Supervised Deep Learning and Classification of Single-Event Transients

T. Peyton, *Student Member, IEEE,* J. L. Carpenter, *Student Member, IEEE,* S. Camp, *Student Member, IEEE,* M. Fadul, *Member, IEEE,* B. Dean, *Student Member, IEEE,* D. R. Reising, *Senior Member, IEEE* and T. D. Loveless, *Senior Member, IEEE*

***Abstract*—This article describes a method to detect and classify single-event transients (SET) to determine the originating circuit node impacted by ionizing radiation. SETs were measured via two-photon absorption (TPA) laser excitation on a custom CMOS phase-locked loop (PLL), and Convolutional Neural Networks (CNN) were used to classify the spatial dependencies of the transient responses. A clustering technique is described to identify groups of related circuit nodes and achieves over 90% identification accuracy.**

***Index Terms*—Supervised machine learning, Convolutional neural network, Single event transients, Radiation effects, Phase locked loop**

I. Introduction

MACHINE Learning (ML) has emerged in the field of radiation effects for its ability to identify anomalous behavior during experimentation [1]–[8], for rate prediction [9]–[13], and for aiding in radiation effects mitigation [1], [14]–[16]. Radiation effects in ML-enabled hardware are also being studied [17], [18]. K-Nearest Neighbors (KNN), a clustering ML algorithm, was one of the first techniques used to detect the presence of single-event transients (SET) within complex RF waveforms with a binary classifier (*i.e.*, the models either detected an SET or did not) [7], [16]. Other work uses a similar classifier for determining the state of a device having accumulated total ionizing dose (TID) [1].

This article describes a deep learning method using Convolutional Neural Networks (CNN) to detect SETs and classify the results based on the originating circuit node. Data obtained from irradiation of a CMOS phase-locked loop (PLL) with a two-photon absorption (TPA) laser was used to train and validate the CNN model. An initial 8-bin classifier was used to segment observed phenomena. A method of sorting the resulting confusion matrix is illustrated to reduce the number of classes by identifying confusion groups. These confusion groups correlate to specific electrically connected nodes within the circuit under study. Over 90% accuracy in identifying the correct circuit node impacted by the radiation is achieved with the reduced multi-classifier. This work documents the rapid advancements possible with artificial intelligence (AI) applications in radiation effects. The ability to identify precise locations within an arbitrarily complex circuit based purely on observed behavior paves the way for novel advancements in real-time data analysis, improved error rate analyses, and dynamic mitigation approaches.

This work was supported in part by the Defense Threat Reduction Agency under award number HDTRA1-17-1-0003 and the DoD through the SCALE Microelectronics Program and the NSF REU Program under award number 1757777.

Authors are with the Electrical Engineering Department, University of Tennessee at Chattanooga, 615 McCallie Ave., Dept. 2342, Chattanooga, TN, 37403, USA; Phone: 423-425-2353, Fax: 423-425-5229, Email: qtx464@mocs.utc.edu

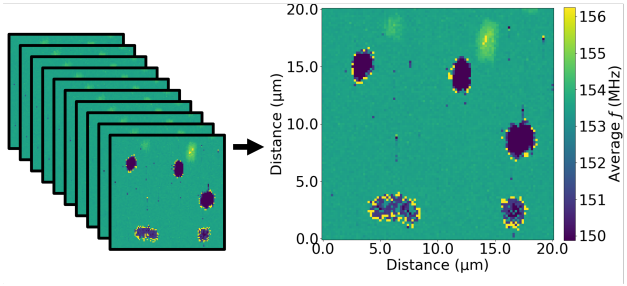

Fig. 1: Ten 20 μm by 20 μm spatial maps displaying the sensitivity of the CP circuit to SETs, measured by the average frequency at the output of the PLL during laser exposure at each x-y position. Data were obtained via laser TPA experiments with a spatial resolution of 0.2 μm, producing the 100x100 pixel images [19].

II. Background

A common challenge with ML is that a significant, often unavailable quantity of data is required to train valid models. These data used in this work have been thoroughly validated and published in theoretical [19], simulation and modeling [20], and radiation-hardening-by-design (RHBD) studies [20], [21]. In addition, in recent years, these data have been used in benchmark studies for ML-driven radiation effects analyses due in part to the large number of experimentally measured samples to aid in model training [4], [7]. Specifically, in [4], [7], supervised ML using a K-Nearest Neighbors (KNN) algorithm was used to develop a binary classifier for determining the presence of a SET within a waveform. However, KNN requires storing all training data and is difficult to scale. Consequently, it can be computationally expensive to evaluate new data with KNN. This effort uses a supervised deep learning technique that leverages artificial neural networks to develop a model for analyzing radiation effects data.

