IEEE Transactions on Quantum Engineering

Received 15 October 2024; revised 12 June 2025; accepted 2 August 2025; date of publication 6 August 2025; date of current version 9 September 2025.

Digital Object Identifier 10.1109/TQE.2025.3596392

Automated Charge Transition Detection in Quantum Dot Charge Stability Diagrams

FABIAN HADER[1], FABIAN FUCHS[1], SARAH FLEITMANN[1], KARIN HAVEMANN[1], BENEDIKT SCHERER[1], JAN VOGELBRUCH[1], LOTTE GECK[1,2], AND STEFAN VAN WAASEN[1,3]

[1]Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, 52425 Jülich, Germany
[2]Faculty of Electrical Engineering and Information Technology, RWTH Aachen University, 52062 Aachen, Germany
[3]Faculty of Engineering, University of Duisburg-Essen, 47057 Duisburg, Germany

Corresponding author: Fabian Hader (e-mail: f.hader@fz-juelich.de).

ABSTRACT Gate-defined semiconductor quantum dots require an appropriate number of electrons to function as qubits. The number of electrons is usually tuned by analyzing charge stability diagrams, in which charge transitions manifest as edges. Therefore, to fully automate qubit tuning, it is necessary to recognize these edges automatically and reliably. This article investigates possible detection methods, describes their training with simulated data from the SimCATS framework, and performs a quantitative comparison with a future hardware implementation in mind. Furthermore, we investigated the quality of the optimized approaches on experimentally measured data from a GaAs and a SiGe qubit sample.

INDEX TERMS Automated tuning, charge stability diagram (CSD), quantum computing, semiconductor quantum dots.

I. INTRODUCTION

Tuning the number of electrons in quantum dots (QDs) is essential for creating gate-defined semiconductor quantum bits (qubits). Two-dimensional charge stability diagrams (CSDs) reveal a change in the number of electrons, hereafter charge transition (CT), as an edge in the pixel information (usually floating-point representation). They can be recorded, for example, by using the conductance change of a nearby electrostatically coupled sensor dot or a quantum point contact [1]. The CTs must be detected robustly to realize complete automation of the tuning process. However, considering scalability, the complexity of the approaches should be minimized. Ultimately, we propose that automated tuning should be integrated into the cryostat to reduce the wiring problem (e.g., limited bandwidth and heat dissipation) originating from the connection of room temperature electronics into the cryostat [2], [3]. If it is impossible to embed all parts of the tuning (e.g., space and dissipated heat limitations), primarily integrating the initial data processing steps can reduce the amount of data to be transmitted. In particular, knowing only the binary edge information for individual pixels is sufficient for analyzing CTs. Therefore, we consider this step an essential candidate for cryogenic hardware implementation and investigate possible detection approaches, including classical and machine learning (ML) methods. We used the simulation framework SimCATS [4] to generate training and test data. In addition, we analyzed the applicability of the selected approaches to experimental data.

The rest of this article is organized as follows. In Section II, we comprehensively introduce state-of-the-art tuning approaches that use CT detection. Then, we describe the applied datasets in Section III and the metrics and methods selected for the evaluation in Section IV. Next, we describe the selected detection approaches in Section V and their training in Section VI. In Section VII, we evaluate their detection quality and, for the ML candidates, the number of parameters and speed. Finally, Section VIII concludes this article and presents potential improvements.

II. BACKGROUND

Before tuning the number of electrons in QDs, automated tuning approaches adjust the QD device to a stable global configuration of known topology in the state space with a known number of charge islands [5]. In our case, we form a double quantum dot (DQD) and tune the number of charges in each dot. A common strategy is to empty the QDs (unloading phase) and then reload the desired number of electrons on each (reloading phase). This procedure requires identifying CTs in the CSDs, as the exact number of charges cannot be sensed directly [5]. Subsequently, fine-tuning of couplings

between multiple dots is typically performed. Different authors used classical image processing and different kinds of ML methods for these tasks.

Approaches to tuning qubits to a stable known topology comprise classical ML methods and deep learning methods. Darulová et al. [6] used fitting procedures and compared different classical classifiers trained on simulated and experimental data. The cross-architecture tuning solution using artificial intelligence [7] uses a Gaussian process model of the gate voltage hypersurface and a random classifier iteratively to tune multiple parameters at once. The proposed approach was demonstrated on three device architectures and material systems. Deep learning methods for this task use proprietary convolutional neural networks (CNNs) [8], [9], [10], [11] or the AlexNet model [12]. The networks are either trained on simulated data only [8], [9], [10], [11] or a mix of experimental and simulated data [12].

First approaches to tuning QDs to a specific charge regime involved classical image processing. Baart et al. [13] used template matching (Gabor filter) to identify the most bottom-left CTs and set voltages slightly above to enter the single-electron regime. For single QDs, Lapointe-Major et al. [14] suggested using a modified Hough transform or the EDLines algorithm. Classical ML methods were proposed by Moon et al. [15] and van Straaten et al. [16]. Moon et al. [15] used a probabilistic ML model in an iterative and two-stage manner to identify candidate locations on the gate voltage hypersurface, measure the data therein, and evaluate the transport features. In contrast, van Straaten et al. [16] moved to hypervolumes and performed a hypothesis (Kolmogorov–Smirnov) test if the volume contains only noise. If not, a random walk combined with a score function is employed to search for DQD features near hypervolumes. The deep learning approaches for charge regime tuning mainly differ in CNN models, training data, and data dimensionality. Durrer et al. [17] used several different CNNs to predict the presence of CTs in the unloading phase and the presence and orientation for the reloading phase of a GaAs triple-QD device operated in the DQD mode. The work of [18] classifies preprocessed CSD patches (5×5 pixel) via an extremely small feedforward neural network to predict the presence of CTs and to enable a future network in memristor arrays. The model was trained on synthetic data and robustly transferred to experimental data. The physics-informed tuning proposed by Ziegler et al. [19] uses rays rather than 2-D measurements. The method combines the ray-based classification classifier [20] or a CNN [21] with physics knowledge to first navigate to the target global state and then performs ray measurements to tune to the target charge state.

Fine-tuning tunnel couplings also include proposed methods from all categories. van Diepen et al. [22] fitted the interdot transition (IDT) sensor signals to a classical anticrossing model. Mills et al. [23] established virtual gates before using the Hough line transform and template matching for interdot coupling tuning. Similarly, Monical et al. [24] used a generalization of the Hough transform (Hough anticrossing transform) but detected the locations and inclinations of possible triple points. Teske et al. [25] proposed a classical ML method by combining Bayesian statistics with adapted Kalman filtering. The deep learning approach of [26] realizes an unsupervised generative model that employs variational autoencoders, consisting of an encoder and decoder both embodied in a neural network.

The work in [12] and [27] comprises all of the aforementioned steps [12] or claims to perform fully autonomous tuning to Rabi oscillations [27]. Both methods use deep learning or Bayesian optimization and computer vision techniques in the case of [27]. Deep learning has also been used to perform autonomous measurements [28], [29] or single-shot readouts of charge and spin states [30].

III. DATASETS

We generated datasets[1] for training and evaluating edge detection approaches using our simulation framework SimCATS [4] and used hand-labeled experimental data from the Quantum Technology Group of RWTH Aachen[2] to assess the generalization ability.

Some neural network architectures require image resolutions divisible by a power of 2, e.g., because the resolution is halved in each encoder step and doubled in the decoder. With at most five such steps for the investigated models, we required a resolution divisible by 32. Since the initially available experimental data from a GaAs sample[3] with a QD employed as sensor dot have a resolution of 100×100 pixels ($30\,\text{mV} \times 30\,\text{mV}$), we chose the closest resolution of 96×96 pixels ($28.8\,\text{mV} \times 28.8\,\text{mV}$) to be able to test the networks later on experimental data.[4]

We obtained parameter ranges for the SimCATS simulations from the data of the GaAs sample, as described in [4]. Furthermore, we aimed for a diverse dataset to improve the networks' generalizability. Therefore, the generation procedure randomly selected all model parameters from the extracted ranges listed in Table 1. While sampling most values from a uniform distribution, some parameters that describe the structure of the total charge transitions (TCTs)[5] were sampled from a normal distribution because this matches our observations in the experimental data. In addition to the parameters described in [4], we supply the following parameters for the lead-to-dot transitions (LDTs) and IDTs[6] of the ith TCT:

[1] Using our Python package `SimCATS-Datasets` [31].

[2] https://www.quantuminfo.physik.rwth-aachen.de/cms/quantuminfo/forschung/~xwpl/quantum-technology-group/

[3] Similar to the one described in [32].

[4] The experimental GaAs data were reduced by removing the first and last two rows and columns.

[5] TCTs represent the borders between regions of the CSD containing a fixed number of electrons in the system. The ith TCT separates the regions containing $i-1$ and i electrons.

[6] An LDT describes a transition between the dot system and the leads and an IDT describes the tunneling of an electron between two dots.

