Machine learning for analyzing and characterizing InAsSb-based nBn photodetectors

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 025006 https://doi.org/10.1088/2632-2153/abcf89

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
3 August 2020

REVISED
3 November 2020

ACCEPTED FOR PUBLICATION
1 December 2020

PUBLISHED
29 December 2020

PAPER

Machine learning for analyzing and characterizing InAsSb-based nBn photodetectors

Andreu Glasmann[1], **Alexandros Kyrtsos**[1] and **Enrico Bellotti**[1,2]

[1] Department of Electrical and Computer Engineering, Boston University, Boston MA 02215, United States of America
[2] Division of Materials Science and Engineering, Boston University, Boston MA 02215, United States of America

E-mail: andreug@bu.edu

Keywords: capacitance, neural networks, infrared photodetectors, convolutional neural networks, barrier detectors, machine learning

Abstract

This paper discusses two cases of applying artificial neural networks to the capacitance–voltage characteristics of InAsSb-based barrier infrared detectors. In the first case, we discuss a methodology for training a fully-connected feedforward network to predict the capacitance of the device as a function of the absorber, barrier, and contact doping densities, the barrier thickness, and the applied voltage. We verify the model's performance with physics-based justification of trends observed in single parameter sweeps, partial dependence plots, and two examples of gradient-based sensitivity analysis. The second case focuses on the development of a convolutional neural network that addresses the inverse problem, where a capacitance–voltage profile is used to predict the architectural properties of the device. The advantage of this approach is a more comprehensive characterization of a device by capacitance–voltage profiling than may be possible with other techniques. Finally, both approaches are material and device agnostic, and can be applied to other semiconductor device characteristics.

1. Introduction

Recent advances in computing capabilities have led to an unprecedented rise in data generation and availability. As such, there has been a need for rapid development of tools to aid researchers in identifying trends hidden deep within high-dimensional datasets. Machine learning [1–3], a subset of artificial intelligence concerned with studying techniques for building data-driven mathematical models, offers a set of tools that are well-suited to address this need. In addition to computing hardware becoming more powerful, improvements in machine learning algorithms and the development of user-friendly machine learning frameworks has made these types of tools more accessible than ever.

Machine learning has gained renewed interest recently due to new neural network architectures achieving state of the art performance in numerous disciplines for applications in image and speech recognition, language processing, and data analysis. For instance, machine learning techniques have started to be routinely used, or researched for use, in many diverse scientific and engineering fields, including bioinformatics [4], materials science [5–7], and radiology [8, 9]. In this paper, we discuss machine learning in the scope of semiconductor device development.

The semiconductor industry has been using machine learning to address topics in modeling, manufacturing, and recently, design. For example, artificial neural networks (ANNs) have shown to be useful as a method of creating compact models for circuit simulators [10–13]. ANNs can reproduce device characteristics over wider ranges of operating conditions where analytic equations may no longer be valid. From another perspective, ANNs could also be used to create a statistically representative compact model for a batch of fabricated devices that have been characterized over temperature, voltage, and frequency, creating a more realistic representation of the product in the circuit simulation. In either case, once trained, ANNs are faster to evaluate than a numerical approach, and can offer higher accuracy and superior generalization than a look-up table.

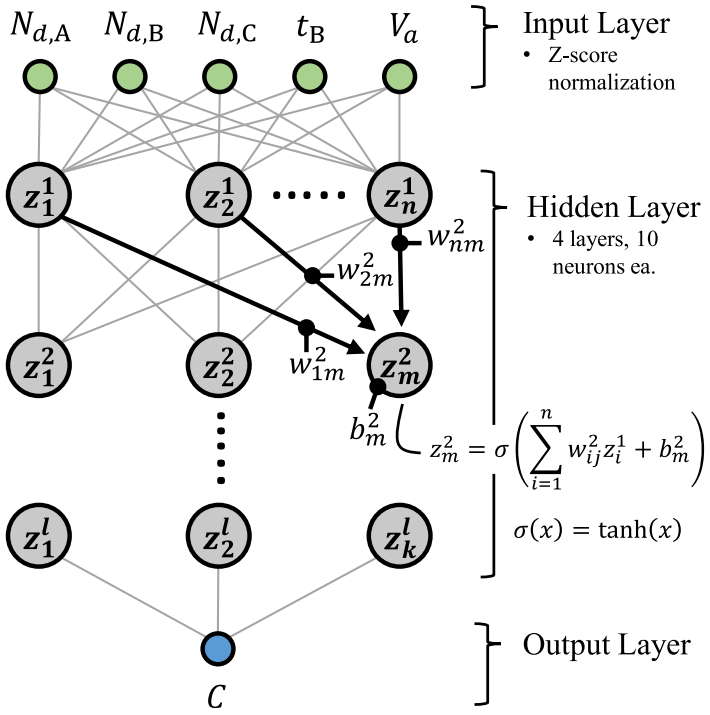

Figure 1. Diagram illustrating a neural network used in this work that predicts a device's capacitance based on architectural and operational variables.

Another compelling example in the semiconductor industry is the application of machine learning to manufacturing. By maintaining logs of each fabrication cycle, facilities can generate vast amounts of data each year. This data can be leveraged using machine learning techniques to improve models used for fault detection, predictive maintenance, correlating processing parameters to yield, and develop advanced metrology techniques [14–18]. A notable example is the use of machine learning to aid in wafer fault detection [19, 20]. By monitoring the progress of a wafer through a series of processing steps, a trained convolutional neural network can prevent accruing additional cost by using pattern recognition to find defects on the wafer, identifying the faulty process, and preventing the wafer from continuing through the fabrication line. The neural network can have higher detection accuracy than a human technician, and also offer insight into ways to improve the fabrication line by analyzing the features that led the network to its prediction.

A widely encountered challenge in device development is understanding how each design parameter affects performance. It quickly becomes difficult to understand parameter-performance correlations as the number of free parameters increases. There has been recent progress in using approaches based on machine learning to address this problem [21]. However, these techniques have yet to be applied to infrared photodetectors, where the device architecture and material are extensively engineered to meet performance requirements in demanding applications.

As is the case in many areas of engineering, research regarding infrared sensor design and fabrication is predominantly driven by finding ways to reduce cost while simultaneously improving performance and adding new capabilities. The barrier detector design [22–24] has achieved success for midwave-infrared sensing in this aspect by allowing the detector to reach background-limited performance at higher operating temperatures, reducing the cooling requirements for the imaging system. Realization of these types of innovative designs requires sufficient theoretical knowledge of the device operation, often through physics-based analytical or numerical modeling. There has been much effort in modeling barrier infrared detectors in recent years [25, 26], and as a result, the methodology and materials models are now relatively mature. As modeling capabilities and computing hardware continue to improve, we are able to more efficiently study complex device designs. Machine learning offers tools to both ameliorate analysis of the data to make definitive conclusions, and construct physics-based surrogate models for complex device phenomena. It is the aim of this work to demonstrate a methodology for applying machine learning to simulation results to create models for studying, analyzing, and optimizing device characteristics. To this end, we look at two approaches of applying neural networks to the capacitance–voltage (C–V) characteristics