A. Device Under Test

A mixed-signal PLL circuit, fabricated in the IBM 130 nm CMRF8RF CMOS technology [21], was selected as the device under test (DUT). The device contains a phase-frequency

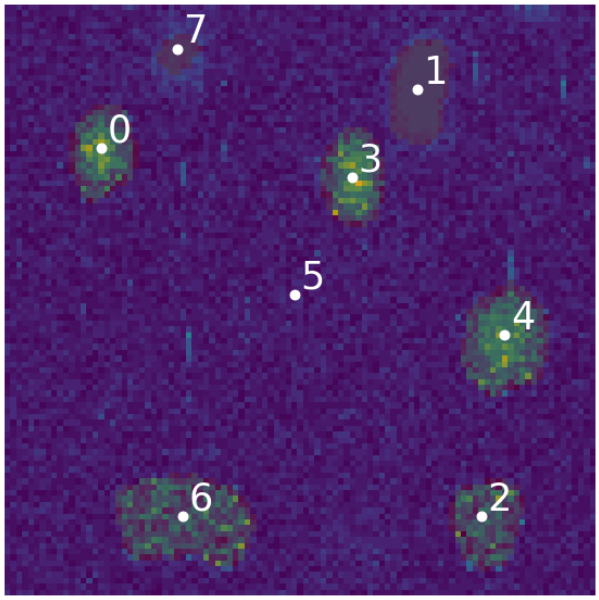

Fig. 2: The 20 µm by 20 µm spatial map of the sensitivity of the CP circuit to SETs, measured as the relative frequency perturbations at the output of the PLL, with assigned cluster labels. Labels 0-7 were automatically determined from image segmentation using thresholding and binary morphology operations.

detector (PFD), a charge pump (CP), a low-pass filter (LPF), and a voltage-controlled oscillator (VCO), which has a center frequency of 200 MHz and a maximum frequency of 530 MHz. The locking range of the PLL is between 40 and 350 MHz, with a gain of 7.75 GHz/V.

A 20 µm by 20 µm area of the CP circuit was exposed to radiation using a two-photon absorption (TPA) laser with a diameter of 1.1 µm (measured at $1/e$ of the peak intensity) at the Naval Research Laboratory. The device was mounted on a motorized xyz translation platform and moved by 0.2 µm steps in the x- and y-axes. The z-axis remained fixed during the entire experiment after determining the worst-case response. Waveforms were recorded at the PLL circuit's output at 10,000 unique strike locations for a 100x100 pixel image, as shown in Fig. 1 where the average frequency at the output of the PLL during laser exposure is visualized at each x-y position. In addition, ten unique SETs were recorded at each strike location resulting in a total of 100,000 transients partitioned into ten spatial sensitivity maps [19].

The initial spatial sensitivity map shown in Fig. 1, illustrating the relative frequency perturbations measured at the output of the PLL, was processed using thresholding and binary morphology operations. Connected component labeling was used to automatically assign unique cluster labels for each sensitive region, as shown in Fig. 2. The algorithm identified eight unique segmented components, including the background (*i.e.*, responses indistinguishable from noise), which was initially assigned to group 5. Each cluster, representing a sensitive region in the CP, has an average of 180 strike points for 1,800 transient waveforms within the ten spatial sensitivity maps. Each waveform consists of 500 sample points of the instantaneous frequency measured every 1.6 ns. Example waveforms in this data set can be seen in Fig. 3 and are detailed in [19].

B. Data Preparation

These data were split into a training set and a testing set; 80% of these data (*i.e.,* eight spatial maps) were used to train the CNN models, and the remaining 20% of these data (*i.e.,* two spatial maps) were used to test the accuracy of the models in evaluating new, unseen data. This partition resulted in up to 80,000 possible SETs in the training dataset and 20,000 possible SETs for the testing dataset. The split was stratified, meaning the same proportion of each cluster from the entire dataset was provided to the training and testing datasets. The training portion was used to update models' weights (*i.e.,* filter coefficients), while the testing portion was reserved to evaluate the models.

Additionally, the training portion used k-fold cross-validation to estimate the models' ability to process new data, with 5-folds as illustrated in Fig. 4. The entire training dataset was split into five segments, and five separate but identical models were trained, each with a single segment used for validation and the rest for training. The validation segment determines how fast, measured in the number of epochs or training cycles, the model learns against data it has not seen before and can be compared with the training data's accuracy. Validating data is essential in tracking the model's progress during training and preventing overfitting. After overfitting occurs, any further optimization is specific to the training data and not a generalization of new information.

Further, k-fold cross-validation gives insight into any biases the dataset may contain during training. If, for example, one model overfits quickly compared to the others, it indicates that a portion of the dataset is unbalanced and does not reflect the rest of the dataset. Conversely, if the validation portion always performs poorly, then that portion is not adequately represented by the rest of the training data. The five models have comparable performance in this case.

An additional use case for k-fold cross-validation is model selection. By comparing each model's output against the shared test dataset, the best model can be used to give a slight advantage. However, since the dataset was very balanced, the differences in accuracy in the final models were marginal, so the first model was used.

III. CNN Model

A. Convolutional Neural Network

Convolutional Neural Networks (CNNs) are supervised learning Multi-Layer Perceptron (MLP)-based Neural Networks designed to extract and learn features from multi-dimensional data [22]. CNNs can process one-dimensional (1D) data such as time-interval vectors and two-dimensional and three-dimensional grids of values such as image pixels [22]. CNNs are trained using a back-propagation algorithm to estimate parameters that minimize a specific objective function. The objective function measures the error between the CNN model's output (prediction) and the ground truth and is referred to as a Loss function.

