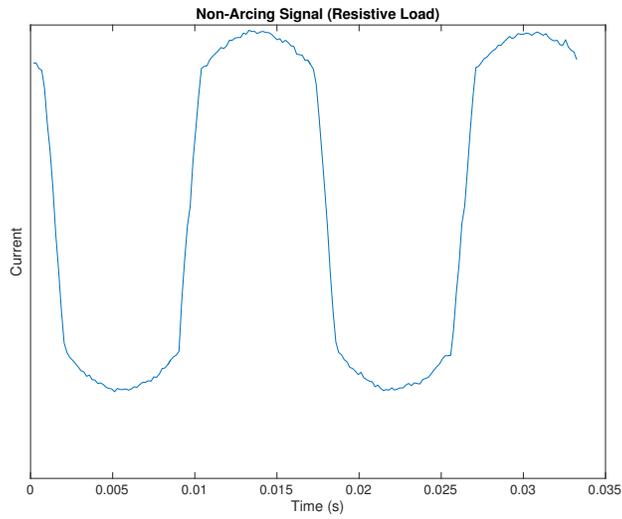


Figure 1: This figure shows a typical smooth and periodic current trace for a resistive electrical load.

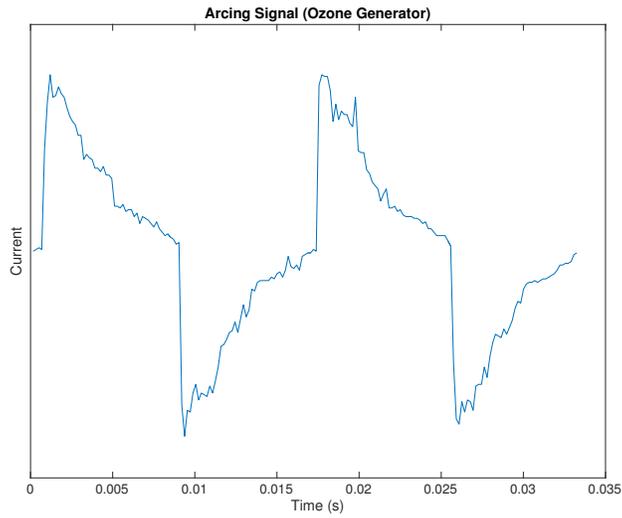


Figure 2: In this arcing current trace, note the shoulders as the electrical potential climbs before creating an arc.

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

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

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

101 **4. Experimental Setup**

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

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

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

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

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

125 Current data were recorded on a Raspberry Pi Model 3 microcomputer

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

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

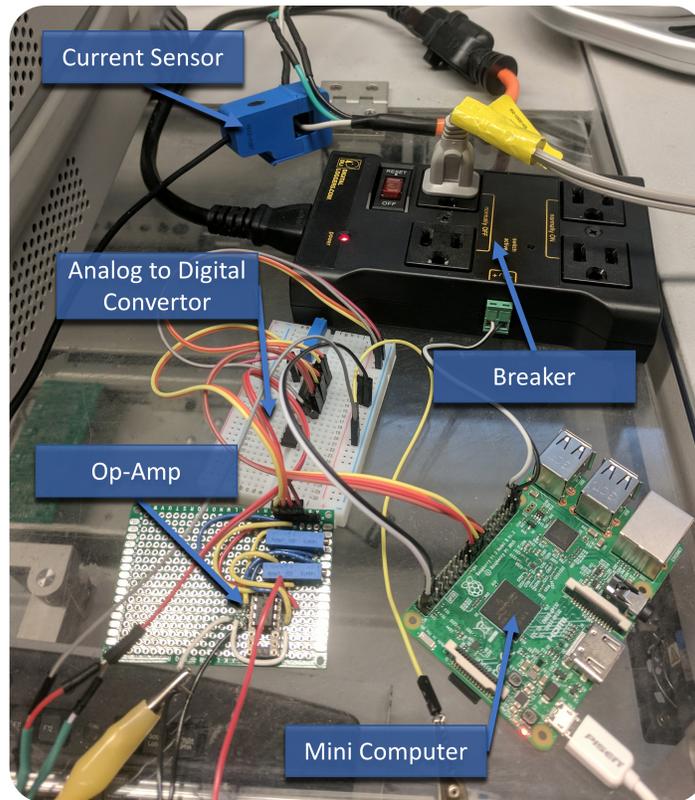


Figure 3: This image shows the experimental setup for data collection (and later, for testing real-time classification).