IEEE Transactions on Quantum Engineering

TABLE 1. Parameter Ranges for the Simulated Datasets and Their Distributions [Uniform $U(a, b)$, Normal $\mathcal{N}(\mu, \sigma^2)$, or Exponential $\exp(\lambda)$]

	Parameter	Minimum	Maximum	Distribution
Total charge transitions	V'_{P1} intercept of $ldt_{i,j}$, $(j{=}1, 2)$ $(s_{i,j})$ [V]	$1.0{\cdot}10^{-2}$	$2.5{\cdot}10^{-2}$	$\mathcal{N}(\mu, \sigma^2)$
	Slope of $ldt_{i,1}$ $(m_{i,1})$	$-4.4{\cdot}10^{-1}$	$-8.0{\cdot}10^{-2}$	$\mathcal{N}(\mu, \sigma^2)$
	Slope of $ldt_{i,2}$ $(m_{i,2})$	$2.1{\cdot}10^{-1}$	$5.5{\cdot}10^{-1}$	$\mathcal{N}(\mu, \sigma^2)$
	Angle between $ldt_{i,1}$ and $ldt_{i,2}$ (θ_{ld_i}) [rad]	$4.4{\cdot}10^{-1}$	1.7	
	Angle between idt_i and $ldt_{i,2}$ (θ_{id_i}) [rad]	$5.8{\cdot}10^{-1}$	1.4	$U(a, b)$
	Length of idt_i (s_{id_i}) [V]	$2.6{\cdot}10^{-3}$	$9.9{\cdot}10^{-3}$	$U(a, b)$
	Width of idt_i (w_{id_i}) [V]	$4.3{\cdot}10^{-4}$	$8.1{\cdot}10^{-3}$	$U(a, b)$
	Relation of idt_i length to w_{id_i}	$8.7{\cdot}10^{-1}$	9.1	
Sensor	Number of Coulomb peaks	3	6	$U(a, b)$
	Lorentzian scaling factor influencing the height of the Coulomb peaks (a)	$2.2{\cdot}10^{-2}$	$1.9{\cdot}10^{-1}$	$\exp(\lambda)$
	Lorentzian width influencing the width of the Coulomb peaks (γ)	$9.6{\cdot}10^{-4}$	$3.0{\cdot}10^{-3}$	$U(a, b)$
	Lever arm dot 1, coupling between dot 1 and the sensor dot (α_1)	$-8.0{\cdot}10^{-4}$	$-9.6{\cdot}10^{-5}$	$U(a, b)$
	Lever arm dot 2, coupling between dot 2 and the sensor dot (α_2)	$-5.2{\cdot}10^{-4}$	$-6.3{\cdot}10^{-5}$	$U(a, b)$
	Lever arm gate 1, coupling between plunger gate 1 and the sensor dot (β_1)	$2.8{\cdot}10^{-2}$	$1.5{\cdot}10^{-1}$	$U(a, b)$
	Lever arm gate 2, coupling between plunger gate 2 and the sensor dot (β_2)	$1.4{\cdot}10^{-2}$	$3.0{\cdot}10^{-1}$	$U(a, b)$
	Relation of β_1 to β_2	$5.0{\cdot}10^{-1}$	2.0	
Occupation distortions	Fermi–Dirac transition blurring `sigma`	$7.5{\cdot}10^{-5}$	$6.0{\cdot}10^{-4}$	$\mathcal{N}(\mu, \sigma^2)$
Sensor potential distortions	Pink noise `sigma`	$1.0{\cdot}10^{-10}$	$5.0{\cdot}10^{-4}$	$\exp(\lambda)$
Sensor response distortions	White noise `sigma`	$1.0{\cdot}10^{-10}$	$5.0{\cdot}10^{-4}$	$\exp(\lambda)$

Table rows that do not include information about a distribution are only used for a validity check of the relation of sampled parameters. The parameter names in parentheses refer to the variables used in the SimCATS paper [4] and in Fig. 1. TCT parameters are given in the rotated voltage space (V'_{P1}, V'_{P2}). In addition to the distortions mentioned here, we also applied dot jumps as occupation distortion and random telegraph noise as sensor potential and sensor response distortion, using the parameters from the original SimCATS default_configs["GaAs_v1"] [33].

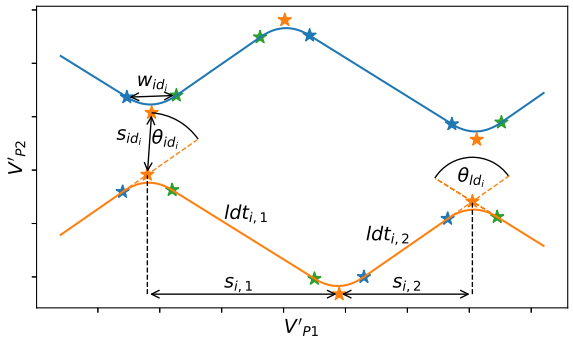

FIGURE 1. Visualization of parameters used to define the TCTs for the geometric simulation in SimCATS.

1) the relation between the slopes of the LDT of dot 1 $\text{ldt}_{i,1}$ and the LDT of dot 2 $\text{ldt}_{i,2}$ (θ_{ld_i});
2) the relation between the slopes of the IDT idt_i and $\text{ldt}_{i,1}$ (θ_{id_i});
3) the length of idt_i (s_{id_i});
4) the width of idt_i (w_{id_i})[7];
5) the relation between s_{id_i} and w_{id_i}.

Fig. 1 visualizes the various parameters of the used TCT model, with the TCTs displayed in the 45° rotated voltage space, which is denoted as (V'_{P1}, V'_{P2}) (see [4, Sect. III.A]). In anticipation of future improvements in sample quality, we decided to include lower noise levels more often than the very high levels that are sometimes observed in the measurements. Therefore, we used an exponential distribution for these values.

For normal distributions, we set μ to the center and σ to a sixth of the range to best represent the distribution in the interval. Furthermore, we selected the scale for the exponential rate to reach the 99% quantile at 60% of the sampling range. Generally, we resampled parameters outside the given ranges.

We do not require that the sampled parameters are physically plausible, as they differ between different samples, and experimental observations are not necessarily interpretable. Thus, we expect a more diverse dataset to lead to better generalization, which also leads to a better technology independence for the trained models.

The simulated train/validation/test dataset featured 10 000/1000/1000 different TCT and sensor configurations, each with 100 CSDs generated with randomly sampled distortion strengths.

The two experimental test datasets consisted of 439 (plunger, plunger) GaAs CSDs and 81 (plunger, barrier) SiGe[8] CSDs. As visible in Fig. 2, the GaAs sample data show DQD features, and the SiGe data single QD features. They have a resolution of 96 × 96 pixels (100 mV × 100 mV and 150 mV × 150 mV). We selected the SiGe voltage ranges to achieve a good balance between microscopic and macroscopic features compared with the GaAs data, although we still observed different properties. Specifically, the CTs are more blurred, and the spacing between CTs is lower, leading to more CTs per image. Nearly all SiGe CSDs featured visible CTs, as the Quantum Technology Group of RWTH Aachen specifically recorded them for our evaluation.

While the ground truth CT masks for the simulated datasets are available automatically, we manually generated masks for the experimental data. Therefore, five researchers labeled all images manually, and we combined the labels into a single binary mask afterward to improve the representation of weak lines. Notably, the SiGe data were more difficult to

[7]The width of idt_i defines the rounding at the triple points, which is controlled by the distance between the Bézier anchors $b_{i,j}$, $(j{=}1, 2)$ (see [4, Sect. III.A]).

[8]Sample similar as described in [34]

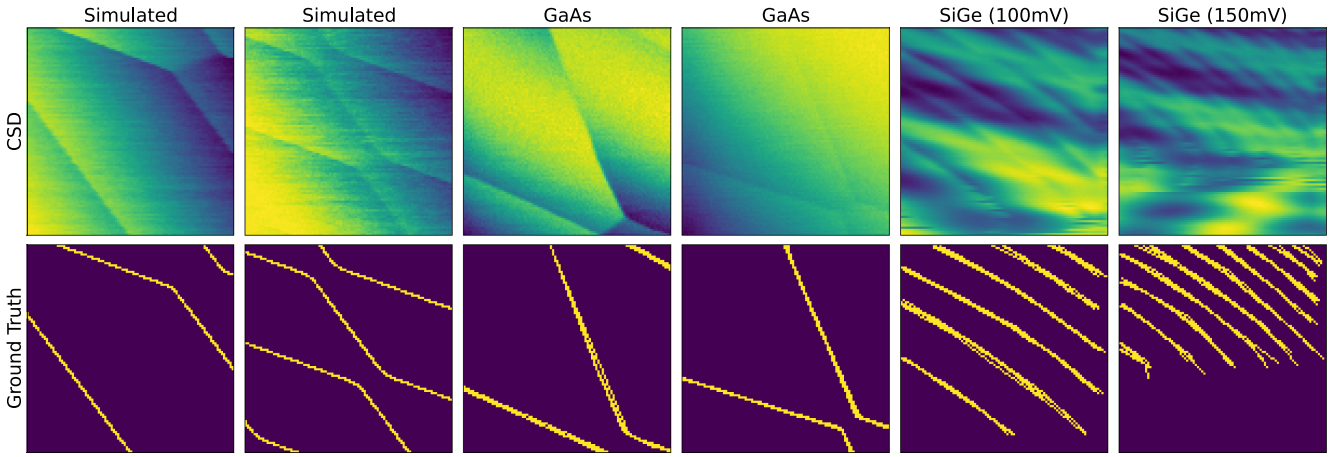

FIGURE 2. Examples for CSDs of the final evaluation datasets and their corresponding ground truth masks.

label, which resulted in broader line segments for the combined manual labels (see Fig. 2).

IV. METRICS AND EVALUATION METHODS

We used the Dice similarity coefficient (DICE) to compare the results of the different approaches and assess their quality

$$\mathrm{DICE} = \frac{2|X \cap Y|}{|X| + |Y|}.$$

This metric measures the similarity between the predicted segmentation mask X and the ground truth segmentation mask Y. In our implementation, we set the DICE score to 1 in case both masks are empty. The disadvantage of the DICE score is that it expects pixel-precise segmentation and severely punishes even misalignments of one pixel. Due to the manual labeling, we did not expect pixel-precise segmentation of the experimental data. The normalized surface Dice (S-DICE) score [35] solves this problem by introducing a class-specific threshold for an accepted segmentation deviation in the pixel space.