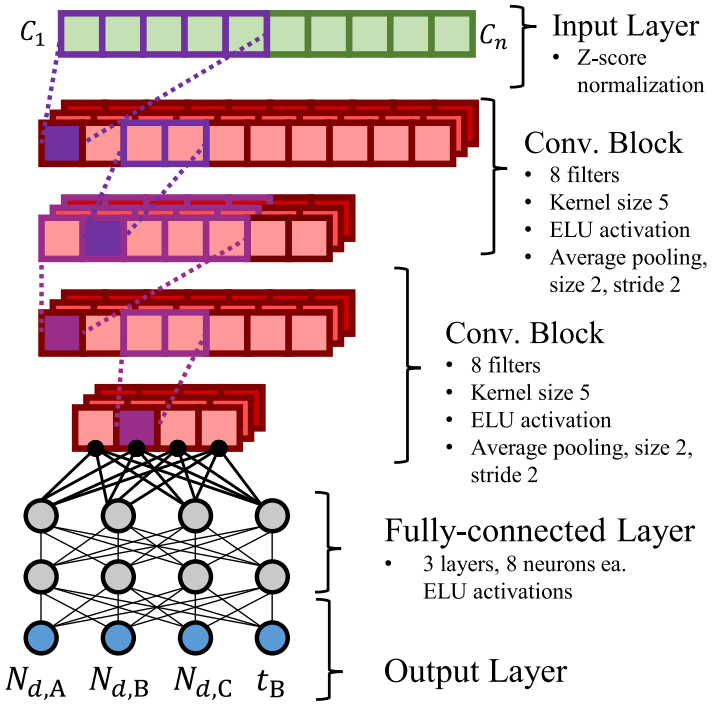

Figure 2. The proposed convolutional neural network design for predicting device properties from capacitance–voltage characteristics.

of InAsSb-based nBn photodetectors. To our knowledge, this is the first report of applying neural networks to this type of device and its C–V characteristics.

While capacitance is not a common figure of merit when discussing infrared photodetector performance, it is a common technique for characterizing semiconductor materials [27]. Moreover, it has been shown [28–30] that the C–V characteristics for barrier-style devices can elucidate more about the underlying materials. The C–V profile can be used to extract information about the doping density in all three critical device layers, the absorber, barrier, and contact, as well as the thickness of the barrier. To this end, this paper covers two approaches for applying neural networks to C–V characteristics.

First, we show that an ANN can be trained to accurately predict the capacitance based on a subset of the device's architectural parameters. In this case, we include the doping densities of the absorber, barrier, and contact layers, the thickness of the barrier layer, and the applied voltage as input features for the model. Once trained, the model is used as an analysis tool to understand the role of each feature in shaping the C–V characteristics.

Second, inspired by recent developments in image recognition, we address the inverse problem by using a convolutional neural network to create an enhanced characterization tool for C–V analysis. The convolutional network is used in place of conventional analytic techniques, and may be applied to other devices and cases where analytical models are not valid. In this paper, we train a convolutional neural network to predict the doping densities of the absorber, barrier, and contact layers, and the thickness of the barrier layer and show excellent generalization over a wide range of values for each parameter.

This manuscript is organized as follows: section 2 introduces the neural networks studied in this work, and discusses the C–V characteristics of barrier infrared detectors; section 3 presents the methodology used for data acquisition and training the networks; section 4 presents the results; section 5 concludes the article.

2. Background

2.1. Neural networks: multilayer perceptron

Machine learning provides many algorithms for creating predictive models from data. One approach, and the one we adopt in this work, are neural networks. We briefly discuss a few of the basic concepts of neural networks in this section.

The first type of network we present in this work is based on the multilayer perceptron model where information is propagated through a set of fully-connected layers. This type of feedforward network calculates a prediction vector, $\hat{\boldsymbol{y}} = \{\hat{y}_1, \ldots, \hat{y}_k\}$, by performing a series of simple calculations on an input

vector, $\mathbf{x} = \{x_1, \ldots, x_p\}$. The vectors $\mathbf{x}$ and $\hat{\boldsymbol{y}}$ represent the input and output layers respectively, and elements of $\mathbf{x}$ are referred to as features or predictors. The calculations that lead the input to the output take place in the hidden layer. The hidden layer contains layers of fully-connected elements, or neurons. The number of layers, and neurons in each layer, need to be optimized for the given problem. Consider two sequential layers H_1 and H_2 with n and m neurons respectively. The value calculated by the jth neuron in H_2 is given by

$$z_j^2 = \sigma\left(\sum_{i=1}^{n} w_{ij}^2 z_j^1 + b_j^2\right),$$

where w_{ij}^2, b_j^2, and z_j^1 denote the weight associated with the connection between the ith neuron in H_1 and the jth neuron in H_2, the jth neuron's bias, and the activated value from the ith neuron. The function, σ, is the activation function for this neuron, and is a way to introduce non-linearity into the model. There are many choices of activation functions, and more are researched to improve computational efficiency, stability, convergence rate, and accuracy. In this work we will use hyperbolic tangent,

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

and the exponential linear unit, or ELU, given by [31]

$$\sigma(z) = \begin{cases} z & z > 0 \\ e^z - 1 & z \leq 0. \end{cases}$$

As we increase the complexity of the model by adding more neurons and hidden layers, the output will become an increasingly complex function of nested function evaluations.

The sets of weights and biases for each hidden layer are represented as matrices of free variables that are optimized by an optimization algorithm during the learning process. During training, the network is exposed to a set of N samples, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}^T$ of known outputs, $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}^T$, referred to as the training data. An important consideration when designing the network is the complexity of the model. If there are too many weights and biases compared to N, the model may overfit the data, and will not generalize well to unseen data. The weights and biases in the network are optimized to minimize a cost function J, expressed as

$$J(\mathbf{X};\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\mathbf{y}_i, \hat{\mathbf{y}}_i(\mathbf{X};\theta)\right). \tag{1}$$

$\mathcal{L}$ is the loss function that is used to calculate the error in each prediction for every sample. θ is used to denote the set of all hyperparameters used in the model, for instance, the weights, biases, learning rates, and data processing. For regression, a common choice, and what we use here, is the mean squared error

$$\mathcal{L}(\mathbf{y}, \hat{\boldsymbol{y}}) = \frac{1}{k}\sum_{j=1}^{k} \left(y_j - \hat{y}_j\right)^2.$$

Similarly to σ, there are several choices for J and $\mathcal{L}$ for other types of problems.

The recent reemergence of neural networks is due in part to improvements in optimization algorithms for updating the weights and biases. For large-scale applications, the numbers of weights and biases can be such that the task of optimizing hundreds of thousands of free parameters is non-trivial, either suffering from issues with stability or being simply computationally prohibitive. One of the most popular, and a particularly intuitive approach, is to use the method of steepest descent. The gradients of equation (1) with respect to the weights and biases are calculated by repeatedly applying the chain rule—an approach called backpropagation. These gradients are then used to update each weight and bias in the network in such a way that the cost function is, ideally, reduced with every iteration.

2.2. Convolutional neural networks

Convolutional neural networks, or convnets, are frequently used for pattern recognition due to their ability to identify relevant features in an image by learning the correspondence between neighboring pixels [32]. The key to the success of convnets is their use of learned sets of weights and biases, called filters, used to perform convolutions over clusters of pixels [1]. The outputs of these convolutions are referred to as feature maps. These maps can highlight pertinent information in the image; for example, feature maps have shown to emphasize edges of objects with various orientations [3, 33, 34].

Basic convolutional networks are based on the following design. The input, considering for example a greyscale image, is simply a structured array of pixel intensity values. This array is passed to a series of convolutional blocks. These blocks are composed of convolutional and pooling layers. The purpose of the convolutional blocks is to extract the salient information from the image, which is then used by a final fully-connected network in its prediction.