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

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

140 **5. Proof of Concept**

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

144 *5.1. Feature Generation*

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

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

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

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

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

163 Finally, these features were concatenated to form one vector comprising
164 DFT, MFCC and DWT features ($F_{ALL} = F_{DFT} \parallel F_{MFCC} \parallel F_{DWT}$) for each sam-
165 ple current measurement, where \parallel represents vector concatenation.

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

168 5.2. Initial DNN Model

169 To test neural networks’ viability on constrained hardware, we implemented
170 a DNN classifier. Bringing “intelligence to the edge” allows durable goods and
171 infrastructure devices to adapt in the face of new data, and DNNs do not re-
172 quire excessive engineering, even for multi-state classification which non-linear
173 separability.

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

179 All the three hidden layers are Fully-Connected (FC) layers: each neuron
180 connects all the neurons in its previous layer with learnable weights. The re-
181 sponse of a hidden layer unit is calculated by summing over the product of each
182 input signal and its corresponding weight, and passing the summation through
183 a Rectified Linear Unit (ReLU). For the output layer, Sigmoid function is used
184 as the activation function for the binary classification case, and the Softmax
185 function is used for multi-state classification.

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

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

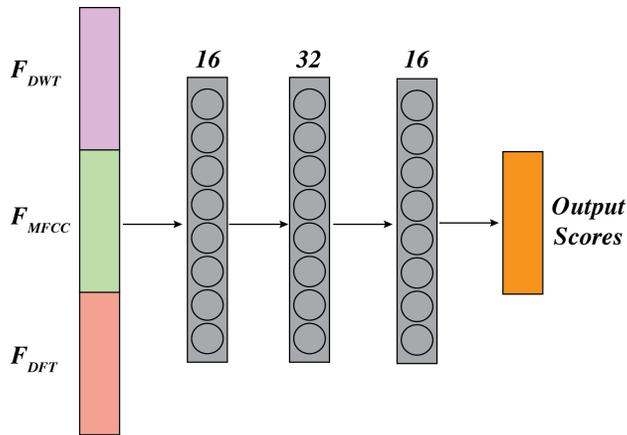


Figure 4: The input to the DNN is a 1D vector comprising $F_{DFT}\|F_{MFCC}\|F_{DWT}$. The model has three hidden layers, with 16, 32 and 16 neurons. The model generates a probability distribution over predefined classes to determine normal versus abnormal circuit operation, as well as identifying specific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

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

197 **6. Early Results and Refinement**

198 This first DNN returned 99.98% classification accuracy on in-sample data
 199 using a one-versus-all approach (ozone [abnormal] versus all other input data
 200 [normal]). With a 0.2s data segment, these results represent classification on
 201 12,357 unique events representing a roughly 30/70%, with the confusion matrix
 202 shown in Table 1.

203 The false positive rate (good circuit reported as faulty) was 0.01%, almost
 204 eliminating nuisance trips. The false negative rate (bad circuit reported as good)
 205 was 0.05%, suggesting the algorithm may need tuning or biasing to minimize
 206 fire risk. The sensitivity (when an arc is present, how often does the algorithm
 207 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

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

211 Categorizing by device type, the DNN returned a four-class accuracy of
 212 97.46%. The resulting confusion matrix is shown in Table 2.

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

217 We then verified that the model could improve over time by incorporating
 218 15 minutes of additional training data from the same devices. The in-sample
 219 binary performance reached 100% accuracy, and multi-state reached 99.98%.
 220 We validated that the improved model was not overfit by running the classifier
 221 on 10 hours of data from an outsample non-arcing load, with no false positives.

222 *6.1. Definition Updatability*

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

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

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

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

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

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

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

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

261 **7. Building Towards a Connected AFCI**

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

264 The Internet of Things is key to learning and deploying new classification
265 models, ensuring that real-world edge cases make it into training data. Ag-
266 gregated fingerprints and centralized model training allows scalable Cloud re-
267 sources to be leveraged by infrastructure with growing “embedded intelligence.”
268 Connectivity and distributed computation allow constrained microcomputers to
269 participate in on-line and adaptive learning, helping crowdsource classifier devel-
270 opment and enabling remote control that incentivizes the installation of costly
271 “smart” switches.