In addition, we measured the inference time of the models using the CUDA application programming interface because analysis speed is a crucial factor for a scalable tuning solution. The inference time specifies the calculation time required by the neural network for prediction. It does not include the time required to transfer data into GPU memory, which is identical for all approaches.

Furthermore, we used the Python package Calflops [36] to calculate the floating-point operations (FLOPs) and multiply-accumulate operations (MACs) of the networks. These values are independent of the hardware used and, therefore, offer a reasonable estimate of the feasibility of cryogenic implementation on dedicated hardware. Moreover, we compared the number of trainable parameters, which indicates the expected implementation size.

TABLE 2. Collected Approaches, Their Category, Publication Year, and Code Basis

Detector Category	Detector Name	Year	Code Basis
Convolution	CASENet	2017	[37]
	CHRNet	2023	[38]
	DeepLabV3+	2018	[39]
	DFF	2019	[37]
	FPN	2016	[39]
	LDC	2022	[40]
	LinkNet	2017	[39]
	TEED	2023	[41]
	U-Net	2015	[42]
	UNet++	2018	[39]
Transformer	CrackFormer	2023	[43]
	EDTER	2022	[44]
	MA-Net	2020	[39]
	MMViT-Seg	2023	[45]
	SegFormer	2021	[46]
	Segmenter	2021	[47]
	Swin-Unet	2021	[48]
	TransUNet	2021	[49]
State-Space-Model	VM-UNet	2024	[50]
Diffusion	DiffusionEdge	2024	[51]
	MedSegDiff-V2	2023	[52]
Classical	Canny	1986	[53]
	CannyPF	2015	[54]
	ED	2011	[55]
	PhCon+GCanny	1999	[56]
	gPb+GCanny	2008	[57]

V. EDGE DETECTION APPROACHES

Table 2 lists the approaches collected, their publication year, and their code basis.

A. CLASSICAL APPROACHES

1) CANNY

A widely used classical edge detector is the algorithm proposed by Canny [58]. It computes the gradient magnitude and orientation at each pixel and applies nonmaximum suppression and subsequent hysteresis thresholding to determine the edge pixels.

2) CANNYPF

Lu et al. [59] proposed a parameter-free version of the Canny edge detector (CannyPF) that was included in the CannyLines line detection method. This method adaptively selects thresholds based on the distribution of gradient magnitude values of the image pixels.

3) GENERALIZED CANNY (GCANNY)

As other local image features may be more robust at detecting CTs than the gradient, we developed a generalized version of the Canny edge detector. It uses a general feature map to replace the gradient magnitude and an optional orientation map to replace the gradient orientation as its input. After nonmaximum suppression, it performs an additional step to connect small gaps in the resulting image. These connections result from checking for specific patterns of edge and nonedge pixels within a 4×4 window. Finally, it creates a binary edge map using either binary thresholding or hysteresis thresholding. The features phase congruency (PhCon) and globalized probability of boundary (gPb) [60] are used to determine the edge strength.

The idea of PhCon rests upon the local energy model proposed by Morrone and Owens [61], which postulates that image features become noticeable if the Fourier components are maximally in phase. In this article, we computed PhCon using an improved algorithm proposed by Kovesi [62], which applies log-Gabor wavelets rather than Fourier components.

The gPb, proposed by Maire et al. [60], combines different local image features to create a map of boundary probabilities and improves it by considering the global feature distribution afterward.

4) EDGE DRAWING (ED)

ED detects edges by determining the anchor points that are most likely edge pixels based on their gradient magnitude and then establishes links between them [63].

B. ML APPROACHES

1) CONVOLUTION-BASED APPROACHES

A widely used ML approach for image processing is using CNNs. We applied networks developed for edge detection and segmentation tasks to detect CTs.

a) Cascaded and high-resolution network (CHRNet)

CHRNet [64] detects edges using multiscale representations of the image while preserving the high resolution of the output map. Therefore, it concatenates the output of a convolutional block with the result of the previous block, uses batch normalization layers with an active affine parameter as an erosion operation for the homogeneous region in the image, and generates the output of the network by fusing the outputs of each block.

b) Lightweight dense convolutional neural network (LDC)

Several approaches also focus on reducing the network size, leading to faster prediction and lower hardware requirements. One is the LDC [65]. It integrates aspects of the advanced architectures DexiNed [66] and CATS [67] but is notably smaller due to some modifications. With approximately 0.7 million parameters, it requires less than 4% of parameters compared to DexiNed. Despite its reduced size, the LDC achieves competitive quality compared to more complex systems. As DexiNed, the LDC consists of layers structured in blocks. An ablation study comparing the LDC network with three or four blocks demonstrated that the LDC with three blocks has approximately 75% fewer parameters but delivers valuable results [65]. For CT detection, we trained the versions of LDC with either three (LDC-B3) or four blocks (LDC-B4).

c) Tiny and efficient edge detector (TEED)

TEED is another lightweight CNN developed for simplicity, efficiency, and generalization [68]. According to the authors, with only 58 k parameters, its size is less than 0.2% of the state-of-the-art models.

d) Deep category-aware semantic edge detection (CASENet)

CASENet [69] is a CNN architecture based on residual neural network (ResNet) [70] and a skip-layer architecture in which categorywise edge activations at the top convolution layer merge with the corresponding bottom layer features. It uses fixed fusion weights and bases the decision result primarily on high-level features.

e) Dynamic feature fusion (DFF)

DFF [71] bases upon CASENet but uses a feature extractor that normalizes the magnitude scales of multilevel features and adaptive fusion weights for different locations of multilevel feature maps, leading to finer edges.

f) U-Net

A task closely related to category-aware edge detection is semantic segmentation, which assigns objects in an image to different categories. U-Net is one of the most popular semantic segmentation networks. It is a fully CNN architecture consisting of a contracting and an expansive path [72].

In addition to the standard U-Net, we investigated smaller versions of U-Net architectures with fewer layers and channels for convolution. Due to their relatively simple architecture, small U-Nets are exciting candidates for hardware implementation. Our tiny version (*UNet-38k*) has three encoder and decoder layers (four in the standard U-Net), begins with six output channels for the first convolution layer (64 in the standard U-Net), and uses bilinear upsampling.

g) UNet++

UNet++ [73] is a more advanced version of U-Net, which uses deep supervision to segment medical images. The main

difference to U-Net is the use of nested and dense skip connections to reduce the semantic gap between the feature maps of the encoder and decoder.

h) Feature pyramid network (FPN)
The FPN constructs feature pyramids to detect objects at different scales at marginal extra cost [74]. Therefore, it employs a top-down architecture with lateral connections to build high-level semantic feature maps at all scales.

i) DeepLabV3+
Another semantic segmentation approach is DeepLabV3+ [75]. The proposed method combines spatial pyramid pooling with an encoder–decoder structure, resulting in refined segmentation results along object boundaries compared to the predecessor DeepLabV3 [76]. DeepLab approaches generally deploy dilated filters for "atrous convolutions" and atrous spatial pyramid pooling to robustly segment objects at multiple scales [77].

j) LinkNet
LinkNet [78] aims for efficient semantic segmentation by using an 18-layer ResNet [70] as a light encoder and bypassing the input of each encoder layer to the output of the corresponding decoder. Thus, the decoder requires fewer parameters because it shares the knowledge learned by the encoder in each layer.

2) TRANSFORMER-BASED APPROACHES
Another neural network type is the transformer, which has an underlying attention mechanism [79]. While initially designed to process sequential data like text, a variant for image processing named vision transformer (ViT) was developed [80]. However, some models do not directly incorporate a ViT but use special attention blocks, like MA-Net.

a) Multiscale attention network (MA-Net)
MA-Net [81] uses a self-attention mechanism to integrate local features with global dependencies adaptively. Therefore, positionwise and multiscale fusion attention blocks capture the spatial dependencies between pixels in an overall view and the channel dependencies between feature maps.

b) Segmenter
Segmenter [82] uses a ViT as the encoder and employs two decoder variants for semantic segmentation. The decoder is either an ordinary linear layer or a novel mask transformer, which is a transformer encoder with multiple layers. The results presented later in this article use the second option because it performs better on our validation dataset.

c) SegFormer
SegFormer [83] uses a modified ViT as an encoder with a hierarchical structure without positional embedding. The decoder is a lightweight multilayer perceptron that aggregates information from different spatial resolution features arising from the hierarchical structure to combine local and global attention.

d) Edge detection TransformER (EDTER)
EDTER [84] combines a transformer-based encoder and convolution-based decoder to extract precise sharp object boundaries and meaningful edges. It simultaneously exploits the complete contextual information of the image and detailed local cues by operating in two stages. The first stage (EDTER-Global) uses the encoder part of a ViT with coarse image patches, and the second stage uses a modified local ViT encoder with finer image patches. Both stages use a bidirectional multilevel aggregation decoder, and a feature fusion module combines their results before being fed into a final decision head.

e) Mini-mobile vision transformer for segmentation (MMViT-Seg)
MMViT-Seg [85] is a lightweight model in which the encoder subnetwork is a two-path design that effectively captures the global dependence of image features and low-layer spatial details. Therefore, it uses convolutional and MobileViT blocks and a multiquery attention module to fuse multiscale features from different levels in the decoder subnetwork.

f) CrackFormer
CrackFormer [86] combines a SegNet-like [87] encoder–decoder architecture with self-attention. It replaces all convolutional layers (except the first and last) with self-attention blocks and implements a feature fusion module that combines the features from each encoder–decoder stage using self-attention. Each self-attention block consists of two convolution layers, followed by a batch norm and a rectified linear unit activation function, with a self-attention layer in between.

g) TransUNet
TransUNet [88] combines the general concepts of the U-Net architecture with a transformer. Although U-Net is effective in local feature detection, its ability to model long-range dependencies is weak. In contrast, transformers have an innate global self-attention mechanism but only limited localization capabilities due to insufficient low-level details. TransUNet combines these two architectures using a CNN-Transformer-Hybrid for the encoder and convolutional layers in the decoder. Thus, each of one's strengths overcomes the weaknesses of the other.

h) Swin-Unet
Swin-Unet [89] also combines the properties of the traditional U-Net's U-shaped architecture and skip connections with a pure transformer encoder architecture. The proposed method uses Swin transformers [90], a variant of ViTs [80], for the encoder and a Swin-transformer-based decoder with patch-expanding layers to upsample features during the expansive path.