Consider the following example of a convolutional layer in the network. Let I_{ij} denote the intensity of the pixel located in row i and column j. Consider filter f, and assume it has m rows and n columns. Then, the resulting value for the feature map at this position, z^f_{ij}, when the top-left of filter is aligned with pixel ij will be

$$z^f_{ij} = \sigma\left[\sum_{q=0}^{m}\sum_{r=0}^{n}\left(w^f_{qr}I_{i+q,j+m}\right) + b^f\right].$$

The final feature map contains the outputs, z^f for every position on the image that the filter is applied. Each filter contains $n \times m$ weights and a single bias. Notice that now, unlike the multilayer perceptron model in the previous subsection, we have decoupled the number of free parameters in the model from the dimensionality of the preceding layer by sharing the weights and bias in each filter with every input. The dimensions of the feature map are controlled by how the filter is stepped across the image and the dimensions of the filter. The number of filters, the filter dimensions, and the way the filter is stepped across the preceding layer are treated as hyperparameters that are typically determined empirically.

While the number of free parameters associated with the convolutional layers is controlled by the filters, the feature maps may still have high dimensionality. Recall that the output of the convolutional block will be given to a fully-connected network. Since the number of free parameters associated with each set of fully-connected layers is directly proportional to the number of inputs, it can be useful to further reduce the dimension of each layer in the convolutional block. One common approach is to use pooling. Pooling involves calculating an aggregate assumption about a cluster of pixels, usually by finding the average, minimum, or maximum value. The size and stride of the pooling operation are again hyperparameters that must be chosen, but are both commonly fixed at two in order to reduce the dimensionality in half.

While we limited this brief discussion of convnets to images, the pattern recognition ability can be applied to any type of structured data where information can be found in localized clusters of points. For this reason, convnets are continued to be researched for applications in signal processing. A few compelling examples include using convnets to formulate diagnoses based off of medical signals, such as echocardiograms [35, 36] and electroencephalograms [37]. Following similar logic, a semiconductor device's electrical characteristics, such as current–voltage or C–V, are representative of the underlying device structure and material quality. It is the goal of this paper to introduce a methodology based on convolutional neural networks to predict device-related parameters from C–V characteristics.

2.3. Capacitance–voltage characteristics

Capacitance–voltage profiling is a common approach to measure the doping density of a semiconducting layer within a device [27, 38]. The capacitance of a barrier detector can be approximated as a series capacitance from three regions of the device, or [28, 29, 39]

$$\frac{1}{C} = \frac{1}{C_{j,\mathrm{A}}} + \frac{1}{C_{\mathrm{B}}} + \frac{1}{C_{j,\mathrm{C}}}, \tag{2}$$

where $C_{j,\mathrm{A}}$ and $C_{j,\mathrm{C}}$ are the junction capacitances due to the absorber-barrier and contact-barrier interfaces, and C_{B} is a parallel plate capacitance when the barrier layer is fully depleted given by

$$C_{\mathrm{B}} = \frac{\epsilon_0\epsilon_{\mathrm{r}}}{t_{\mathrm{B}}}, \tag{3}$$

where t_{B}, ϵ_{r}, and ε_0 are the thickness of the barrier layer, relative dielectric constant, and vacuum permittivity. Usually, assumptions are made regarding equation (2) to characterize the barrier detector. For the structure considered in this work, an n-type absorber, N-type barrier, and nth type contact, when a reverse bias is applied, equation (2) can be simplified to a series capacitance of $C_{j,\mathrm{A}}$ and $C_{j,\mathrm{B}}$, since $C_{j,\mathrm{C}}$ will be a relatively large accumulation capacitance. Under this assumption, it is possible to extract the absorber doping density through the conventional technique from metal-oxide-semiconductor theory by analyzing $1/C^2$ [27, 39].

A more complete solution for equation (2) can be obtained by considering the three semiconducting layers and applying Poisson's equation to the two interfaces with appropriate boundary conditions [29, 40]. This approach, however, can be challenging to implement due to requiring a numerical approach to solve the resulting transcendental equation. An alternative is to use a drift-diffusion model to solve a set of coupled

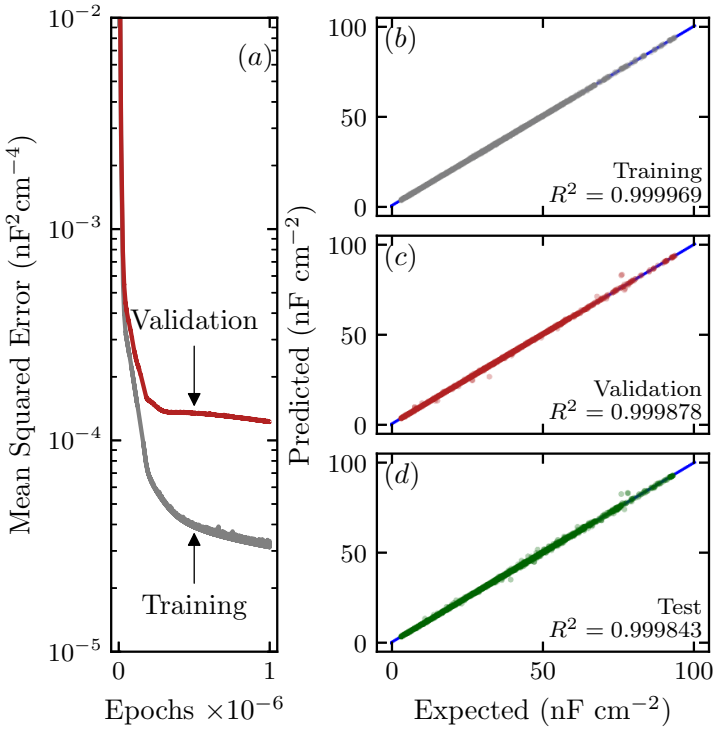

Figure 3. (a) Learning curves showing the mean squared error of the capacitance predicted by the neural network for the training and validation sets during training. Predicted versus expected capacitance for the (b) training and (c) validation sets respectively. The blue lines have unity slope and zero intercept for reference. (d) Predicted versus expected capacitance for a test dataset that contains values of N_d,A, N_d,B, N_d,C, and t_B that are not present in the training or validation sets.

Table 1. A summary of the device architecture used when performing drift-diffusion simulations to calculate capacitance–voltage characteristics.

Layer	Material	Composition, x	$E_g(T = 150\text{K})$ (eV)	Thickness (μm)	Doping (cm^{-3})
Substrate	GaSb	—	0.778	0.5	—
Absorber	$InAs_{1-x}Sb_x$	0.09	0.321	3	n-type 10^{14}–10^{17}
Barrier	$AlAs_{1-x}Sb_x$	0.9	1.696	0.1–0.3	N-type 10^{15}–10^{17}
Contact	$InAs_{1-x}Sb_x$	0.09	0.321	0.1	n-type 10^{15}–10^{17}

equations. Drift-diffusion alleviates many of the constraints imposed by the analytical approach, but the added computational complexity and need for advanced materials models can also be a hinderance. In this paper, we apply machine learning to this problem in two ways. First, by using drift-diffusion simulation results, we train a neural network to create a surrogate model for the C–V relationship that contains the added physical accuracy of drift-diffusion, while being quickly evaluated like analytical models, facilitating rapid analysis of the characteristics. This approach creates a model that can be easily used to study these types of complex problems, and can deepen our comprehension of performance-parameter correlations in multidimensional spaces. Second, we suggest a solution to the inverse case where a more complete characterization of the device is possible by training a convolutional neural network to analyze its C–V profile.

3. Methods

In a previous work, we used Synopsys Sentaurus TCAD to simulate the C–V characteristics of InAsSb-based nBn photodetectors [29, 41]. In the same work, we showed through a combination of analytical and numerical modeling that we are able to reproduce measured C–V data of similar nBn devices. The detector structure used to perform the simulations is summarized in table 1. Simulations were performed assuming a temperature of 150 K, and with a frequency of 1 MHz. More details regarding the simulation methodology, materials parameters, and analysis can be found in other works [29, 42].