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

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

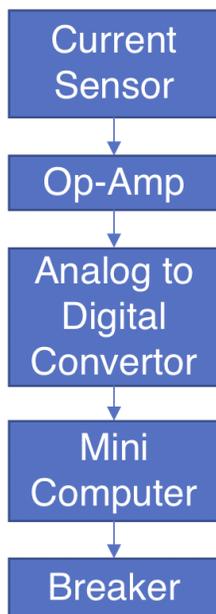


Figure 5: This block diagram shows the elements of the connected AFCI’s data collection and outlet control system.

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

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

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

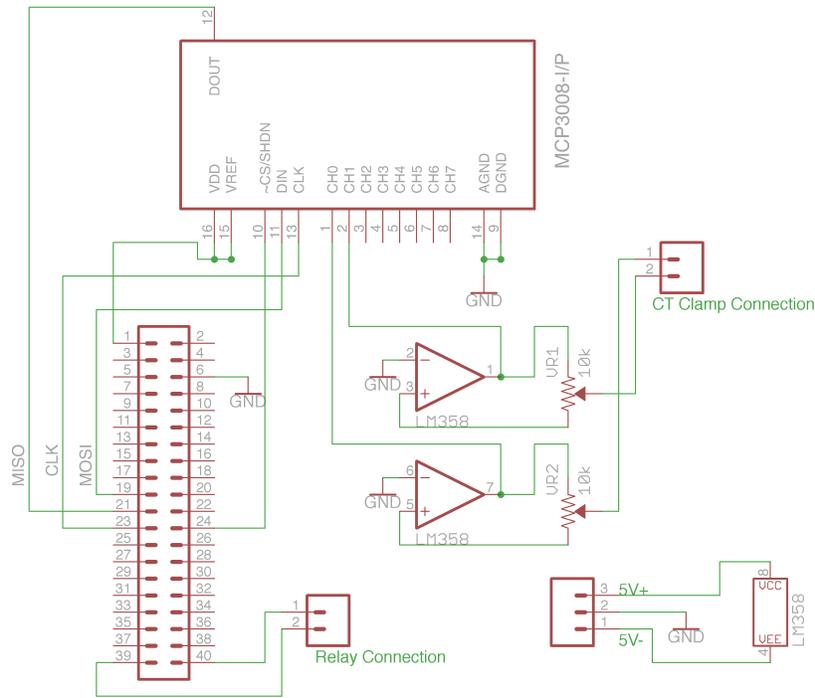


Figure 6: This schematic shows the MCP3008 ADC, a dual LM358 amplification circuit, the CT sensor input, and the Raspberry Pi SPI interface connections.

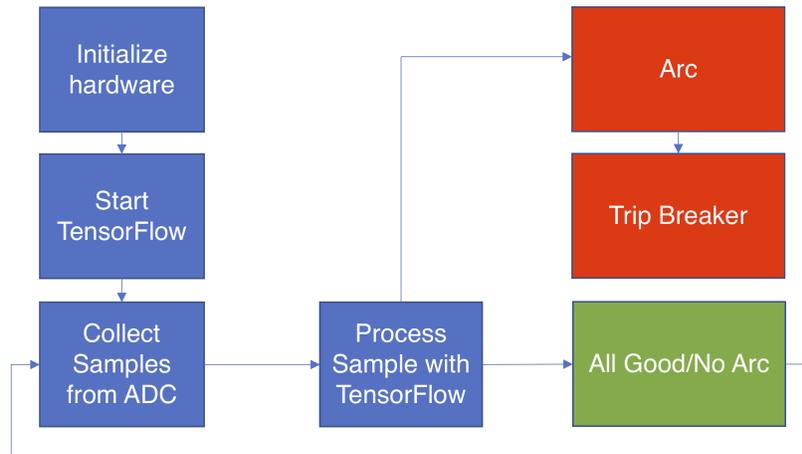


Figure 7: This diagram shows a simplified process flow for arc fault classification.

295 is competitive today’s APCI’s.

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

301 *7.1. Software Challenges*

302 The Raspberry Pi does not run a realtime operating system, making deter-
303 ministic timing impossible. We analyzed our “real-time” data and found that
304 for a target frequency of 5865Hz, the system clock stability had a standard devi-
305 ation of 3%, causing substantial frequency variation (176Hz). As the processor
306 cores loaded with data acquisition, OS functions, and classification, the gener-
307 ated FFT features suffered. This non-uniformity dramatically increased false
308 positives and negatives.