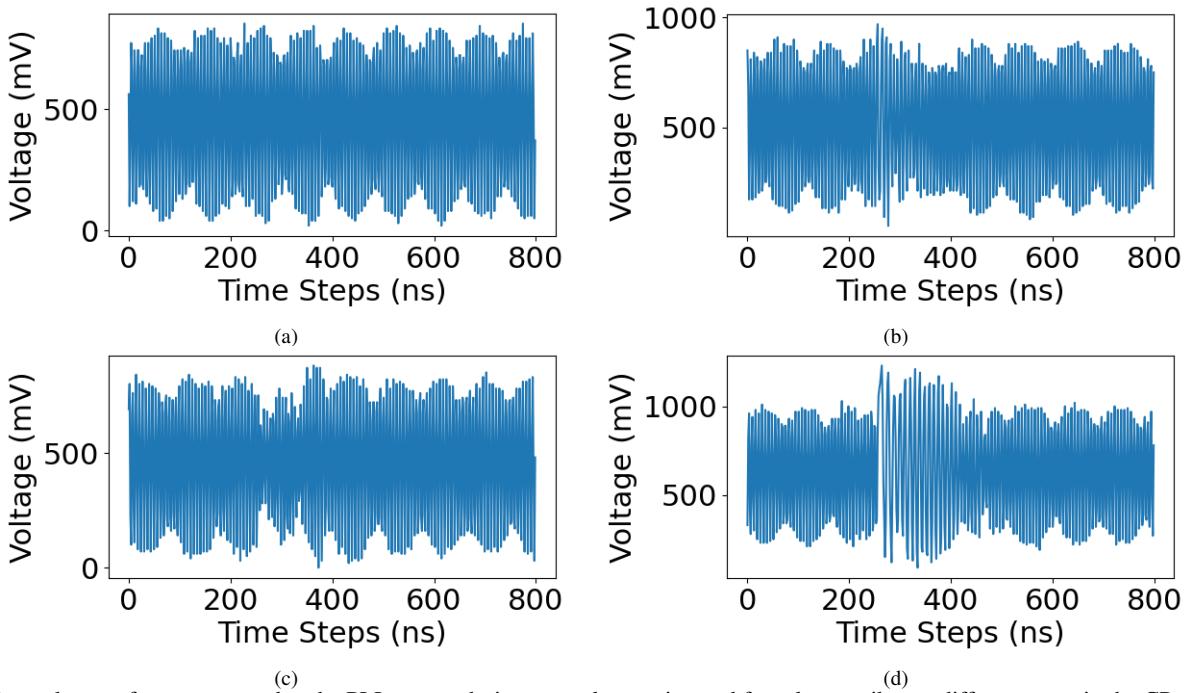

Fig. 3: Example waveforms measured at the PLL output during normal operation and from laser strikes to different areas in the CP sub-circuit of the PLL. A nominal signal is represented by (a) while (b), (c), and (d) contain varied transient responses.

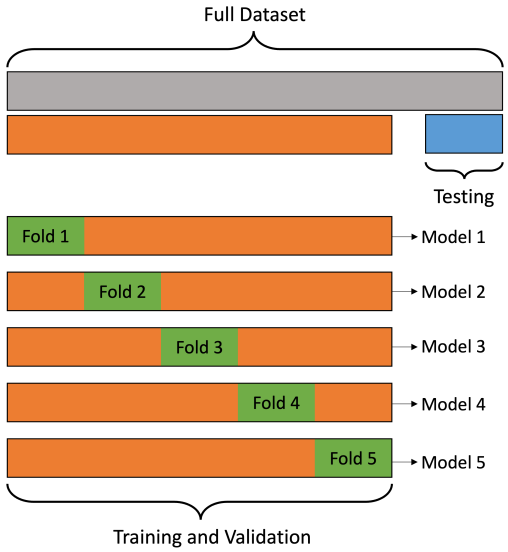

Fig. 4: An example of k-fold validation with 5 folds. After the full dataset is split into training and testing, the training portion is split 5 times, each with a different portion used for validation and represented in green. The final CNN model configuration is copied five times. Each copy is trained and validated on different portions of the data.

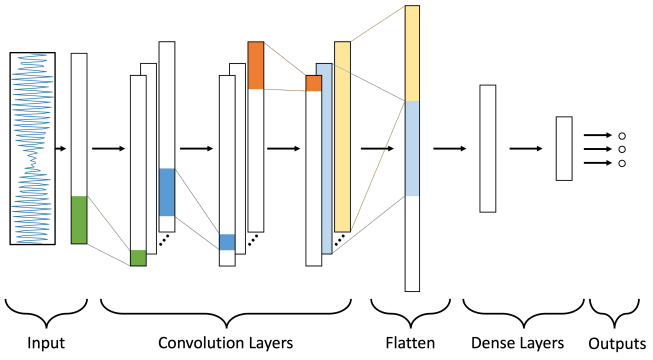

Fig. 5: 1D CNN architecture. The input is a 1D waveform, and the output is a vector representing the cluster prediction probabilities. Each 1D vector in a convolution layer represents the convolved output of a single filter multiplied by the input. The lines between layers indicate the filter weights which are the trainable parameters on the model, as well as the input and output shapes of the operation performed on the weights.