For this study we created a database of drift-diffusion computed capacitances for varied values of the absorber doping density, N_d,A, barrier doping density, N_d,B, contact doping density, N_d,C, barrier thickness, t_B, and the applied voltage, V_a. One prominent issue in creating data-driven models is how the the data is sampled. The easiest approach is using a grid search over the parameters, where every permutation of set of discrete values for each feature is sampled. However, while simple to implement, this causes the model to only observe a small number of values for each feature. Furthermore, if we consider using N discrete values for n features, the number of samples required is N^n, which quickly becomes infeasible for large values of N to ensure adequate mapping of the parameter space. To address this issue, we opted for an approach using a quasi-random sampling approach based on Halton sequences [43–45]. Quasi-random techniques offer better coverage of the parameter space than gridded sampling, while preventing the possibility of clustering by random sampling. In this instance, if we take N samples, there will be N different values for each parameter that are more evenly distributed through the space. In this study, we used Halton sequences to discretize N_d,A, N_d,B, N_d,C, and t_B, while the values of V_a were fixed at 101 evenly spaced points during each simulation.

We used TensorFlow [46] as the framework for initializing, training, and evaluating the neural networks. The first neural network considered in this work is a fully-connected feedforward network. The input layer is comprised of five neurons for N_d,A, N_d,B, N_d,C, t_B, and V_a. The hidden layer contains four layers with ten neurons each. We used hyperbolic tangent as the activation function for each neuron. The output layer is a single neuron with a linear activation representing the predicted value of the capacitance, C. A summary of the network architecture is shown in figure 1. Weights were initialized using Glorot initialization [47]. Biases were initialized as zeros. The weights and biases were optimized using the Adam optimization algorithm [48]. The recommended values for the hyperparameters β_1, β_2, and ε of 0.9, 0.999, and 10^{-7} were verified to be reasonable by monitoring the validation loss during a small grid search of different values for each. Prior to training, we applied a Z-score normalization to the training inputs and labels to transform the sets to have zero mean and unity standard deviation, and used the base-10 logarithmic values of N_d,A, N_d,B, and N_d,C. For stability the initial learning rate was set to 10^{-4}. The networks were trained using 128 quasi-random samples of N_d,A within 10^{14}–10^{17} cm^{-3}, N_d,B within 10^{15}–10^{17} cm^{-3}, N_d,C within 10^{15}–10^{17} cm^{-3}, and t_B within 100–300 nm. To improve the quality of the model, we also include the 16 limiting cases—for example N_d,A $= 10^{14}$ cm^{-3}, N_d,B $= 10^{15}$ cm^{-3}, N_d,C $= 10^{17}$ cm^{-3}, and $t_B = 100$ nm—that make up the corners of the four-dimensional hypercube, giving a total number of simulated cases of 144 with 14 544 individual capactiances. The training and validation sets were constructed by splitting each simulated C–V curve in half, taking every other voltage as training and every other adjacent voltage as validation.

The second considered case uses a convolutional neural network; a summary of the network architecture is shown in figure 2. In this case, a C–V profile is used as the input layer, and is represented as a one-dimensional array with 101 capacitances. The inputs were normalized prior to training using Z-score normalization. The convolutional block is made up of two sets of convolutional layers each followed by average pooling layers. Both of the convolutional layers have eight 5×1 filters with ELU activations. The pooling layers use a stride and step of two. We apply global average pooling to the output of the convolutional block. The result is passed to a fully-connected network with three layers comprised of eight neurons each, and ELU activations. The output is four neurons, representing the values of N_d,A, N_d,B, N_d,C, and t_B. The Adam algorithm with the default hyperparameters was also used for this network with an initial learning rate of 10^{-4}. Note, to better handle the orders of magnitude spanned by N_d,A, N_d,B, and N_d,C, we used the base-10 logarithmic values, but use the true values when reporting the results. A Z-score normalization was also applied to these training labels. A set of 768 quasi-random values of N_d,A, N_d,B, N_d,C, t_B, and the 16 values of the four-dimensional hypercube were using to simulate the C–V profiles to create the training and validation sets, with 90% used for training and 10% for validation. Cases on the corners of the parameter hypercube are weighted higher during training.

4. Results and discussion

4.1. Multilayer perceptron

4.1.1. Learning curve and network generalization

The learning curve for the first neural network is shown in figure 3(a). The network exhibits low validation loss without overfitting, due to the network only containing 401 free variables and being trained with about 7000 samples. Figures 3(b) and (c) present prediction-expectation plots for the training and validation sets. An ideal predicted-expected curve would be a line with unity slope. The quality of fit for the neural network model is quite higher for both datasets, with R^2 values near one. In this case, the validation set contains the same values of N_d,A, N_d,B, N_d,C, and t_B that are present in the training set given how the two sets were constructed. The excellent performance of the network on the validation set confirms that the network is not

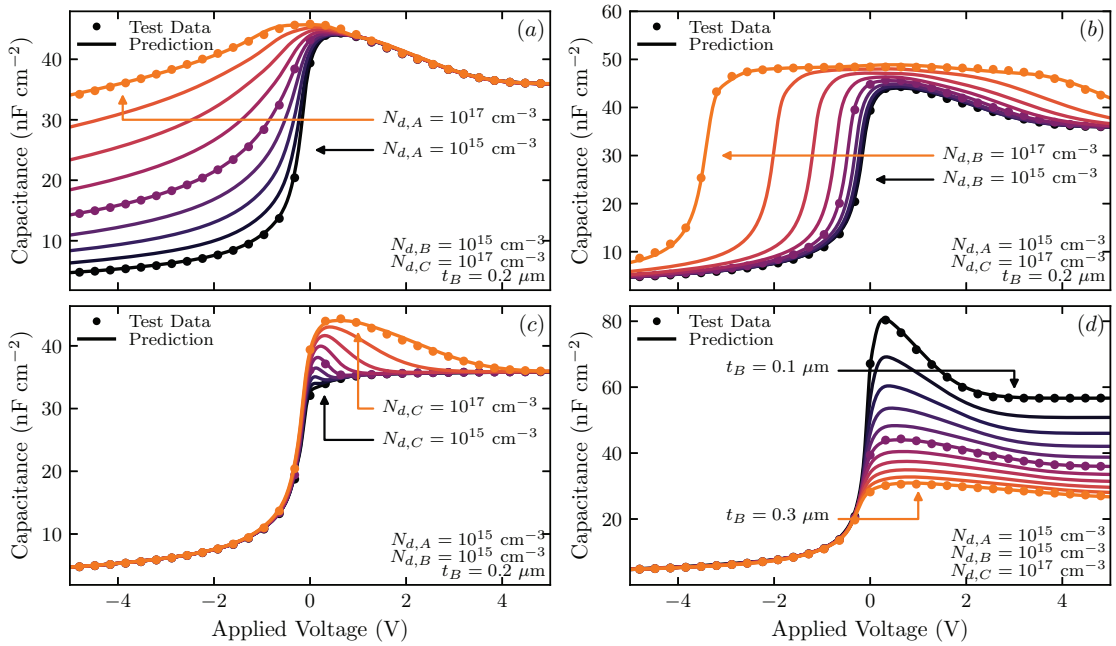

Figure 4. Single parameter sweeps of (a) $N_{d,A}$, (b) $N_{d,B}$, (c) $N_{d,C}$, and (d) t_B, while the other features are fixed, showing neural network generalization to unseen values. All lines are neural network predictions for the capacitance. Markers indicate samples from the test set, which contain values of N_d,A, N_d,B, N_d,C, and t_B that were not used during training.