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

315 **8. Hardware-Accelerated Classification**

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

322 USB sound cards feature integrated clock timing controllers, high-precision
323 analog-to-digital converters, buffered storage, and automated gain compensa-
324 tion to simplify both the hardware and software necessary for data collection.
325 The USB soundcard we used allowed sampling up to 48kHz at 16-bits of reso-
326 lution, and enabled us to eliminate the ADC hardware, op-amp and dual power
327 supply from the AFCI’s bill of materials, reducing cost and simplifying manu-
328 facturability.

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

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

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

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

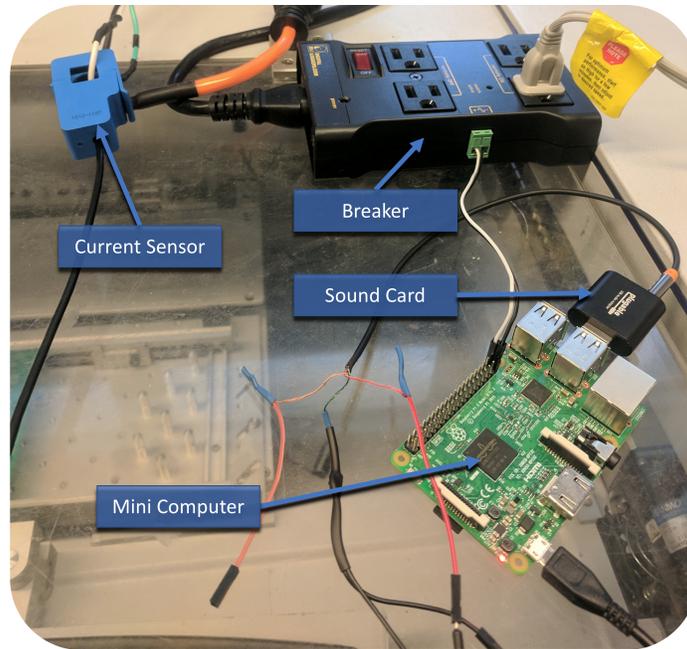


Figure 8: This image shows the experimental setup for data collection using a USB soundcard for hardware acceleration.

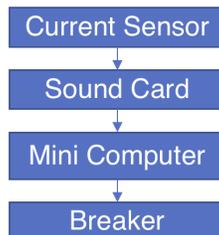


Figure 9: This block diagram shows the elements of the hardware-accelerated connected AFCI's data collection and outlet control system.

350 We repeated the data collection process described in Section 4 and captured
 351 at least an hour and 15 minutes of data for each input type, and captured more
 352 data from the ozone generator to keep classes relatively balanced. We also took
 353 this opportunity to make changes to the model based on our observations about
 354 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

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

360 Using a grid search for optimization, we increased the DFT’s upper fre-
 361 quency limit to 2,500Hz to create additional differentiating features. We did not
 362 increase the feature limit to 24kHz to ensure quick computation and avoid sam-
 363 pling near the capture hardware’s upper limit, where jitter would have more im-
 364 pact. We used a floor of 40Hz to eliminate low-frequency components. The final
 365 feature vector length was $n_{FALL} = 607$, comprising $n_{DFT} = 491$, $n_{MFCC} = 96$,
 366 and $n_{DWT} = 20$.

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

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

373 This model’s binary outsample accuracy is 99.95% with a false positive rate
 374 of 0.004% (once per 83 minutes). Device identification accuracy is 95.61% with
 375 a false positive rate of 1.8%. The model’s primary misclassification mode is
 376 between the fan and the burner, a non-malignant mistake. With continued
 377 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

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

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

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

393 **9. Final Model Weights and Neuron Activation**

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

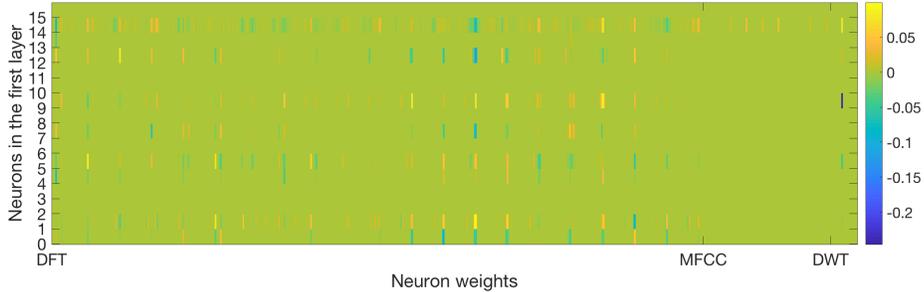


Figure 10: This figure shows the first layer feature weights for the multi-state classifier, plotted by feature vs neuron number. The weights are color coded using the scale on the right side.