In CNNs, each convolutional layer is comprised of multiple elements (*i.e.*, neurons) where each element is associated with the result of element-wise multiplication between the kernel (*i.e.*, filter) and a portion of the input that matches size as the filter [23]. In addition, each convolutional layer is associated with one or more filters where each filter is comprised of multiple elements (*i.e.*, weights). Filter weights are considered the main trainable parameters of the CNN models. Results associated with all neurons within one convolutional layer are computed by performing the convolution of the filter and the data (*e.g.*, layer one strides across the raw data and layer two strides across the output of convolutional layer one) as shown in Fig. 5, illustrating the 1D CNN architecture. The input is the 1D time-sequence waveform under study, and the output is a vector representing the cluster prediction probabilities. Each 1D vector in a convolution layer represents the convolved output of a single filter multiplied by the input. The lines between layers indicate the filter weights, the trainable parameters on the model, and the input and output shapes of the operation performed on the weights.

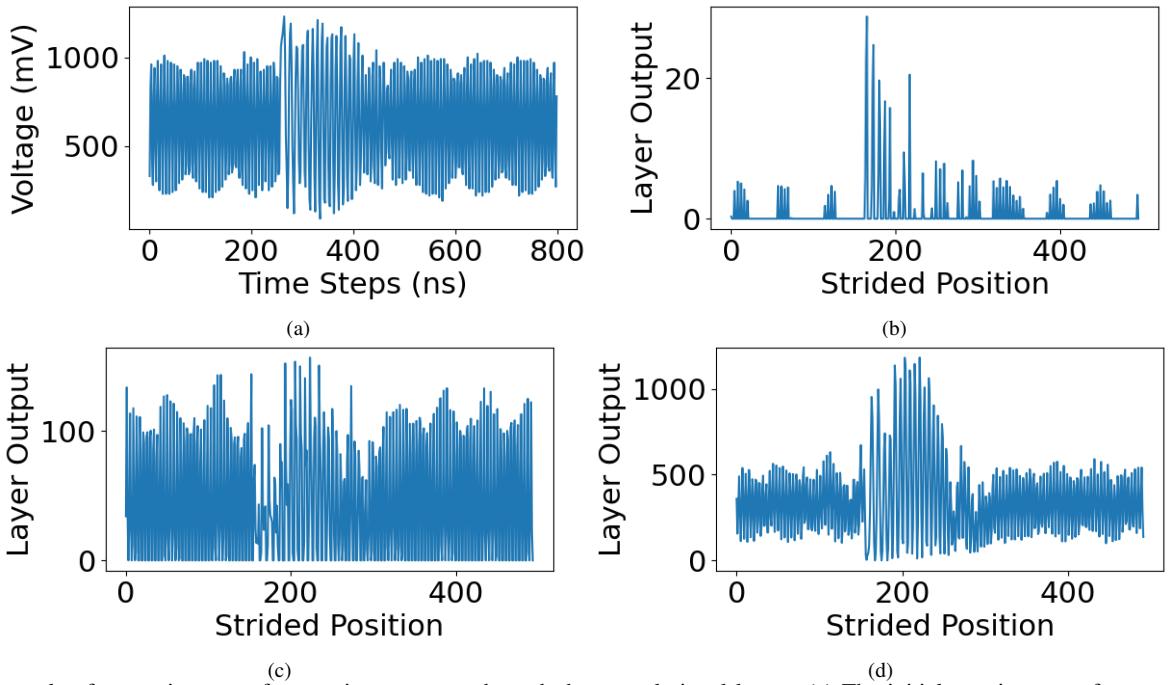

Fig. 6: Example of a transient waveform as it propagates through the convolutional layers. (a) The initial transient waveform at the output of the PLL where the y-axis contains the voltage and the x-axis represents 500 time samples at 1.6 ns steps. (b) Waveform transformed and multiplied with a single filter from the first convolutional layer. The y-axis represents the layer output, and the x-axis represents sample points at which the correlation is performed along the input data. Waveforms transformed from the (c) second and (d) third convolutional layers are also illustrated.

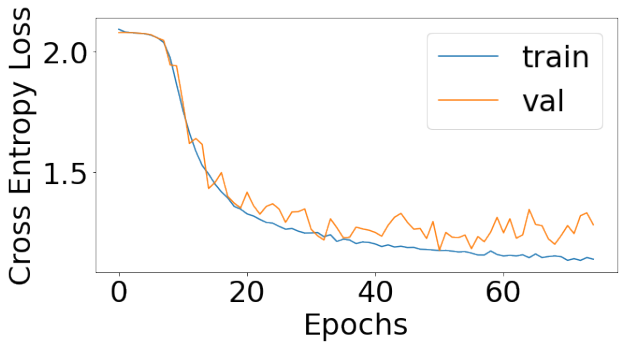

Fig. 7: Cross entropy loss of model over 75 epochs. As the model learns, the loss is minimized. Overfitting starts at approximately 20 epochs but does not become significant until approximately 50 epochs.

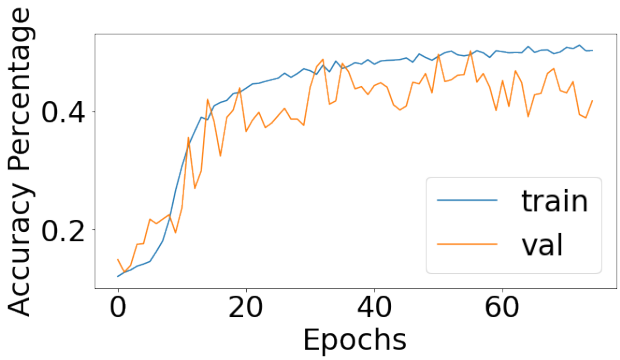

Fig. 8: Initial accuracy of the first model over 75 epochs. Like Fig. 7, the accuracy does not improve after approximately 20 epochs and stalls at approximately 40% accuracy.