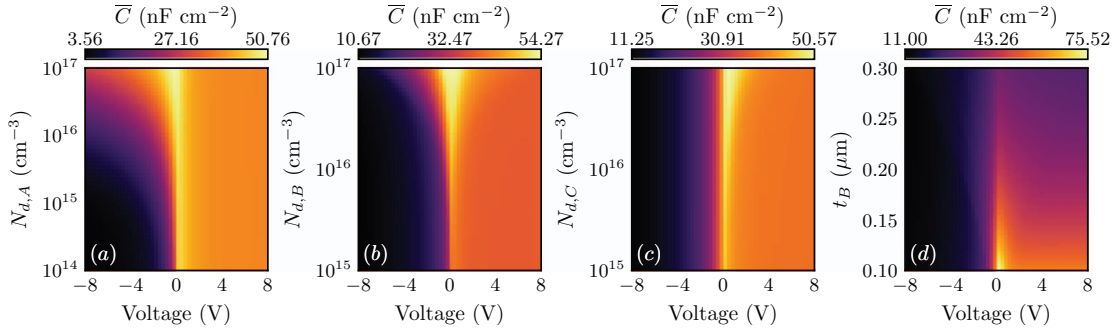

Figure 5. Bivariate partial dependence of the capacitance computed as functions of the applied voltage along with (a) N_d,A, (b) N_d,B, (c) N_d,C, and (d) t_B. The partial dependence value $\bar{C}$ is calculated by fixing V_a and another feature, and averaging the predicted capacitance for each combination of other features in a gridded feature-space. $\bar{C}$ has units of nF cm^{-2}.

overfitting the training data, and generalizes well to new voltage values. To confirm generalization to unknown values of N_d,A, N_d,B, N_d,C, and t_B another dataset was generated. The same predicted-expected curves are shown in figure 3(d), and confirms the ability of the network to generalize well to unseen cases.

Figure 4 presents examples of predicted C–V profiles by the network through univariate parameter sweeps. Note, the network was not trained using any of the exact cases of N_d,A, N_d,B, N_d,C, and t_B shown in figure 4. For each univariate sweep, the predicted C–V curve smoothly follows the expected trends for each parameter, while matching the test data well. For example, in figure 4(a) N_d,A is varied between 10^{15} and 10^{17} cm^{-3}. The network correctly captures the trend where a larger reverse bias applied to the contact layer is required to expand the depletion region in the absorber for high doping densities, manifesting as a capacitance that drops more slowly with voltage. Physically this is due to how the depletion region in the absorbing layer expands with reverse bias; generally, a larger reverse bias is required to extend the depletion region in highly doped semiconductors, and hence, the capacitance will drop at a lower rate with reverse bias. The trained neural network accurately captures this behavior.

The barrier doping density, N_d,B, is varied in figure 4(b). The barrier doping density affects the C–V by changing the flatband voltage where either the absorber (contact) layer changes from accumulation (depletion) to depletion (accumulation) under forward (reverse) or reverse (forward) bias respectively. This voltage is the point at which the capacitance starts to decrease with bias, and indicates that the absorber (under reverse bias) or the contact layer (under forward bias) has a sufficiently large depletion region to

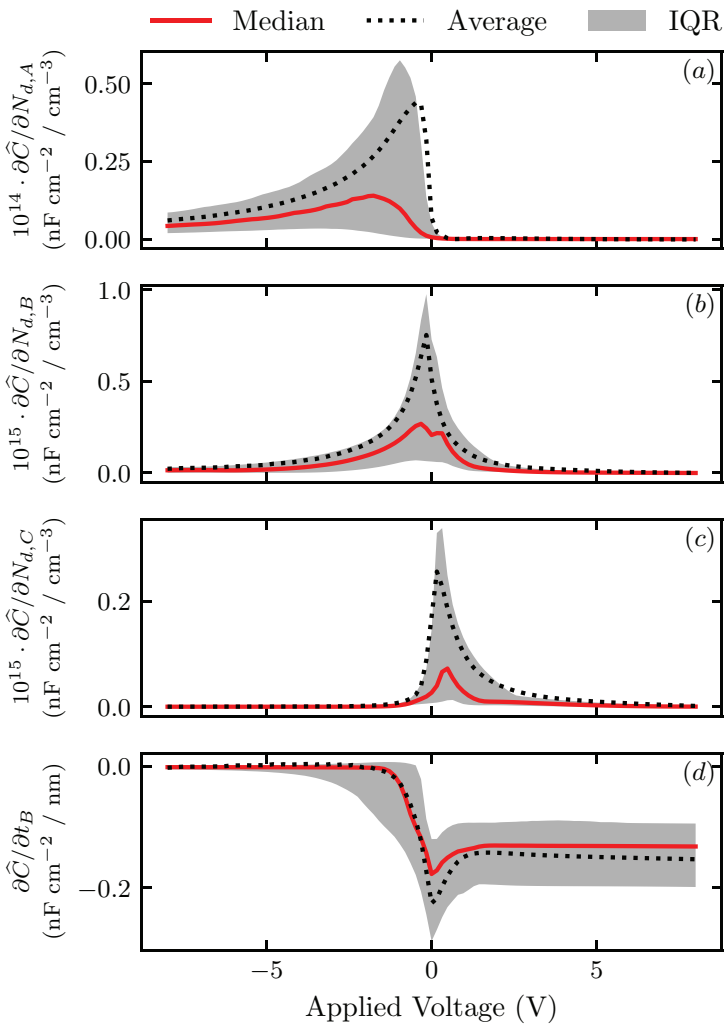

Figure 6. The distribution of partial derivatives of the predicted capacitance, $\widehat{C}$, with respect to (a) N_d,A, (b) N_d,B, (c) N_d,C, (d) t_B. Solid red and dotted black denote the median and average of the distribution, and the shaded region is the interquartile range.

dominate in equation (2). As the barrier doping density increases, the magnitude of the crossover voltage should also increase, as noted in previous works [25, 29]. Again, the neural network accurately reproduces the expected behavior.

The trend in figure 4(c) is the same as in (a), except for forward bias. Under forward bias the contact layer depletes, and can limit the device capacitance. Unlike the absorber, however, the contact is only 100 nm thick, and can fully deplete under forward bias when the doping density is low. When this happens, the capacitance approaches a constant value. Once more, the network is able to reproduce both of these physical phenomena.

Finally, figure 4(d) varies the thickness of the barrier layer. The thickness of the barrier affects the capacitance by increasing the dielectric width in a parallel plate capacitor, as in equation (3). The parallel plate capacitance decreases as the barrier thickness increases, and further limits equation (2). Just like the previous cases, the neural network is able to capture this trend with high fidelity.

4.1.2. Partial dependence plots

One way to gain a global perspective on how each feature shapes the output is to generate partial dependence plots [49–51]. Partial dependence can help the user in understanding the role of a feature in the prediction of any black box function. Figure 5 shows bivariate partial dependence plots illustrating the average predicted capacitance as functions of the applied voltage and other features. We computed partial dependence by using the following approach. First, we create sets of discrete values for N_d,A, N_d,B, N_d,C, t_B, and V_a in ranges 10^{14}–10^{17} cm^{-3}, 10^{15}–10^{17} cm^{-3}, 10^{15}–10^{17} cm^{-3}, 100–300 nm, and −8 to 8 V respectively. Then, one value of V_a is fixed along with one value of the other feature we are investigating. For each combination of values of the other features, along with this pair of fixed values, we compute the average predicted capacitance, $\bar{C}$. This