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

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

409 10. Conclusion and Outlook

410 We successfully demonstrated a real-time implementation of a DNN for arc
 411 fault detection on constrained hardware. This approach exceeds the accuracy
 412 (99.95% vs. 99%) of commercial and academic AFCIs and matches their latency.

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

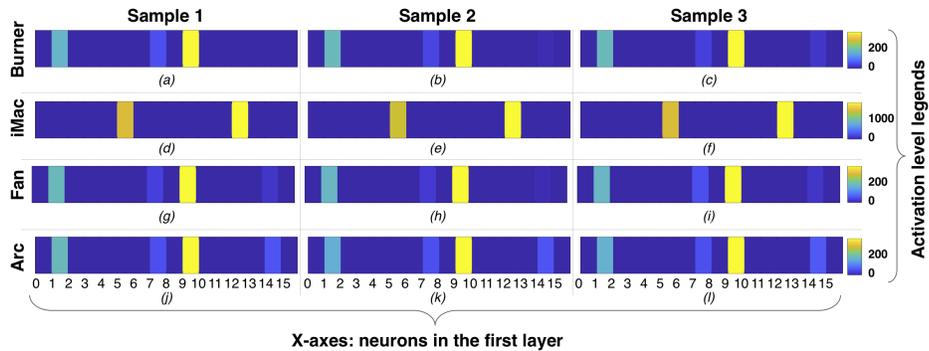


Figure 11: These images show the activation function for three random samples of each device class. (a)-(c) represent a pure resistive load (the burner), (d)-(f) represent the iMac’s switching power supply, (g)-(i) represent the fan, and (j)-(l) represent arcing.

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

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

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

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

437 triggered, choosing to override interruption while assuming liability for damage.
438 Smart AFCIs will become valuable data collection tools, helping precisely
439 identify the signatures of electronic devices while generating house- or city-wide
440 data useful for mapping power consumption, identifying maintenance needs, and
441 more. Our hope is that once end-users realize the value of embedded intelligence,
442 that such techniques will find their ways into all aspects of life, starting from
443 mundane devices.

444 **References**

- 445 [1] Gregory, G.D., Scott, G.W.. The arc-fault circuit interrupter, an emerging
446 product. In: Industrial and Commercial Power Systems Technical Confer-
447 ence, 1998 IEEE. IEEE; 1998, p. 48–55.
- 448 [2] Gregory, G.D., Wong, K., Dvorak, R.. More about arc-fault circuit inter-
449 rupters. In: Conference Record of the Industry Applications Conference;
450 vol. 2. IEEE; 2003, p. 1306–1313.
- 451 [3] Li, C., Dawson, F., Kojori, H., Meyers, C., Yue, E.. Arc
452 fault detection and protection - opportunities and challenges. Tech.
453 Rep.; Society of Automotive Engineers; 2003. URL: [https://www.
454 sae.org/publications/technical-papers/content/2003-01-3037/](https://www.sae.org/publications/technical-papers/content/2003-01-3037/).
455 doi:10.4271/2003-01-3037.
- 456 [4] Johnson, J., Gudgel, B., Meares, A., Fresquez, A.. Se-
457 ries and parallel arc-fault circuit interrupter tests. Tech. Rep.;
458 Sandia Natational Laboratories, Albuquerque, NM, USA; 2013.
459 URL: [http://energy.sandia.gov/wp-content/gallery/uploads/
460 SAND2013-5916-SeriesAndParallelArc-FaultCircuitInterrupterTests1.
461 pdf](http://energy.sandia.gov/wp-content/gallery/uploads/SAND2013-5916-SeriesAndParallelArc-FaultCircuitInterrupterTests1.pdf).
- 462 [5] Li, W.J., Li, Y.C.. Arc fault detection based on wavelet packet. In:
463 Proceedings of 2005 International Conference on Machine Learning and
464 Cybernetics; vol. 3. IEEE; 2005, p. 1783–1788.
- 465 [6] Liu, G., Cao, Y.N., Liu, Y., Liu, Z.. A survey on arc fault detection and
466 wire fault location for aircraft wiring systems. SAE International Journal
467 of Aerospace 2008;1(2008-01-2870):903–914.
- 468 [7] Ma, S., Guan, L.. Arc-fault recognition based on BP neural network. In:
469 Third International Conference on Measuring Technology and Mechatronics
470 Automation; vol. 1. IEEE; 2011, p. 584–586.