Chaining multiple convolutional layers together allows for complex features to be learned. For example, Fig. 6 shows a waveform and the transformations that occur with a single filter from each of the three convolution layers. However, determining the optimal number of filters, kernel size, stride, and the number of layers is a nontrivial task and is discussed in Section IV.C.

B. Model Architecture and Training

The architecture of the initial model is comprised of three 1D convolutional layers, each with 16 filters, a kernel size of 3, and a stride of 1, as seen in Fig. 5. Each used the rectified linear unit (ReLU) activation function to prevent vanishing gradients [24]. The output of the final convolutional layer is flattened, allowing each filter to be fully connected to a dense layer containing 32 fully connected neurons with another dense layer containing 8 neurons, matching the number of unique sensitive subcomponents. The intermediate, fully connected layer before the final layer allows the model to take final position-invariant information from the flattened layer and correlate them together in a non-linear space. The intermediate layer dramatically improved the accuracy compared to a model directly connecting to the output layer. The softmax activation function is applied to the final output layer to calculate the prediction probabilities for each class. The model was trained using the Cross-Entropy Loss function with the Adam optimizer. Figs. 7 and 8 show the cross entropy loss and

accuracy versus the number of epochs, respectively. Overfitting starts at approximately 20 epochs. Batch normalization, L1 regularization, and Dropout are intermediate layers throughout the model to reduce overfitting.

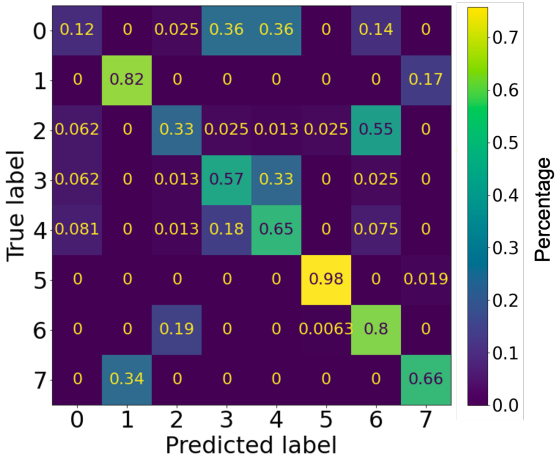

Fig. 9: Initial confusion matrix (true label vs. predicted label) of test data. The x and y cluster labels correspond to the labels from Fig. 2

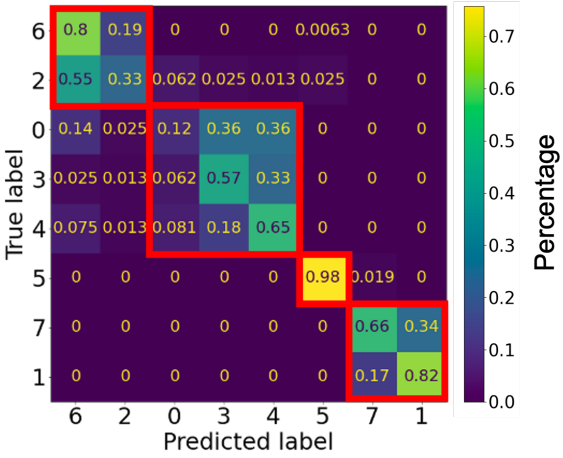

Fig. 10: The lowest-scored permutation of the confusion matrix, highlighting the confusion groups. Sorting occurred by taking the sum of the absolute difference of every cell to the surrounding cells in the confusion matrix and scoring/ranking each permutation of label orders. The lowest-scored permutation represents the most clustered representation of confusion and uncovers classes the model cannot discern.

IV. Classification of SETs Based on Origin of Perturbation

A. Initial Results

The initial model was able to correctly classify 45% of the test samples. This overall accuracy accounts for all correct and incorrect predictions by taking the weighted average of

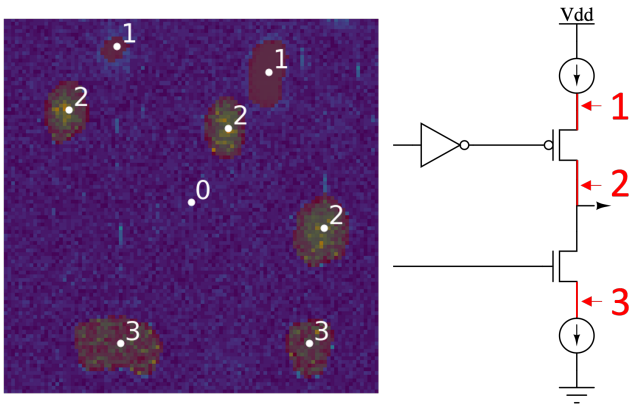

Fig. 11: Re-labeled spatial map of the CP. Each label corresponds to a group of electrically tied diffusions forming a single node.