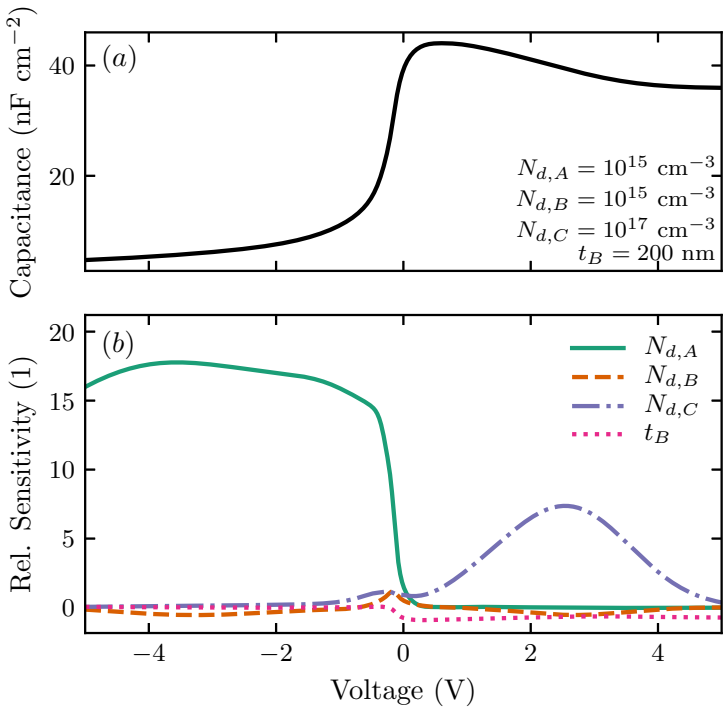

Figure 7. (a) An example of a C–V profile for reference when considering the relative feature sensitivity in (b). Curves in (b) were computed using equation (4).

represents the partial dependence of the capacitance with respect to the fixed features by averaging the effect from the others.

Using figure 5(a) as an example, we can observe how the capacitance is shaped by the absorber doping density. N_d,A has little affect on the value of the forward bias capacitance, as shown by a constant value of $\bar{C}$ with varied N_d,A. This is consistent with the previous discussion of figure 4; under forward bias the absorber-barrier interface is under accumulation, and has a larger capacitance than either the parallel plate capacitance determined by the barrier thickness, or the contact layer's junction capacitance. On average, the capacitance rapidly drops under reverse bias for low values of N_d,A, consistent with a low doped absorber layer rapidly depleting.

Shown in figure 5(b), N_d,B broadens the C–V profile for large doping densities, consistent with how N_d,B affects the crossover voltages for the absorber and contact layers. Similarly to figures 5(a) and (c) shows that N_d,C only affects the C–V under forward bias, and that for low values of N_d,C the capacitance quickly reaches a constant value, again consistent with the layer fully depleting. Figure 5(d) reveals more information about the t_B dependence of the C–V than figure 4(d). On average, t_B is important over the full voltage range, since t_B can directly limit the total capacitance in equation (2) through C_B.

4.1.3. Sensitivity analysis

The partial dependence plots in the previous section are useful for providing insight into the average value of the prediction due to each feature. Another analysis that can be useful is to quantify how sensitive the output is to changes in the features. With automatic differentiation we have direct access to the gradient of the capacitance with respect to the inputs. Using a similar approach to the partial density plots, we create a set comprised of every possible combination of features in the parameter-space, compute the gradients, and find the interquartile range, median, and average values. The result is shown in figure 6, where for every voltage we have computed a distribution of partial derivatives of the capacitance with respect to each of the features. This enables both a qualitative and quantitative understanding of the extent of the effect that each has on the capacitance. For example, as previously discussed, figure 6(a) shows that changes in N_d,A have little impact on the capacitance under forward bias, and is most influential at low reverse bias. However, it also shows that a incremental change of N_d,A by 10^{14} cm^{-3}, the capacitance, on average, will rise by 0.2 nF cm^{-2} at about -2 V.

Of course, we can also study a specific case, and look at the gradients to understand which parameter should be changed to meet a design requirement. For example, depending on the application it may be beneficial to reduce the capacitance under reverse bias to increase the speed of the device. Figure 7 shows an

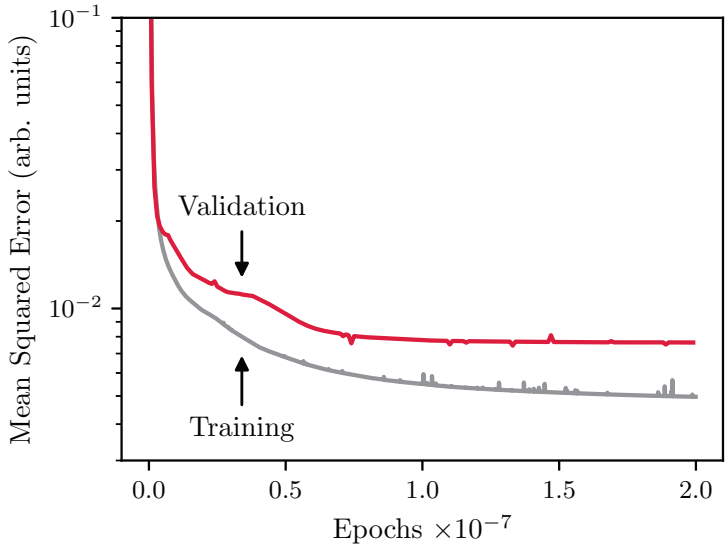

Figure 8. Learning curve for training a convolutional neural network to predict N_d,A, N_d,B, N_d,C, and t_B from a C–V characteristic. The y-axis is the average of the mean squared errors for each of the four outputs.

Table 2. A summary of the performance of a trained convolutional neural network used to predict N_d,A, N_d,B, N_d,C, and t_B from a C–V profile.

Feature	Dataset	Avg. % Err.	Median % Err.	R^2
N_d,A	Training	−0.273	0.100	0.9931
	Validation	−0.123	0.941	0.9941
N_d,B	Training	−0.940	0.052	0.9815
	Validation	−5.830	−3.145	0.9797
N_d,C	Training	−0.650	−0.275	0.9838
	Validation	3.303	3.510	0.9930
t_B	Training	−0.015	−0.041	0.9982
	Validation	−0.055	−0.043	0.9991

example where we are considering the bias-dependence of the capacitance with values of 10^{15} cm^{-3}, 10^{15} cm^{-3}, 10^{17} cm^{-3}, and 200 nm for N_d,A, N_d,B, N_d,C, and t_B respectively. We directly compare the impact by each feature by calculating a relative sensitivity given by

$$S_i^* = \frac{\partial C}{\partial x} \cdot \frac{x}{C}, \tag{4}$$

for $x \in N_d$, A, N_d,B, $\{N_d$,C, $t_B\}$. The figure clearly shows that reducing N_d,A would have the largest effect in achieving a lower capacitance, but also offers information on the other features as well.

4.2. Convolutional neural network

The previous discussion established neural networks as a robust tool for predicting and analyzing device characteristics. In this section we consider the inverse case, where a convolutional neural network is trained to characterize the device's architectural properties. From this perspective, the neural network is used as an enhanced metrology technique for extracting more information from the C–V characteristic.

The learning curve for training the convolutional neural network to predict N_d,A, N_d,B, N_d,C, and t_B is shown in figure 8, and the quality of the fit is summarized in table 2. Since the complexity of the model was reduced to accommodate the limited number of training samples, the validation error follows the training error without overfitting. The average and median errors in both the training and validation sets are low, with R^2 values are near unity for all cases. With more samples, and a more complex network, these errors could be reduced further.