TABLE 3. Training Hyperparameters of the Collected ML Approaches, Which are Sorted by Detector Name

Detector Name	Training Settings		Optimizer Settings			Scheduler Settings
	Epochs	Batch Size	Name	Learning Rate	Weight Decay	Name
CASENet	5	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
CHRNet	5	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
CrackFormer	5	64	AdamW	$1.0{\cdot}10^{-1}$	$1.0{\cdot}10^{-3}$	OneCycleLR
DeepLabV3+	5	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
DFF	5	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
DiffusionEdge	–	–	–	–	–	–
EDTER	1	16	AdamW	$5.0{\cdot}10^{-4}$	$3.0{\cdot}10^{-3}$	OneCycleLR
EDTER-Global	1	16	AdamW	$5.0{\cdot}10^{-4}$	$3.0{\cdot}10^{-1}$	OneCycleLR
FPN	4	64	AdamW	$1.0{\cdot}10^{-1}$	$1.0{\cdot}10^{-3}$	OneCycleLR
LDC-B3	5	64	AdamW	$1.0{\cdot}10^{-2}$	$1.0{\cdot}10^{-3}$	OneCycleLR
LDC-B4	5	64	AdamW	$1.0{\cdot}10^{-2}$	$1.0{\cdot}10^{-3}$	OneCycleLR
LinkNet	10	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
MA-Net	5	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
MedSegDiff-V2	1	32	AdamW	$1.0{\cdot}10^{-4}$	0	LinearLR
MMViT-Seg	5	16	Adam	$1.0{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	OneCycleLR
SegFormer	1	256	AdamW	$3.0{\cdot}10^{-3}$	$3.0{\cdot}10^{-1}$	OneCycleLR
Segmenter	5	32	AdamW	$3.0{\cdot}10^{-3}$	$3.0{\cdot}10^{-3}$	OneCycleLR
Swin-Unet	10	256	AdamW	$1.0{\cdot}10^{-3}$	$1.0{\cdot}10^{-2}$	OneCycleLR
TEED	5	8	AdamW	$1.0{\cdot}10^{-2}$	$2.0{\cdot}10^{-3}$	OneCycleLR
TransUNet	4	64	AdamW	$1.0{\cdot}10^{-1}$	$1.0{\cdot}10^{-3}$	OneCycleLR
U-Net	5	64	AdamW	$2.0{\cdot}10^{-1}$	$1.0{\cdot}10^{-4}$	OneCycleLR
UNet-38k	5	64	AdamW	$2.0{\cdot}10^{-1}$	$1.0{\cdot}10^{-4}$	OneCycleLR
UNet++	5	64	AdamW	$1.0{\cdot}10^{-1}$	$1.0{\cdot}10^{-3}$	OneCycleLR
VM-UNet	5	32	AdamW	$1.0{\cdot}10^{-3}$	$1.0{\cdot}10^{-2}$	OneCycleLR

The optimizers and schedulers refer to the implementations in the torch.optim package [96]. For DiffusionEdge, we did not supply parameters because we used the original parameters.

3) STATE-SPACE-MODEL-BASED APPROACHES

State-space models are an alternative to transformers for processing long sequences [91].

a) Vision Mamba UNet (VM-UNet)

VM-UNet [92] uses such a state-space model. Like Swin-Unet, it incorporates the architectural benefits of U-Net and nonconvolutional layers. Instead of building upon ViTs or, more specifically, Swin transformers, it builds upon visual state-space models (VMambas), a recent architecture that combines a global receptive field with linear complexity.

4) DIFFUSION-BASED APPROACHES

Diffusion models learn forward and reverse diffusion and are often used for image denoising, image inpainting, and image generation. Therefore, forward diffusion usually involves adding Gaussian noise to the original image, and reverse diffusion inverts that diffusion process, thereby reconstructing the original image.

a) Diffusion probabilistic model for crisp edge detection (DiffusionEdge)

DiffusionEdge [93] is a diffusion probabilistic model (DPM) for the general task of edge detection. The authors claimed that they avoided expensive computational resources and retained the final performance by applying a DPM in the latent space. Thus, they enabled the classic cross-entropy loss to optimize parameters in the latent space. In addition, they adapted a decoupled architecture to speed up the denoising process and proposed an adaptive Fourier filter to adjust the latent features of specific frequencies. This combination should result in very accurate and crisp edge maps.

b) Medical image segmentation with diffusion probabilistic model (MedSegDiff)

MedSegDiff [94] is a DPM for general medical image segmentation. It uses a modified U-Net with conditional encoding and a feature frequency parser in the reverse diffusion stage. The dynamic conditional strategy enables stepwise attention, and the feature frequency parser eliminates the high-frequency noise introduced by the former.

MedSegDiff-V2 [95] improves MedSegDiff [94] by introducing ViT mechanisms into the DPM. It uses convolutional U-Nets as feature extractors but combines features using a transformer.

VI. TRAINING

Where applicable, we trained the networks using the original publication's optimizers, schedulers, loss functions, and hyperparameters. In addition, we trained the networks using a combination of the `AdamW` optimizer and the `OneCycleLR` scheduler (implemented in `torch.optim` [96]), which is the de facto state-of-the-art superconvergence method proposed in [97]. In this case, the loss function consists of a combination of `BCEWithLogitsLoss` (implemented in `torch.nn` [96]) and DICE loss. Subsequently, we evaluated the training results of the networks on the validation dataset described in Section III and selected the best training checkpoint for final comparison. Table 3 provides an overview of the hyperparameters used to train the final checkpoint for comparison. Classical approaches, except for gPb+GCanny, were optimized using differential evolution implemented in Scipy. For gPb+GCanny, we used a simple grid search because only one parameter was optimized.

TABLE 4. Size of the Neural Networks and Their Inference Time, FLOPs, and MACs

Detector		Parameters	Inference Time [ms]		GFLOPs	GMACs
Name	Category	[million]	Batch Size 1	Batch Size 64	[per image]	[per image]
UNet-38k	Convolution	0.038	0.956	3.295	0.077	0.038
TEED	Convolution	0.059	1.118	4.070	0.270	0.134
LDC-B3	Convolution	0.156	1.269	5.155	0.559	0.276
LDC-B4	Convolution	0.674	2.247	6.813	0.954	0.472
MMViT-Seg	Transformer	1.013	23.810	66.641	0.721	0.354
CHRNet	Convolution	1.450	2.545	19.167	4.085	2.037
SegFormer	Transformer	4.446	5.157	15.429	5.897	2.946
CrackFormer	Transformer	4.961	21.825	123.081	6.323	3.080
Segmenter	Transformer	6.455	4.971	3775.265	14.517	7.235
LinkNet	Convolution	11.658	2.190	6.360	1.420	0.706
U-Net	Convolution	17.262	1.747	36.417	11.237	5.612
CASENet	Convolution	21.793	2.409	23.293	16.220	8.100
DFF	Convolution	21.799	2.683	24.545	16.226	8.103
FPN	Convolution	23.149	3.514	20.702	3.805	1.899
UNet++	Convolution	26.072	4.637	25.619	10.298	5.141
Swin-Unet	Transformer	27.154	5.994	32.387	2.160	1.074
VM-UNet	State-Space-Model	27.424	7.432	99.503	–	–
MA-Net	Transformer	31.777	4.337	14.145	4.628	2.309
DeepLabV3+	Convolution	45.663	6.779	22.873	7.834	3.906
MedSegDiff-V2	Diffusion	95.723	1726.027	41106.613	–	–
TransUNet	Transformer	105.153	13.828	74.393	21.016	10.489
DiffusionEdge	Diffusion	297.583	357.617	1246.634	–	–
EDTER-Global	Transformer	322.386	16.215	12851.895	254.208	127.008
EDTER	Transformer	416.964	37.759	25147.160	373.765	186.685

We measured the inference times on an NVIDIA RTX A5000 but could not calculate FLOPs and MACs for all networks.