- 471 [8] Liu, X., Liu, X., Hou, C., Leng, X., Lai, Z.. Arc fault diagnosis and
472 analysis based on wavelet neural network. In: 1st International Conference
473 on Electric Power Equipment-Switching Technology. IEEE; 2011, p. 187–
474 190.
- 475 [9] Kai, Y., Rencheng, Z., Jianhong, Y., Jianhua, D., Shouhong, C., Ran,
476 T.. Series arc fault diagnostic method based on fractal dimension and
477 Support Vector Machine. Transactions of China Electrotechnical Society
478 2016;31(2):70. URL: [http://www.ces-transaction.com//EN/abstract/
479 article_3596.shtml](http://www.ces-transaction.com//EN/abstract/article_3596.shtml).
- 480 [10] Roverso, D.. Plant diagnostics by transient classification: The ALADDIN
481 approach. International Journal of Intelligent Systems 2002;17(8):767–790.
482 doi:10.1002/int.10049.
- 483 [11] Kwon, K., Kim, J., Seong, P.. Hidden Markov model-based real-time
484 transient identifications in nuclear power plants. International journal of
485 intelligent systems 2002;17(8):791–811.
- 486 [12] Zia, T., Bruckner, D., Zaidi, A.. A hidden Markov model based procedure
487 for identifying household electric loads. In: 37th Annual Conference of the
488 IEEE Industrial Electronics Society. 2011, p. 3218–3223. doi:10.1109/
489 IECON.2011.6119826.
- 490 [13] Cheng, H., Chen, X., Liu, F., Wang, C.. Series arc fault detection
491 and implementation based on the short-time fourier transform. In: Power
492 and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific. IEEE;
493 2010, p. 1–4.
- 494 [14] Li, D., Song, Z., Wang, J., Geng, Y., Chen, H., Yu, L., et al. A method
495 for residential series arc fault detection and identification. In: Proceedings
496 of the 55th IEEE Holm Conference on Electrical Contacts. IEEE; 2009, p.
497 8–14.

- 498 [15] Yang, K., Zhang, R., Yang, J., Liu, C., Chen, S., Zhang, F.. A novel
499 arc fault detector for early detection of electrical fires. *Sensors* 2016;16(4).
500 doi:10.3390/s16040500; basel, Switzerland.
- 501 [16] Ming, Z., Tian, Y., Zhang, F.. Design of arc fault detection system based
502 on CAN bus. In: *International Conference on Applied Superconductivity
503 and Electromagnetic Devices*. IEEE; 2009, p. 308–311.
- 504 [17] Koziy, K., Gou, B., Aslakson, J.. A low-cost power-quality meter with
505 series arc-fault detection capability for smart grid. *IEEE Transactions on
506 Power Delivery* 2013;28(3):1584–1591.
- 507 [18] Siegel, J.E., Kumar, S., Ehrenberg, I., Sarma, S.E.. Engine misfire detec-
508 tion with pervasive mobile audio. In: *Proceedings of European Conference
509 on Machine Learning and Principles and Practice of Knowledge Discovery
510 in Databases*. Cham: Springer International Publishing. ISBN 978-3-319-
511 46131-1; 2016, p. 226–241. doi:10.1007/978-3-319-46131-1_26.
- 512 [19] Siegel, J.E., Bhattacharyya, R., Kumar, S., Sarma, S.E.. Air filter
513 particulate loading detection using smartphone audio and optimized en-
514 semble classification. *Engineering Applications of Artificial Intelligence*
515 2017;66:104–112. URL: [https://www.sciencedirect.com/science/
516 article/abs/pii/S0952197617302294](https://www.sciencedirect.com/science/article/abs/pii/S0952197617302294). doi:10.1016/j.engappai.2017.
517 09.015.
- 518 [20] Kingma, D.P., Ba, J.. Adam: A method for stochastic optimization.
519 arXiv preprint arXiv:14126980 2014;.
- 520 [21] LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.. Efficient backprop. In:
521 *Neural networks: Tricks of the trade*. Springer; 1998, p. 9–50.
- 522 [22] Goertzel, G.. An algorithm for the evaluation of finite trigonometric series.
523 *The American Mathematical Monthly* 1958;65(1):34–35.