The performance for the convolutional network is shown in figure 9. Shown in figure 9(a), the absorber doping density is accurately predicted for all cases within 10^{14}–10^{17} cm^{-3}, with a majority of cases falling within 0.8 to 1.2× the expected value. While a 20% error in these outliers may seem high, to put it in perspective this would indicate that a doping density of 10^{15} cm^{-3} would be either over- or underestimated

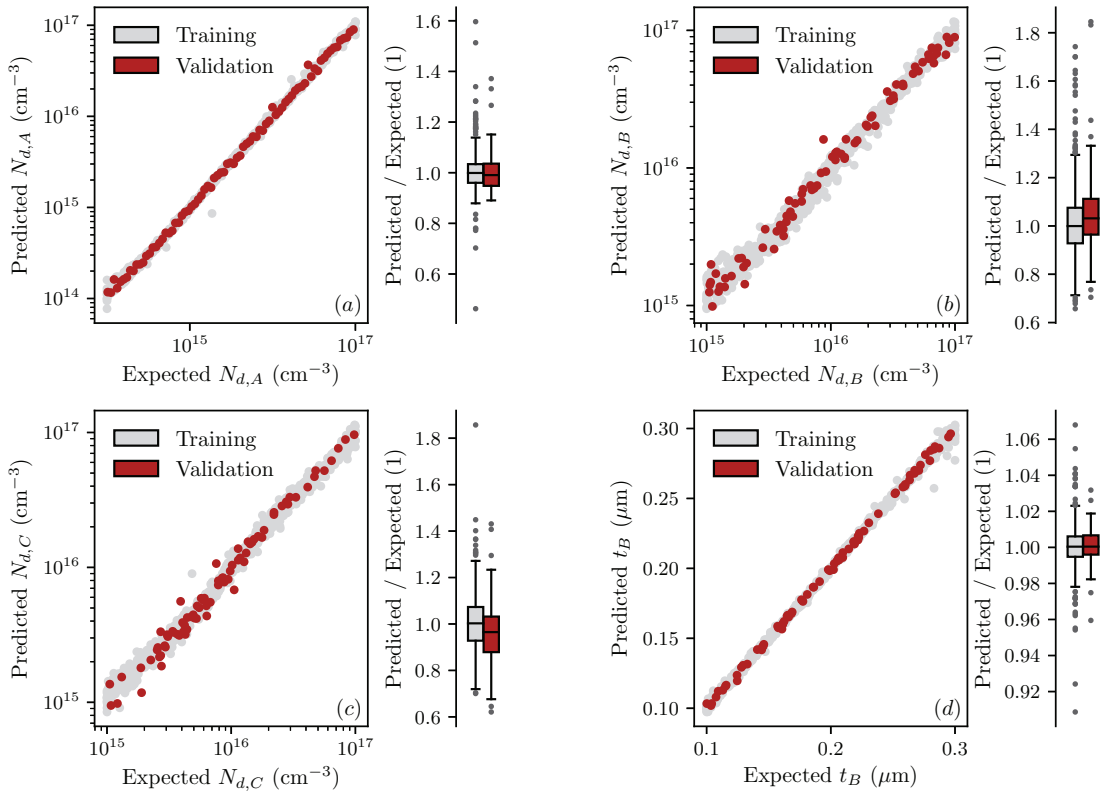

Figure 9. Predicted versus expected curves for (a) N_d,A, (b) N_d,B, (c) N_d,C, (d) t_B and the associated distribution of predicted-expected ratios.

as 1.2×10^{15} or 8×10^{14} cm^{-3}, which for these orders of magnitude is reasonable. Similar behaviors are seen for N_d,B and N_d,C; all doping densities are correctly predicted within a factor of two of the expected values. The thickness of the barrier layer very accurately predicted, with the worst case underestimating by about 25 nm; most cases are within 5–15 nm depending on the thickness of the layer. Nearly all of distributions have medians near unity.

Figure 9 also provides insight into cases that are difficult to predict or obtain information about the features. Cases of low N_d,C and high t_B have wider distributions of predictions. For low N_d,C, the contact layer quickly depletes and the capacitance is constant without any distinctive features, similar to what was shown in figure 4(c). Similarly, a large value of t_B causes the C–V to flatten has the parallel plate capacitance from the barrier dominates, reducing the important of aspects of the profile, like shown in figure 4(d).

5. Conclusion

In this work, we introduced a methodology for applying machine learning to analyze and characterize semiconductor devices based on their electrical characteristics. Specifically, we looked at two cases using ANNs trained with C–V data obtained by drift-diffusion simulations of InAsSb barrier infrared detectors. First, we created a simple model to predict the capacitance based on supplied values for the absorber, barrier, and contact layer doping densities, and the barrier thickness. We demonstrated that this model provides a high quality fit, generalizes well to unseen data, and reproduces the behavior expected when considering the underlying device physics that determine the capacitance of the device. Moreover, we showed the usefulness of this approach in analyzing the global parameter space associated with the prediction through partial dependence plots. This approach was also shown to have potential use in helping with critical design decisions by analyzing the capacitance's sensitivity to each feature.

While capacitance is not typically the critical metric used when discussing photodetector parameters, it is a useful diagnostic tool, and elucidates more about this particular type of infrared photodetector than other competing designs. However, the methodology discussed here is agnostic of the characteristic, so while we focused on the C–V relationship, the approach could be extended to other figures of merit and used to optimize performance around different sets of features. For instance, a model could be created to include an additional metric, such as dark current, and used to understand the tradeoff between lowering capacitance and photodetector noise with the objective of meeting application-specific performance goals. Additional

features could also be included to create a more comprehensive representation of the device for further optimization. Finally, it is worth noting that the methodology introduced in this paper is agnostic of the material or device under study, so a similar approach could be used to study other types of devices—such as those that use strained superlattices as absorbing layers—their performance, and the role of their associated design features. The key challenge in extending the methodology to other devices lies in the ability to generate or acquire additional data.

The second contribution from this work was the demonstration of using a trained convolutional neural network to analyze a barrier detector's C–V characteristics to predict N_d,A, N_d,B, N_d,C, and t_B. The model provides value as a characterization tool by alleviating constraints imposed by other analysis approaches and enabling the possibility of extracting additional diagnostic information from a measurement. The model was shown to perform well over a wide range of absorber, barrier, and contact layer doping densities, and barrier thicknesses. While this study was limited to a small number of device parameters, the complexity of the model could be increased by simulating additional cases to encompass more details about a device, including for example, the thickness and composition of each semiconducting layer. Of course, inclusion of additional features may require a more complex network and acquisition of additional data, leading to a higher computational cost in training the model to achieve a reasonable accuracy. Another opportunity would be to train the model to characterize heterointerfaces and other semiconductor surface properties by characterizing band offsets, defect energy levels, or trap densities. In the future, it could prove useful to explore the feasibility of using data augmentation or synthetic data generated by the feedforward model, or both, to improve model generalization and prediction accuracy.

The data that support the findings of this study may be made available upon reasonable request from the authors.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgment

The authors gratefully acknowledge financial support from the U.S. Army Research Laboratory through the Center for Semiconductor Modeling under Grant No. W911NF-12-2-0023 managed by Dr Meredith Reed. Computational resources were provided by an Army Research Office DURIP Award under Grant No. W911NF-19-0161.