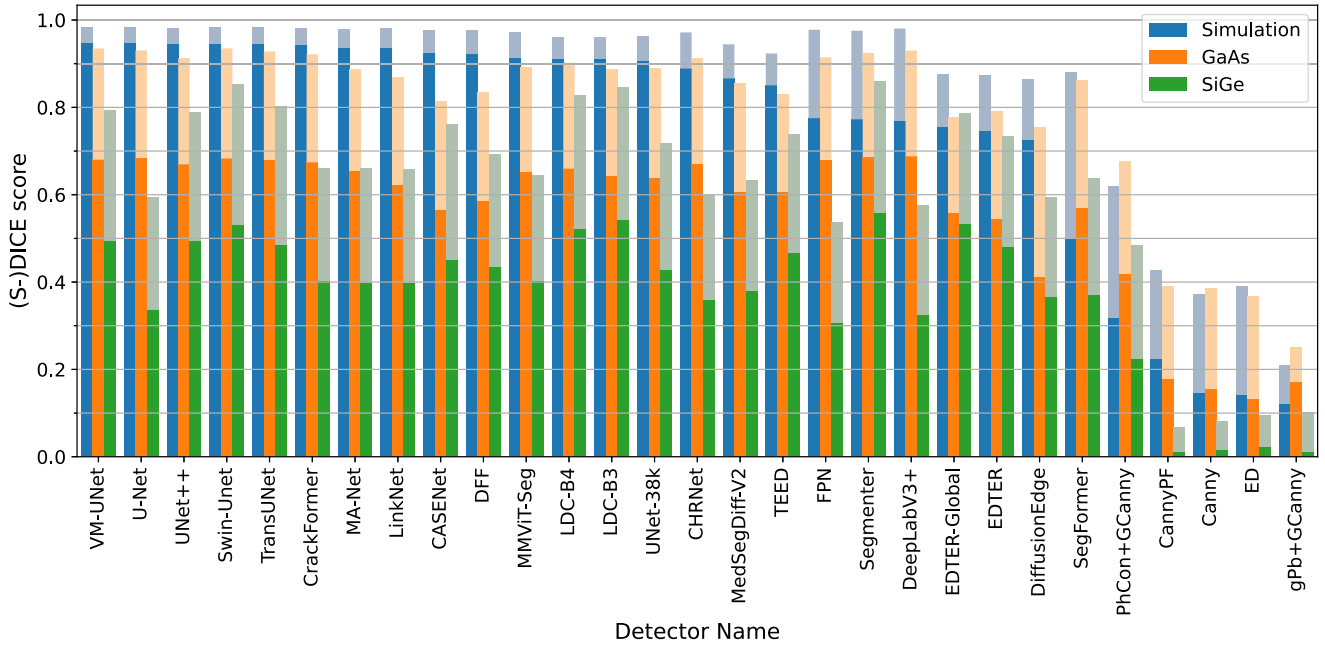

FIGURE 3. Bar plot visualizing the results from Table 5. For each detector, the solid portion of the bar represents the DICE score, while the semitransparent extension indicates the corresponding S-DICE score. The detectors are arranged in accordance with their DICE score on simulation data, which is consistent with the ordering in Table 5.

VII. EVALUATION

We evaluated the collected approaches from Section V on the test sets described in Section III using the methods from Section IV.

Table 4 lists the number of parameters, the inference times, the FLOPs, and the MACs of all ML models. There are only four candidates with less than one million parameters. Of those, especially UNet-38k and TEED are very small, with 38 041 and 58 622 parameters, respectively. In addition, both are primarily based on convolutions and have a relatively simple network structure, which we consider beneficial for hardware implementation. The inference times for single images (batch size 1) significantly differed between the approach categories. The fastest approaches were from the convolution-based and the slowest from the diffusion-based category. In particular, DiffusionEdge and MedSegDiff-V2 are highly time-consuming and, therefore, are less applicable to scalable tuning solutions. Notably, tiny networks do

IEEE Transactions on Quantum Engineering

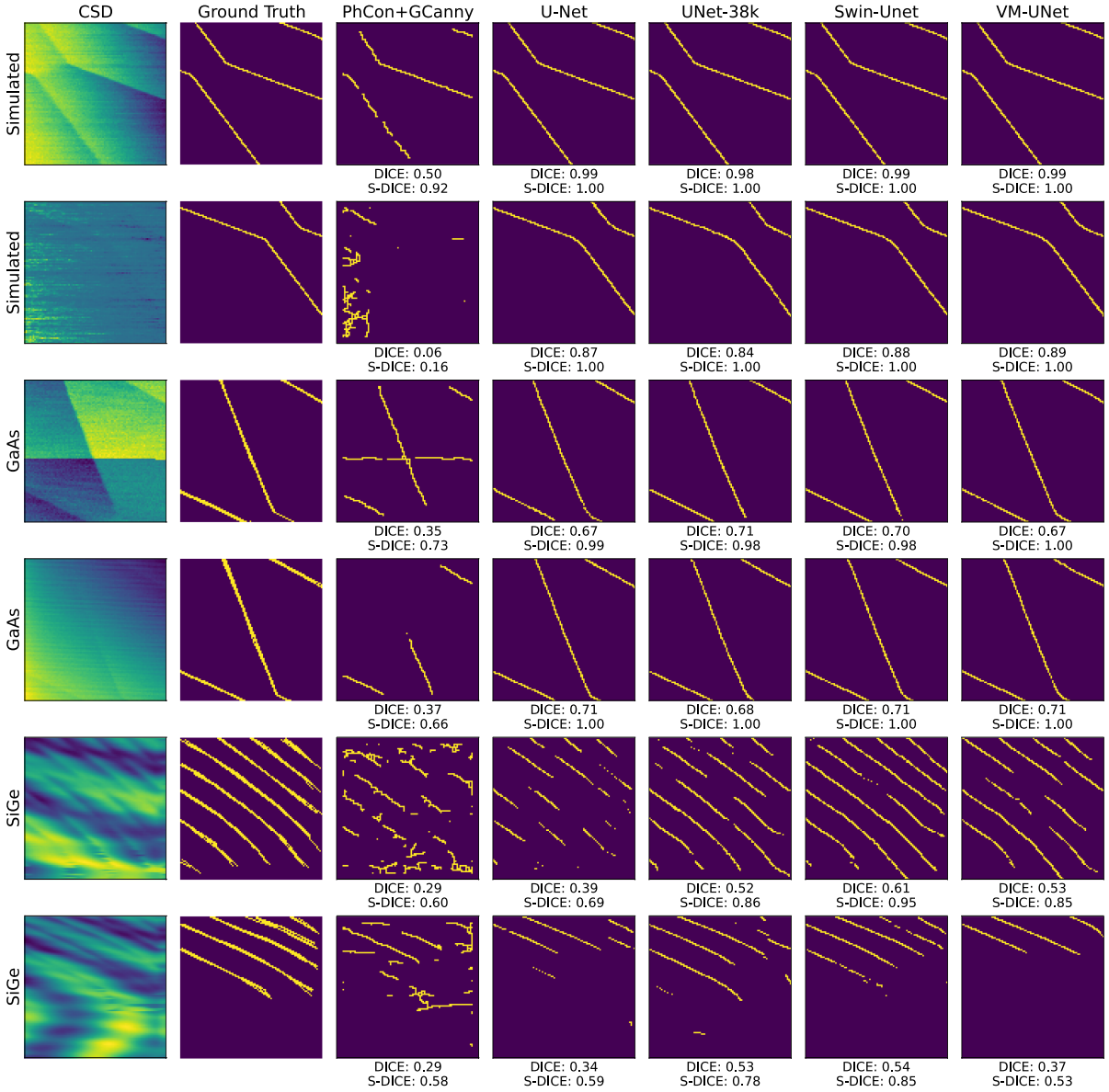

FIGURE 4. **Exemplary predictions on the test datasets along with their corresponding DICE and S-DICE scores. The examples were chosen to encompass a broad spectrum of cases. The top two rows present simulated data. The upper row corresponds to a sensor dot positioned on the flank of a Coulomb peak, within the regime of optimal charge sensitivity. The lower row illustrates a case where the sensor dot resides in a less favorable operating regime. The GaAs measurements display two illustrative cases: the upper panel features pronounced CTs superimposed with strong random telegraph noise, which manifests as a spurious linear feature that must not be misinterpreted as a CT; the lower panel exhibits only faintly discernible CTs. The SiGe data show single QD features with variations in both the angular orientation and sharpness of the CTs.**

not fully utilize the GPU and enable even more predictions when multiple such models run in parallel. In a scalable fully automated tuning setup, a CT detector could analyze CSDs of multiple different qubits at the same time. Therefore, we recorded inference times for a batch of 64 images. The given numbers of FLOPs and MACs depend on the number of parameters and the architecture. From the results in Table 4, we see that small CNNs require fewer FLOPs and MACs than other architectures or convolutional networks with more parameters.