the correct and incorrect predictions versus the number of samples for that class within the entire dataset. However, overall accuracy cannot provide insight into the specific classes the model performs well or poorly with. Therefore, a confusion matrix is used to display the predictive accuracy for each class, as shown in Fig. 9. The x-axis represents the model's predicted cluster label, while the y-axis represents the correct cluster label for the sample. The leading diagonal represents the cases when the predicted and correct labels are identical. The initial model accurately predicts the absence of SETs from the waveforms from strikes in the non-sensitive area (*i.e.*, background noise, or label 5 of Fig. 2) on 98% of the test dataset. However, this initial model cannot determine the correct classes for SETs with high accuracy for many different cluster labels.

$$b = \sum_{x=0}^{n}\sum_{y=0}^{n-1} |A_{x,\ y+1} - A_{x,y}| + \sum_{y=0}^{n}\sum_{x=0}^{n-1} |A_{x+1,\ y} - A_{x,y}| \quad (1)$$

Algorithm 1 Scoring Function

Require: A matrix A of size n x n

```
function ScoreConfusionMatrix(A)
    column_sum = sum(abs(diff(A, axis=0)))
    row_sum = sum(abs(diff(A, axis=1)))
    return column_sum + row_sum
end function
```

While some patterns within the confusion matrix are seen, the confusion matrix is sporadic and difficult to interpret. There are 8!, or 40,320 different ways to order the axes in an 8x8 confusion matrix. The optimal order would group the cells with similar upset characteristics, illuminating the model's inability to differentiate characteristics from the signals. To find the optimal order, the score (b) is computed for each permutation of the confusion matrix according to (1). A permutation's score is determined as the sum of the difference between every index position value and each index position's value to the right and below. The permutation with the lowest

score represents the optimally ordered confusion matrix. In (1), A is an n by n confusion matrix permutation, x is the column index, and y is the row index. The equation is also modeled as a high-level algorithm utilizing a matrix library (such as numpy) in Algorithm 1. The variable $column\_sum$ is the sum of the absolute difference between every index value along each column, whereas row_sum is computed along every row. The return value (b) is the total value for the matrix.

The lowest-scored permutation designates the most clustered representation of the matrix and uncovers groups of confusion the model cannot discern, as shown in Fig. 10. The algorithm effectively reveals the permutation with the lowest scoring difference between all values, which will be the most clustered. The top percentage of results will produce the same effect in different variations and positions many permutations.

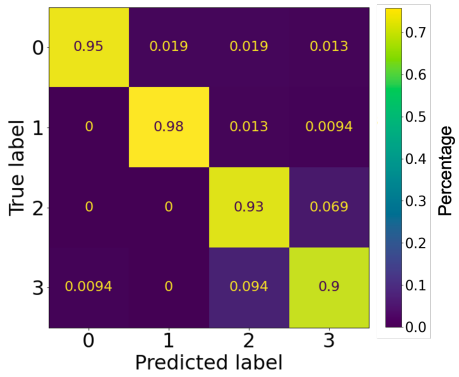

Fig. 12: Final confusion matrix (true label vs. predicted label) of test data with labeled groups 0-3, showing prediction accuracies of over 90% for each.

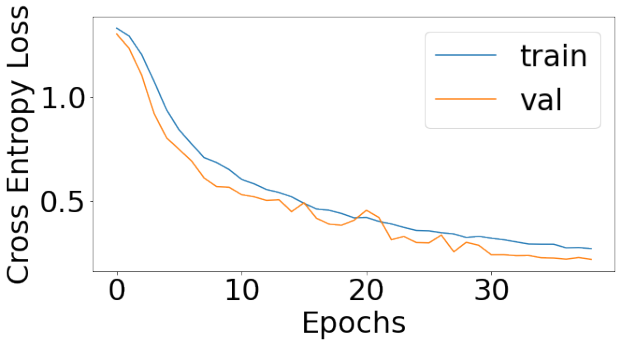

Fig. 13: Cross entropy loss of the reduced model after 50 epochs.

As seen in Fig. 10, clusters labeled 7 and 1 were approximately 70% accurate but often misidentified between the two. A similar pattern was found in clusters 0, 3, and 4, and again between 6 and 2, with significantly less accuracy. The model's inability to distinguish these clusters is due to the groups being electrically tied subcomponents within the CP.

B. Re-labeled Results

These data were re-labeled with cluster labels 0 through 3, representing the four main confusion groups highlighted

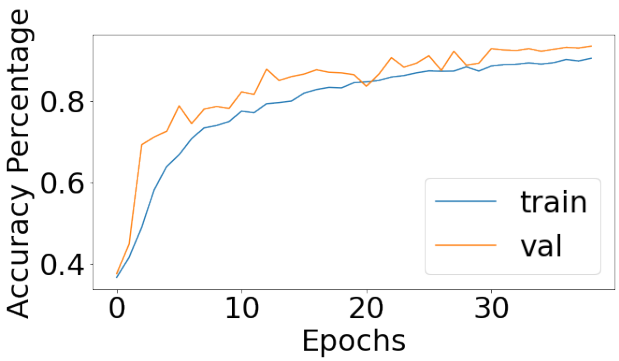

Fig. 14: Accuracy of the reduced model after 50 epochs.

in Fig. 10. In other words, the algorithm identified the four linearly independent variables (clusters) within the model. This dimensionality reduction was accomplished purely through the permutation sorting algorithm previously described, without consideration of the electrical connectivity. However, further examination of data illustrates that the three sensitive regions of interest (labels 1, 2, and 3) correspond to electrically tied components within the simplified schematic of the CP subcircuit, as seen in Fig. 11. Therefore, the CNN model was retrained with these re-labeled groups of confusion (*i.e.*, linearly dependent variables) representing electrically-tied components. This re-labeling reduced the total number of clusters to 4, resulting in prediction accuracies of 90% or greater, as shown in the final confusion matrix in Fig. 12. Figs. 13 and 14 show the cross entropy loss and accuracy of the reduced model over 50 epochs.