ORCID iDs

Andreu Glasmann https://orcid.org/0000-0002-1179-3240
Alexandros Kyrtsos https://orcid.org/0000-0002-8916-3791

References

[1] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[2] Jordan M and Mitchell T 2015 Machine learning: trends, perspectives and prospects *Science* **349** 255
[3] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436
[4] Larrañaga P *et al* 2006 Machine learning in bioinformatics *Briefings Bioinform.* **7** 86
[5] Pilania G, Gubernatis J and Lookman T 2017 Multi-fidelity machine learning models for accurate bandgap predictions of solids *Comput. Mater. Sci.* **129** 156
[6] Schmidt J, Marques M R, Botti S and Marques M A 2019 Recent advances and applications of machine learning in solid-state materials science *npj Computat. Mater.* **5** 1
[7] Rajan K 2005 Materials informatics *Mater. Today* **8** 38
[8] Choy G *et al* 2018 Current applications and future impact of machine learning in radiology *Radiology* **288** 318
[9] Lundervold A S and Lundervold A 2019 An overview of deep learning in medical imaging focusing on MRI *Zeitschrift für Medizinische Physik* **29** 102
[10] Litovski V B, Radjenovic J I, Mrcarica Z M and Milenkovic S L 1992 MOS transistor modelling using neural network *Electron. Lett.* **28** 1766
[11] Root D E 2012 Future device modeling trends *IEEE Microw. Mag.* **13** 45
[12] Huang A, Zhong Z, Wu W and Guo Y 2016 An artificial neural network-based electrothermal model for GaN HEMTs with dynamic trapping effects consideration *IEEE Trans. Microw. Theory Tech.* **64** 2519
[13] Hammouda H B, Mhiri M, Gafsi Z and Besbes K 2008 Neural-based models of semiconductor devices for SPICE simulator *Am. J. Appl. Sci.* **5** 385
[14] Irani K B, Cheng J, Fayyad U M and Qian Z 1993 Applying machine learning to semiconductor manufacturing *IEEE Expert* **8** 41
[15] Moyne J and Iskandar J 2017 Big data analytics for smart manufacturing: case studies in semiconductor manufacturing *Processes* **5** 39

[16] Susto G A, Pampuri S, Schirru A, Nicolao G D, McLoone S F and Beghi A 2012 Automatic control and machine learning for semiconductor manufacturing: review and challengess *10th European Workshop on Advanced Control and Diagnosis (ACD 2012)* (http://mural.maynoothuniversity.ie/4184/)
[17] Susto G A, Schirru A, Pampuri S, McLoone S and Beghi A 2015 Machine learning for predictive maintenance: a multiple classifier approach *IEEE Trans. Ind. Inform.* **11** 812
[18] Nakata K, Orihara R, Mizuoka Y and Takagi K 2017 A comprehensive big-data-based monitoring system for yield enhancement in semiconductor manufacturing *IEEE Trans. Semicond. Manuf.* **30** 339
[19] Lee K B, Cheon S and Kim C O 2017 A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes *IEEE Trans. Semicond. Manuf.* **30** 135
[20] Huang S-H and Pan Y-C 2015 Automated visual inspection in the semiconductor industry: a survey *Comput. Ind.* **66** 1
[21] Melati D, Grinberg Y, Dezfouli M K, Janz S, Cheben P, Schmid J H, Sánchez-Postigo A and Xu D-X 2019 Mapping the global design space of nanophotonic components using machine learning pattern recognition *Nat. Commun.* **10** 1
[22] White A M 1987 Infra red detectors, note *US Patent Specification* 4,679,063
[23] Maimon S and Wicks G 2006 nBn detector, an infrared detector with reduced dark current and higher operating temperature *Appl. Phys. Lett.* **89** 151109
[24] Klipstein P *et al* 2010 MWIR InAsSb XBn detectors for high operating temperatures *Infrared Technology and Applications XXXVI Orlando, FL, USA* (Bellingham, WA: Int. Society for Optics and Photonics) **7660** 76602Y
[25] Reine M, Pinkie B, Schuster J and Bellotti E 2014 Numerical simulation and analytical modeling of InAs nBn infrared detectors with n-type barrier layers *J. Electron. Mater.* **43** 2915
[26] Martyniuk P and Rogalski A 2014 Performance limits of the mid-wave InAsSb/AlAsSb nBn HOT infrared detector *Opt. Quantum Electron.* **46** 581
[27] Schroder D K 2006 chapter 2 *Semiconductor Material and Device Characterization* (New York: Wiley) pp 61–5
[28] Rhiger D R, Smith E P, Kolasa B P, Kim J K, Klem J F and Hawkins S D 2016 Analysis of III–V superlattice nBn device characteristics *J. Electron. Mater.* **45** 4646
[29] Glasmann A, Prigozhin I and Bellotti E 2019 Understanding the C-V characteristics of InAsSb-based nBn infrared detectors with N-and P-Type barrier layers through numerical modeling *IEEE J. Electron Devices Soc.* **7** 534
[30] Klipstein P *et al* 2008 (Int. Society for Optics and Photonics) XBn barrier photodetectors for high sensitivty and high operating temperature infrared sensors *Infrared Technology and Applications XXXiV Orlando, FL, USA* Bellingham, WA (https://doi.org/10.1117/12.778848)
[31] Clevert D-A, Unterthiner T and Hochreiter S 2015 Fast and accurate deep network learning by exponential linear units (Elus) (arXiv:1511.07289)
[32] Long J L, Zhang N and Darrell T 2014 Do Convnets Learn Correspondence? *Advances in Neural Information Processing Systems 27 Palais des Congrès de Montréal, Montréal, CANADA* MIT Press San Diego, CA
[33] Zeiler M D and Fergus R 2014 Visualizing and understanding convolutional networks *European Conference on Computer Vision* (Berlin: Springer) pp 818–33
[34] Yosinski J, Clune J, Nguyen A, Fuchs T and Lipson H 2015 Understanding neural networks through deep visualization (arXiv:1506.06579)
[35] Acharya U R, Fujita H, Oh S L, Hagiwara Y, Tan J H and Adam M 2017 Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals *Inf. Sci.* **415** 190
[36] Kiranyaz S, Ince T and Gabbouj M 2015 Real-time patient-specific ECG classification by1-D convolutional neural networks *IEEE Trans. Biomed. Eng.* **63** 664
[37] Acharya U R, Oh S L, Hagiwara Y, Tan J H and Adeli H 2018 Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals *Comput. Biol. Med.* **100** 270
[38] Sze S M and Ng K K 2006 *Physics of Semiconductor Devices* (New York: Wiley) ch 4, pp 200–12
[39] Klem J F, Kim J K, Cich M J, Hawkins S D, Fortune T and Rienstra J 2010 (Int. Society for Optics and Photonics) Comparison of nBn and nBp mid-wave barrier infrared photodetectors *Quantum Sensing and Nanophotonic Devices VII San Francisco, CA* Bellingham, WA (https://doi.org/10.1117/12.842772)
[40] Abebe H, Morris H, Cumberbatch E and Tyree V 2007 Compact gate capacitance model with polysilicon depletion effect for MOS device *J. Semicond. Technol. Sci.* **7** 131
[41] Synopsys 2017 *Sentaurus Device User Guide* (N-2017.09)
[42] Schuster J, D'Souza A and Bellotti E 2014 Analysis of InAsSb nBn spectrally filtering photon-trapping structures *Opt. Express* **22** 18987
[43] Halton J H 1960 On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals *Numer. Math.* **2** 84
[44] Halton J H 1964 Algorithm 247: Radical-inverse quasi-random point sequence *Commun. ACM* **7** 701
[45] Kocis L and Whiten W J 1997 Computational investigations of low-discrepancy sequences *ACM Trans. Math. Softw. (TOMS)* **23** 266
[46] TensorFlow 2015 note software TensorFlow: large-scale machine learning on heterogeneous systems (https://www.tensorflow.org/)
[47] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. Thirteenth Int. Conf. on Artificial Intelligence and Statistics Sardinia, Italy* JMLR pp 249–56 (http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf)
[48] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[49] Hastie T, Tibshirani R, Friedman J and Franklin J 2005 *The Elements of Statistical Learning: Data Mining, Inference and Prediction* 2nd edn (Berlin: Springer) ch 10, pp 369–70
[50] Greenwell B M 2017 pdp: an r package for constructing partial dependence plots *R Journal* **9** 421
[51] Friedman J H 2001 Greedy function approximation: a gradient boosting machine *The Annals of Statistics* **29** pp 1189-1232