Table 5 summarizes the achieved metrics, sorted by the DICE score for simulated data and the S-DICE score for experimental data. In addition, Fig. 3 provides a visualization of the metrics, sorted by the DICE score achieved on simulated data. The ML approaches have a clear advantage over classical methods for all test datasets. Regarding the simulated data, U-Net-based architectures performed best (top 5), but, in general, many approaches achieved convincing results. Remarkably, the small UNet-38k scored a DICE score above 0.9. In the group of classical methods, only Ph-

TABLE 5. Metrics Calculated for All Detectors on the Three Test Datasets

Simulated Data			GaAs Data			SiGe Data		
Detector Name	**DICE**	**S-DICE**	**Detector Name**	**DICE**	**S-DICE**	**Detector Name**	**DICE**	**S-DICE**
VM-UNet	0.948611	0.984723	Swin-Unet	0.682991	0.935084	Segmenter	0.560009	0.860184
U-Net	0.947980	0.983640	VM-UNet	0.680310	0.934989	Swin-Unet	0.529968	0.852079
UNet++	0.946648	0.981668	U-Net	0.684296	0.930579	LDC-B3	0.542285	0.846194
Swin-Unet	0.946016	0.984461	DeepLabV3+	0.687865	0.930045	LDC-B4	0.521855	0.827330
TransUNet	0.945229	0.982025	TransUNet	0.679336	0.928166	TransUNet	0.485132	0.802789
CrackFormer	0.942269	0.980714	Segmenter	0.686771	0.924734	VM-UNet	0.494951	0.793315
MA-Net	0.937779	0.979485	CrackFormer	0.674050	0.921575	UNet++	0.494466	0.789553
LinkNet	0.935520	0.980958	FPN	0.680005	0.914432	EDTER-Global	0.533004	0.785397
CASENet	0.925680	0.977085	CHRNet	0.670291	0.912873	CASENet	0.451935	0.761143
DFF	0.923153	0.976307	UNet++	0.669483	0.912810	TEED	0.465905	0.737797
MMViT-Seg	0.912982	0.972229	LDC-B4	0.659616	0.902324	EDTER	0.481658	0.734945
LDC-B4	0.912331	0.961914	MMViT-Seg	0.653716	0.892694	UNet-38 k	0.426954	0.717193
LDC-B3	0.912062	0.961828	UNet-38 k	0.640319	0.890312	DFF	0.434880	0.692505
UNet-38 k	0.907693	0.963606	LDC-B3	0.643522	0.888084	CrackFormer	0.404251	0.660857
CHRNet	0.888840	0.971695	MA-Net	0.654963	0.886866	MA-Net	0.398269	0.660703
MedSegDiff-V2	0.866172	0.944311	LinkNet	0.623341	0.870476	LinkNet	0.398087	0.658808
TEED	0.850516	0.923503	SegFormer	0.571092	0.862221	MMViT-Seg	0.402314	0.644663
FPN	0.775319	0.977872	MedSegDiff-V2	0.607878	0.855302	SegFormer	0.370511	0.637816
Segmenter	0.772588	0.975522	DFF	0.586459	0.834489	MedSegDiff-V2	0.380844	0.633545
DeepLabV3+	0.768866	0.980459	TEED	0.606017	0.829216	CHRNet	0.358583	0.598633
EDTER-Global	0.755435	0.875510	CASENet	0.565923	0.815075	DiffusionEdge	0.367199	0.594880
EDTER	0.747603	0.874246	EDTER	0.544918	0.791837	U-Net	0.335262	0.593540
DiffusionEdge	0.726628	0.864524	EDTER-Global	0.559983	0.777607	DeepLabV3+	0.326072	0.576167
SegFormer	0.500667	0.880487	DiffusionEdge	0.411255	0.753454	FPN	0.306444	0.535457
PhCon+GCanny	0.317213	0.619396	PhCon+GCanny	0.419984	0.677735	PhCon+GCanny	0.223991	0.484963
CannyPF	0.224394	0.427694	CannyPF	0.178615	0.390894	gPb+GCanny	0.012346	0.102248
Canny	0.145899	0.371413	Canny	0.156118	0.385560	ED	0.022423	0.093937
ED	0.141768	0.390697	ED	0.132090	0.367629	Canny	0.016921	0.081467
gPb+GCanny	0.120203	0.209391	gPb+GCanny	0.170843	0.249936	CannyPF	0.012614	0.068481

We sorted the simulated data results by the mean DICE score and the experimental data results (GaAs and SiGe) by the mean S-DICE score, because the inaccuracies in the manual labels limit the significance of the DICE score. We calculate the S-DICE score with an accepted segmentation deviation of two pixels.

Con+GCanny was barely usable, scoring an S-DICE score of 0.62. Notably, the global part of EDTER outperformed the entire algorithm. Due to the necessary adaptation of the global patch size (from 16 to 8) to our small image resolution (96×96 pixels), the cues of the local stage (patch size of 4) became too similar to the global ones, which resulted in loss of global information. The results obtained on the GaAs dataset provide a good measure for the approaches' generalization ability from simulated to experimental data. Our simulated data originate from GaAs parameter ranges; thus, we expect similar scores for promising methods. Because of the uncertainty of manual labels, we used the S-DICE score for comparison. Again, U-Net-based architectures scored the best (four out of the top five). DeepLabV3+ is unable to perform pixel-perfect segmentations, as its S-DICE score is considerably better than its DICE score on the simulated data. The proposed UNet-38k anew achieved good results (S-DICE score above 0.89). Again, only PhCon+GCanny achieved usable results for the classical approaches, which were even better than those for the simulated data.

With the SiGe dataset, we analyzed the edge detection capabilities using an entirely different qubit sample architecture and material. As expected, this dataset's mean S-DICE score is lower, because of the fairly different feature properties described in Section III. Surprisingly, U-Net-based architectures no longer perform best; in particular, the original U-Net architecture is deteriorating. This indicates overfitting to double-dot features that are not present in the SiGe data. At the same time, Segmenter and LDC performed more robust than their competitors. Again, we observed competitive results for the UNet-38k (S-DICE score above 0.71), indicating that overfitting was not present, presumably because of its reduced capacity to learn more complex structures. All the classical approaches achieved poor results on the SiGe data because of blurrier transitions.

To summarize, we consider many approaches to have good overall analysis capabilities. Among the classical approaches, PhCon+GCanny is the only approach that delivered valuable results. The best convolution-based approach appears to be U-Net, which performed poorly on the SiGe data.[9] Surprisingly robust and efficient, the scaled-down UNet-38k approach did not cause problems with the SiGe data. Swin-Unet emerged as the best approach among the analyzed transformer models. The state-space model approach VM-Unet also detected CTs robustly. Only diffusion networks are unsuitable for our application because they are too complex for energy-efficient hardware implementation and demonstrated poor metrics in our analysis. Fig. 4 shows exemplary predictions of the best representatives of the respective approach categories. Although PhCon+GCanny provided useful predictions for some images, it is very susceptible to distortions and the properties of the CT features. The ML approaches did not noticeably differ in their simulated and GaAs data predictions. Apparent differences were only visible when generalizing to the SiGe data. For example,

[9]The poor performance on SiGe data is explained with overfitting, which can be avoided as shown with UNet-38k.

U-Net demonstrated significant gaps in the predicted edges, whereas UNet-38k performed considerably better, and Swin-Unet worked the best.

VIII. CONCLUSION

In this study, we evaluated various classical and ML approaches for detecting CTs in CSDs using simulated data from the SimCATS framework. Our focus was on the search for suitable approach categories for robust detection with potential for future qubit-near hardware implementation.

We subdivided the ML-based approaches into convolution-, transformer-, state-space-model-, and diffusion-based approaches to analyze the abilities of different architectures. The analysis of the detection metrics has shown that ML-based approaches are far superior to their classical competitors. All investigated ML architecture categories featured valuable candidates except for the diffusion-based approaches. The results demonstrate that approaches trained on our simulated dataset can generalize to experimental data.

We recommend convolution-based architectures for hardware implementation due to their low complexity and convincing detection results. Furthermore, we emphasize the possibility of creating smaller versions of networks that still have sufficient detection ability, as demonstrated with UNet-38k.

For experiments unconstrained by computational limitations or strict energy efficiency requirements, we recommend Swin-Unet, which consistently achieved top-tier performance across all evaluated datasets.

Future research should investigate further tiny versions of convolution-based networks, e.g., using neural architecture search [98], [99] and hyperparameter optimization [100] techniques. In addition to network size, a reduced data representation rather than 32-bit floating-point numbers is preferable for energy-optimized hardware implementations. The test dataset is available for benchmarking [101]. Furthermore, dedicated analysis hardware must prove its applicability via in-field tests at cryogenic temperatures. In this context, specialized hardware components like memristor crossbar arrays should also be considered [18].

REFERENCES

[1] J. M. Elzerman et al., "Few-electron quantum dot circuit with integrated charge read out," *Phys. Rev. B*, vol. 67, no. 16, Art. no. 161308, Apr. 2003, doi: 10.1103/PhysRevB.67.161308.

[2] L. Geck, A. Kruth, H. Bluhm, S. van Waasen, and S. Heinen, "Control electronics for semiconductor spin qubits," *Quantum Sci. Technol.*, vol. 5, 2019, Art. no. 015004, doi: 10.1088/2058-9565/ab5e07.

[3] A. Ruffino, T.-Y. Yang, J. Michniewicz, Y. Peng, E. Charbon, and M. F. Gonzalez-Zalba, "A cryo-CMOS chip that integrates silicon quantum dots and multiplexed dispersive readout electronics," *Nat. Electron.*, vol. 5, no. 1, pp. 53–59, Jan. 2022, doi: 10.1038/s41928-021-00687-6.

[4] F. Hader, S. Fleitmann, J. Vogelbruch, L. Geck, and S. V. Waasen, "Simulation of charge stability diagrams for automated tuning solutions (SimCATS)," *IEEE Trans. Quantum Eng.*, vol. 5, 2024, Art. no. 5500414, doi: 10.1109/TQE.2024.3445967.

[5] J. P. Zwolak and J. M. Taylor, "Colloquium: Advances in automation of quantum dot devices control," *Rev. Modern Phys.*, vol. 95, no. 1, 2023, Art. no. 011006, doi: 10.1103/RevModPhys.95.011006.

[6] J. Darulová et al., "Autonomous tuning and charge state detection of gate defined quantum dots," 2019, *arXiv:1911.10709*, doi: 10.48550/arXiv.1911.10709.

[7] B. Severin et al., "Cross-architecture tuning of silicon and SiGe-based quantum devices using machine learning," *Sci. Rep.*, vol. 14, no. 1, Jul. 2024, Art. no. 17281, doi: 10.1038/s41598-024-67787-z.