C. Hyperparameter Tuning

A final step to ensure the highest performing accuracy on a model is to perform hyperparameter tuning. Parameters are defined as any component weights and biases that are learned during training. Hyperparameters, however, are related to the model's structure. Hyperparameters include the number and size of the kernels in each convolution layer, the size of the stride, and the size of the kernels in the pooling layer. Hyperparameters play a significant impact on the speed, ability, and amount that a model can learn.

Some examples of hyperparameters for the model used in this work are shown in Fig. 15, and include the number of convolution layers, number of filters, size of each filter, and number of neurons in the final layer. Many methods exist for finding the optimal set of hyperparameter configurations [25]–[27]. In this case, it was sufficient to manually adjust the hyperparameters to create the initial model composition described in Section III-B. However, a standard grid search was used to obtain a final model with the optimal set. Each hyperparameter was given a range or subset of values to be permuted. Nine hundred fifty models were trained over 500 epochs, and the state of the models were saved after every 10 epochs. The optimal model was determined by the one with lowest loss and highest accuracy at the point before overfitting began to occur. Each model was trained and evaluated on the same training and testing dataset.

Fig. 15 shows a simplified correlation matrix of the major hyperparameters with respect to model loss, accuracy, validation loss, and validation accuracy as generated with Pearsons correlation coefficient [28] using the pandas correlation function [29]. The p values were also computed to be less than 1×10^{-5} in each case, indicating a strong statistical significance for each of the correlations. In Fig. 15, the training and testing loss and accuracy are compared across the major hyperparameters. The number of convolution layers varied from 1 to 4 layers. The number of filters was evaluated between 3, 6, 9, 16, 32, and 64. The filter size varied between 3, 5, 7, and 9. The number of final neurons varied between 8, 16, 32, 64, and 128. Every change in a single variable resulted in a separate model and was evaluated with every other change, resulting in the 950 model variations. The results from Fig. 15 show that the number of convolution layers has weak to moderate positive correlation, but the most significant impact on increasing accuracy and validation accuracy as well as the largest impact on decreasing loss and validation loss with weak to moderate negative correlation. Filter size has no correlation with validation loss, accuracy, and validation accuracy with negligible correlation. On the other hand, the filter and the final number of neurons are almost equal in correlation but only in loss and accuracy. This result indicates that models with more numbers of filters or final layer neurons are more prone to overfitting, as they are not as correlated to the validation accuracy.

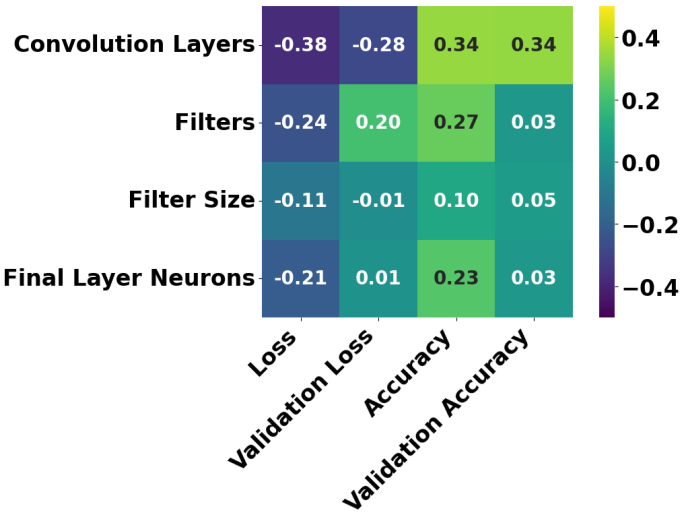

Fig. 15: Correlation matrix displaying positive and negative correlations between hyperparameters. Correlations were calculated using Pearsons correlation coefficient. A value of -1 and 1 represent a perfect negative and positive correlation, respectively.

V. Conclusion

This work documents the first use of CNN for determining features within SET waveforms that allow the identification of ion strike locations within a circuit. The model was over 90% accurate at classifying SETs based on the circuit node of origin. Not only is the ML model able to achieve a high degree of accuracy in locating an SETs origin, but a process of sorting the resulting confusion matrices is provided that uncovers information about the DUT configuration. This approach could lead to the possibility of detecting, mitigating, and reacting to single-event effects in real-time. It also allows for more robust measurement equipment for real-time data analysis during experimentation. Future research could involve the extraction of the physical characteristics of the circuit or the uncovering of more information about the model's decision-making process.

References

[1] B. Patel, M. Joplin, R. C. Boggs, D. R. Reising, M. W. McCurdy, L. W. Massengill, and T. D. Loveless, "Ionizing radiation effects spectroscopy for analysis of total-ionizing dose degradation in rf circuits," *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 61–68, Jan 2019.

[2] M. G. Trindade, A. Coelho, C. Valadares, R. A. C. Viera, S. Rey, B. Cheymol, M. Baylac, R. Velazco, and R. P. Bastos, "Assessment of a hardware-implemented machine learning technique under neutron irradiation," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1441–1448, July 2019.

[3] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, July 2019, pp. 72–78.

[4] T. D. Loveless, B. Patel, D. R. Reising, R. Roca, M. Allen, L. W. Massengill, and D. McMorrow, "Ionizing radiation effects spectroscopy for analysis of single-event transients," *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 99–107, Jan 2020.

[5] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Composing graph theory and deep neural networks to evaluate seu type soft error effects," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, June 2020, pp. 530–535.

[6] M. G. Trindade, R. P. Bastos, R. Garibotti, L. Ost, M. Letiche, and J. Beaucour, "Assessment of machine learning algorithms for near-sensor computing under radiation soft errors," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nov 2020, pp. 494–498.

[7] T. D. Loveless, D. R. Reising, J. C. Cancelleri, L. W. Massengill, and D. McMorrow, "Analysis of single-event transients (sets) using machine learning (ml) and ionizing radiation effects spectroscopy (ires)," *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1600–1606, Aug 2021.

[8] R. Song, J. Shi, J. Shao, and X. Zhang, "Machine learning based set propagation prediction for large scale integrated circuits," in *2021 IEEE 14th International Conference on ASIC (ASICON)*, Oct 2021, pp. 475–479.

[9] M. Hashimoto, W. Liao, and S. Hirokawa, "Soft error rate estimation with tcad and machine learning," in *2017 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Sep. 2017, pp. 129–132.

[10] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning to tackle the challenges of transient and soft errors in complex circuits," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, July 2019, pp. 7–14.

[11] T. Lange *et al.*, "On the estimation of complex circuits functional failure rate by machine learning techniques," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, June 2019, pp. 35–41.

[12] C. Xu, Y. Liu, X. Liao, J. Cheng, and Y. Yang, "Machine learning regression-based single-event transient modeling method for circuit-level simulation," *IEEE Transactions on Electron Devices*, vol. 68, no. 11, pp. 5758–5764, Nov 2021.

[13] D. L. Hansen, D. Czajkowski, and B. Vermeire, "Using machine learning to determine proton cross-sections from heavy-ion data," *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 264–272, March 2022.

[14] L. C. Adams, J. Howard, E. J. Barth, R. D. Schrimpf, R. A. Reed, R. A. Peters, and A. F. Witulski, "Machine learning techniques for mitigating sensor ionizing dose failures in robotic systems," in *2018 18th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, Sep. 2018, pp. 48–53.

[15] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning clustering techniques for selective mitigation of critical design features," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, July 2020, pp. 31–38.

[16] A. Ildefonso, J. P. Kimball, A. Khachatrian, Y. Mensah, J. W. Teng, G. N. Tzintzarov, S. G. Rao, A. Moradinia, J. D. Cressler, and D. McMorrow, "Using machine learning to mitigate single-event upsets in rf circuits and systems," *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 381–389, March 2022.

[17] S. Jian, J. Jiang, K. Lu, and Y. Zhang, "Seu-tolerant restricted boltzmann machine learning on dsp-based fault detection," in *2014 12th International Conference on Signal Processing (ICSP)*, Oct 2014, pp. 1503–1506.

[18] S. Roffe and A. D. George, "Evaluation of algorithm-based fault tolerance for machine learning and computer vision under neutron radiation," in *2020 IEEE Aerospace Conference*, March 2020, pp. 4334–4342.

[19] T. D. Loveless, L. W. Massengill, W. T. Holman, B. L. Bhuva, D. McMorrow, and J. H. Warner, "A generalized linear model for single event transient propagation in phase-locked loops," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 5, pp. 2933–2947, Oct 2010.

[20] T. D. Loveless, L. W. Massengill, B. L. Bhuva, W. T. Holman, A. F. Witulski, and Y. Boulghassoul, "A hardened-by-design technique for rf digital phase-locked loops," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3432–3438, Dec 2006.

[21] T. D. Loveless, L. W. Massengill, W. T. Holman, and B. L. Bhuva, "Modeling and mitigating single-event transients in voltage-controlled oscillators," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 6, pp. 2561–2567, Dec 2007.

[22] Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[23] Patterson, J. and A. Gibson, *Deep Learning A Practitioners Approach*. O'Reilly Media, 2017.

[24] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[25] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Learning hyperparameter optimization initializations," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2015, pp. 339–349.

[26] W. Alawad, M. Zohdy, and D. Debnath, "Tuning hyperparameters of decision tree classifiers using computationally efficient schemes," in *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, Sep. 2018, pp. 168–169.

[27] T. T. Joy, S. Rana, S. Gupta, and S. Venkatesh, "Hyperparameter tuning for big data using bayesian optimisation," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 2574–2579.

[28] D. Freedman, R. Pisani, and R. Purves, "Statistics (international student edition)," *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[29] J. Reback *et al.*, "pandas-dev/pandas: Pandas 1.4.4," Aug. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7037953