[8] S. S. Kalantre et al., "Machine learning techniques for state recognition and auto-tuning in quantum dots," *npj Quantum Inf.*, vol. 5, no. 1, 2019, Art. no. 6, doi: 10.1038/s41534-018-0118-7.

[9] J. P. Zwolak et al., "Autotuning of double dot devices in situ with machine learning," *Phys. Rev. Appl.*, vol. 13, 2020, Art. no. 034075, doi: 10.1103/PhysRevApplied.13.034075.

[10] J. Ziegler et al., "Toward robust autotuning of noisy quantum dot devices," 2021, *arXiv:2108.00043*, doi: 10.48550/arXiv.2108.00043.

[11] Y. Muto et al., "Visual explanations of machine learning model estimating charge states in quantum dots," *APL Mach. Learn.*, vol. 2, 2024, Art. no. 026110, doi: 10.1063/5.0193621.

[12] H. Liu et al., "An automated approach for consecutive tuning of quantum dot arrays," *Appl. Phys. Lett.*, vol. 121, no. 8, 2022, Art. no. 084002, doi: 10.1063/5.0111128.

[13] T. A. Baart, P. T. Eendebak, C. Reichl, W. Wegscheider, and L. M. K. Vandersypen, "Computer-automated tuning of semiconductor double quantum dots into the single-electron regime," *Appl. Phys. Lett.*, vol. 108, no. 21, May 2016, Art. no. 213104, doi: 10.1063/1.4952624.

[14] M. Lapointe-Major et al., "Algorithm for automated tuning of a quantum dot into the single-electron regime," *Phys. Rev. B*, vol. 102, no. 8, Aug. 2020, Art. no. 085301, doi: 10.1103/PhysRevB.102.085301.

[15] H. Moon et al., "Machine learning enables completely automatic tuning of a quantum device faster than human experts," *Nat. Commun.*, vol. 11, no. 1, Art. no. 4161, 2020, doi: 10.1038/s41467-020-17835-9.

[16] B. van Straaten et al., "All RF-based tuning algorithm for quantum devices using machine learning," 2022, *arXiv:2211.04504*, doi: 10.48550/arXiv.2211.04504.

[17] R. Durrer et al., "Automated tuning of double quantum dots into specific charge states using neural networks," *Phys. Rev. Appl.*, vol. 13, no. 5, Art. no. 054019, May 2020, doi: 10.1103/PhysRevApplied.13.054019.

[18] S. Czischek et al., "Miniaturizing neural networks for charge state autotuning in quantum dots," *Mach. Learn.: Sci. Technol.*, vol. 3, no. 1, Mar. 2022, Art. no. 015001, doi: 10.1088/2632-2153/ac34db.

[19] J. Ziegler, F. Luthi, M. Ramsey, F. Borjans, G. Zheng, and J. P. Zwolak, "Tuning arrays with rays: Physics-informed tuning of quantum dot charge states," *Phys. Rev. Appl.*, vol. 20, no. 3, Sep. 2023, Art. no. 034067, doi: 10.1103/PhysRevApplied.20.034067.

[20] J. P. Zwolak et al., "Ray-based framework for state identification in quantum dot devices," *PRX Quantum*, vol. 2, no. 2, 2021, Art. no. 020335, doi: 10.1103/PRXQuantum.2.020335.

[21] J. P. Zwolak, S. S. Kalantre, X. Wu, S. Ragole, and J. M. Taylor, "QFlow lite dataset: A machine-learning approach to the charge states in quantum dot experiments," *PLOS One*, vol. 13, no. 10, 2018, Art. no. e0205844, doi: 10.1371/journal.pone.0205844.

[22] C. J. van Diepen et al., "Automated tuning of inter-dot tunnel coupling in double quantum dots," *Appl. Phys. Lett.*, vol. 113, no. 3, 2018, Art. no. 033101, doi: 10.1063/1.5031034.

[23] A. R. Mills et al., "Computer-automated tuning procedures for semiconductor quantum dot arrays," *Appl. Phys. Lett.*, vol. 115, no. 11, 2019, Art. no. 113501, doi: 10.1063/1.5121444.

[24] C. Monical, P. Lewis, A. Agron, K. Larson, and A. Mounce, "Image processing algorithms for tuning quantum devices and nitrogen-vacancy imaging," Sandia Nat. Lab., USDOE Nat. Nuclear Secur. Admin., Washington, DC, USA, Tech. Rep. SAND-2019-11786, 2019, doi: 10.2172/1662017.

[25] J. D. Teske et al., "A machine learning approach for automated fine-tuning of semiconductor spin qubits," *Appl. Phys. Lett.*, vol. 114, no. 13, 2019, Art. no. 133102, doi: 10.1063/1.5088412.

[26] N. M. van Esbroeck et al., "Quantum device fine-tuning using unsupervised embedding learning," *New J. Phys.*, vol. 22, no. 9, 2020, Art. no. 095003, doi: 10.1088/1367-2630/abb64c.

[27] J. Schuff et al., "Fully autonomous tuning of a spin qubit," 2024, *arXiv:2402.03931*, doi: 10.48550/arXiv.2402.03931.

[28] D. T. Lennon et al., "Efficiently measuring a quantum device using machine learning," *npj Quantum Inf.*, vol. 5, no. 1, 2019, Art. no. 79, doi: 10.1038/s41534-019-0193-4.

[29] V. Nguyen et al., "Deep reinforcement learning for efficient measurement of quantum devices," *npj Quantum Inf.*, vol. 7, no. 1, pp. 1–9, 2021, doi: 10.1038/s41534-021-00434-x.

[30] Y. Matsumoto, T. Fujita, A. Ludwig, A. D. Wieck, K. Komatani, and A. Oiwa, "Noise-robust classification of single-shot electron spin readouts using a deep neural network," *npj Quantum Inf.*, vol. 7, no. 1, pp. 1–7, 2021, doi: 10.1038/s41534-021-00470-7.

[31] F. Hader, "SimCATS-datasets," Dec. 2023. [Online]. Available: https://github.com/f-hader/SimCATS-Datasets

[32] C. Volk et al., "Loading a quantum-dot based "qubyte" register," *npj Quantum Inf.*, vol. 5, no. 1, Apr. 2019, Art. no. 29, doi: 10.1038/s41534-019-0146-y.

[33] F. Hader, "SimCATS," Dec. 2023. [Online]. Available: https://github.com/f-hader/SimCATS

[34] T. Struck et al., "Spin-EPR-pair separation by conveyor-mode single electron shuttling in Si/SiGe," *Nat. Commun.*, vol. 15, no. 1, Feb. 2024, Art. no. 1325, doi: 10.1038/s41467-024-45583-7.

[35] S. Seidlitz et al., "Robust deep learning-based semantic organ segmentation in hyperspectral images," *Med. Image Anal.*, vol. 80, Aug. 2022, Art. no. 102488, doi: 10.1016/j.media.2022.102488.

[36] X. Ye, "Calflops: A FLOPs and Params calculate tool for neural networks in PyTorch framework," 2023. [Online]. Available: https://github.com/MrYxJ/calculate-flops.pytorch

[37] "DFF and CASENet source code." Accessed: Feb. 26, 2024. [Online]. Available: https://github.com/Lavender105/DFF

[38] "CHRNet source code." Accessed: Mar. 22, 2024. [Online]. Available: https://github.com/elharroussomar/Refined-Edge-Detection-With-Cascaded-and-High-Resolution-Convolutional-Network

[39] P. Iakubovskii, "Segmentation models PyTorch," 2019. [Online]. Available: https://github.com/qubvel/segmentation_models.pytorch

[40] "LDC source code." Accessed: Mar. 25, 2024. [Online]. Available: https://github.com/xavysp/LDC

[41] "TEED source code." Accessed: May 16, 2024. [Online]. Available: https://github.com/xavysp/TEED

[42] "U-net source code." Accessed: Jan. 25, 2024. [Online]. Available: https://github.com/milesial/Pytorch-UNet

[43] "CrackFormer source code." Accessed: Mar. 22, 2024. [Online]. Available: https://github.com/LouisNUST/CrackFormer-II

[44] "EDTER source code." Accessed: Jan. 20, 2024. [Online]. Available: https://github.com/MengyangPu/EDTER

[45] "MMViT-Seg source code." Accessed: Apr. 17, 2024. [Online]. Available: https://github.com/yywbkn/MMViT-Segmentation

[46] "SegFormer source code." Accessed: Jan. 22, 2024. [Online]. Available: https://github.com/NVlabs/SegFormer

[47] "Segmenter source code." Accessed: Feb. 26, 2024. [Online]. Available: https://github.com/rstrudel/segmenter

[48] "Swin-Unet source code." Accessed: Feb. 15, 2024. [Online]. Available: https://github.com/HuCaoFighting/Swin-Unet

[49] "TransUNet source code." Accessed: Feb. 22, 2024. [Online]. Available: https://github.com/Beckschen/TransUNet

[50] "VM-UNet source code." Accessed: Feb. 13, 2024. [Online]. Available: https://github.com/JCruan519/VM-UNet

[51] "DiffusionEdge source code." Accessed: Feb. 26, 2024. [Online]. Available: https://github.com/GuHuangAI/DiffusionEdge

[52] "MedSegDiff-V2 source code." Accessed: Apr. 22, 2024. [Online]. Available: https://github.com/MedicineToken/MedSegDiff

[53] "Canny edge detector software interface." Accessed: Jul. 29, 2024. [Online]. Available: https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.canny

[54] "CannyPF software interface." Accessed: Jul. 29, 2024. [Online]. Available: https://cvrs.whu.edu.cn/cannylines/#ct1

[55] "Edge drawing software interface." Accessed: Jul. 29, 2024. [Online]. Available: https://github.com/shaojunluo/EDLinePython

[56] "Phase congruency software interface." Accessed: Jul. 29, 2024. [Online]. Available: https://github.com/peterkovesi/ImagePhaseCongruency.jl

[57] "gPb software interface." Accessed: Jul. 29, 2024. [Online]. Available: https://github.com/HiDiYANG/gPb-GSoC

[58] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI- 8, pp. 679–698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[59] X. Lu, J. Yao, K. Li, and L. Li, "Cannylines: A parameter-free line segment detector," in *Proc. IEEE Int. Conf. Image Process.*, 2015, pp. 507–511, doi: 10.1109/ICIP.2015.7350850.

[60] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8, doi: 10.1109/CVPR.2008.4587420.

[61] M. Morrone and R. Owens, "Feature detection from local energy," *Pattern Recognit. Lett.*, vol. 6, no. 5, pp. 303–313, 1987, doi: 10.1016/0167-8655(87)90013-4.

[62] P. Kovesi, "Image features from phase congruency," *Videre: J. Comput. Vis. Res.*, vol. 1, no. 3, pp. 1–26, 1999.

[63] C. Akinlar and C. Topal, "EDLines: A real-time line segment detector with a false detection control," *Pattern Recognit. Lett.*, vol. 32, pp. 1633–164, 2011, doi: 10.1016/j.patrec.2011.06.001.

[64] O. Elharrouss, Y. Hmamouche, A. K. Idrissi, B. El Khamlichi, and A. E. Fallah-Seghrouchni, "Refined edge detection with cascaded and high-resolution convolutional network," *Pattern Recognit.*, vol. 138, 2023, Art. no. 109361, doi: 10.1016/j.patcog.2023.109361.

[65] X. Soria, G. Pomboza-Junez, and A. D. Sappa, "LDC: Lightweight dense CNN for edge detection," *IEEE Access*, vol. 10, pp. 68281–68290, 2022, doi: 10.1109/ACCESS.2022.3186344.

[66] X. Soria, A. Sappa, P. Humanante, and A. Akbarinia, "Dense extreme inception network for edge detection," *Pattern Recognit.*, vol. 139, 2023, Art. no. 109461, doi: 10.1016/j.patcog.2023.109461.

[67] L. Huan, N. Xue, X. Zheng, W. He, J. Gong, and G.-S. Xia, "Unmixing convolutional features for crisp edge detection," Jun. 2021, *arXiv:2011.09808*, doi: 10.48550/arXiv.2011.09808.

[68] X. Soria, Y. Li, M. Rouhani, and A. D. Sappa, "Tiny and efficient model for the edge detection generalization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV) Workshops*, 2023, pp. 1364–1373, doi: 10.1109/ICCVW60793.2023.00147.

[69] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "CASENet: Deep category-aware semantic edge detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1761–1770, doi: 10.1109/CVPR.2017.191.

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*, doi: 10.48550/arXiv.1512.03385.

[71] Y. Hu, Y. Chen, X. Li, and J. Feng, "Dynamic feature fusion for semantic edge detection," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 782–788, doi: 10.5555/3367032.3367144.

[72] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, 2015, pp. 234–241.

[73] Z. Zhou et al., "UNet: A nested U-Net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, D. Stoyanov Eds. Cham, Switzerland: Springer, 2018, pp. 3–11, doi: 10.1007/978-3-030-00889-5_1.

[74] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.

[75] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Comput. Vis. Conf.*, 2018, pp. 833–851, doi: 10.1007/978-3-030-01234-249.

[76] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*, doi: 10.48550/arXiv.1706.05587.

[77] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," 2017, *arXiv:1606.00915*, doi: 10.48550/arXiv.1606.00915.

[78] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," in *Proc. IEEE Vis. Commun. Image Process.*, 2017, pp. 1–4, doi: 10.1109/VCIP.2017.8305148.

[79] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010, doi: 10.5555/3295222.3295349.

[80] A. Dosovitskiy et al., "An image is worth 16 × 16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021, doi: 10.48550/arXiv.2010.11929.

[81] T. Fan, G. Wang, Y. Li, and H. Wang, "MA-Net: A multi-scale attention network for liver and tumor segmentation," *IEEE Access*, vol. 8, pp. 179656–179665, 2020, doi: 10.1109/ACCESS.2020.3025372.

[82] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 7242–7252, doi: 10.1109/ICCV48922.2021.00717.

[83] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 12077–1209, doi: 10.5555/3540261.3541185.

[84] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling, "EDTER: Edge detection with transformer," 2022, *arXiv:2203.08566*, doi: 10.48550/arXiv.2203.08566.

[85] Y. Yang, L. Zhang, L. Ren, and X. Wang, "MMViT-Seg: A lightweight transformer and CNN fusion network for COVID-19 segmentation," *Comput. Methods Programs Biomed.*, vol. 230, 2023, Art. no. 107348, doi: 10.1016/j.cmpb.2023.107348.

[86] H. Liu, J. Yang, X. Miao, C. Mertz, and H. Kong, "CrackFormer network for pavement crack segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 9, pp. 9240–9252, Sep. 2023, doi: 10.1109/TITS.2023.3266776.

[87] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[88] J. Chen et al., "TransUNet: Transformers make strong encoders for medical image segmentation," Feb. 2021, *arXiv:2102.04306*, doi: 10.48550/arXiv.2102.04306.

[89] H. Cao et al., "Swin-Unet: Unet-like pure transformer for medical image segmentation," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2022, pp. 205–218, doi: 10.1007/978-3-031-25066-89.

[90] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9992–10002, doi: 10.1109/ICCV48922.2021.00986.

[91] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," 2022, *arXiv:2111.00396*, doi: 10.48550/arXiv.2111.00396.

[92] J. Ruan and S. Xiang, "VM-UNet: Vision mamba UNet for medical image segmentation," Feb. 2024, *arXiv:2402.02491*, doi: 10.48550/arXiv.2402.02491.

[93] Y. Ye, K. Xu, Y. Huang, R. Yi, and Z. Cai, "DiffusionEdge: Diffusion probabilistic model for crisp edge detection," *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 7, pp. 6675–6683, 2024, doi: 10.1609/aaai.v38i7.28490.

[94] J. Wu et al., "MedSegDiff: Medical image segmentation with diffusion probabilistic model," in *Proc. Int. Conf. Med. Imag. Deep Learn.*, 2023, pp. 1623–1639. [Online]. Available: https://proceedings.mlr.press/v227/wu24a.html

[95] J. Wu, W. Ji, H. Fu, M. Xu, Y. Jin, and Y. Xu, "MedSegDiff-V2: Diffusion based medical image segmentation with transformer," *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 6, pp. 6030–6038, 2023, doi: 10.1609/aaai.v38i6.28418.

[96] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Red Hook, NY, USA: Curran Associates Inc., 2019, doi: 10.48550/arXiv.1912.01703.

[97] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2018, *arXiv:1708.07120*, doi: 10.48550/arXiv.1708.07120.

[98] P. Ren et al., "A comprehensive survey of neural architecture search: Challenges and solutions," Mar. 2021, *arXiv:2006.02903*, doi: 10.48550/arXiv.2006.02903.

[99] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," Jan. 2021, *arXiv:2101.09336*, doi: 10.48550/arXiv.2101.09336.

[100] S. Shekhar, A. Bansode, and A. Salim, "A comparative study of hyper-parameter optimization tools," Jan. 2022, *arXiv:2201.06433*, doi: 10.48550/arXiv.2201.06433.

[101] F. Hader, "SimCATS_GaAs_v1_random_variations_v2," Oct. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13903285

Fabian Hader received the B.Sc. degree in scientific programming and the M.Sc. degree in energy economics and informatics from the FH Aachen—University of Applied Sciences, Campus Jülich, Jülich, Germany, in 2019 and 2021, respectively. He is currently working toward the Ph.D. degree in engineering with the University of Duisburg-Essen, Duisburg, Germany.

From 2019 to 2021, he was a Software Engineer with the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich. His research interests include the automatic tuning of quantum dots.

Fabian Fuchs received the B.Sc. and M.Sc. degrees in applied mathematics and computer science from the FH Aachen—University of Applied Sciences, Campus Jülich, Jülich, Germany, in 2021 and 2023, respectively, and the M.Sc. degree in mathematics from the University of Wisconsin—Milwaukee, Milwaukee, WI, USA, in 2023. He is currently working toward the Ph.D. degree in computer science with the University of Mannheim, Mannheim, Germany, in cooperation with the Fraunhofer Institute for Industrial Mathematics, Kaiserslautern, Germany.

From 2018 to June 2024, he was a Software Engineer with the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich. His research interests include deep learning and computer vision.

Sarah Fleitmann received the B.Sc. degree in scientific programming and the M.Sc. degree in applied mathematics and informatics from the FH Aachen—University of Applied Sciences, Campus Jülich, Jülich, Germany, in 2020 and 2022, respectively.

Since 2017, she has been a Software Engineer with the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich. Her research interests include the automatic tuning of quantum dots for their operation as qubits.

Karin Havemann received the B.Sc. degree in applied mathematics and computer science from the FH Aachen—University of Applied Sciences, Campus Jülich, Jülich, Germany, in 2022, where she is currently working toward the M.Sc. degree in applied mathematics and computer science.

Since 2023, she has been a Software Engineer with the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich.

Benedikt Scherer received the B.Sc. degree in scientific programming and the M.Sc. degree in applied mathematics and informatics from the FH Aachen—University of Applied Sciences, Campus Jülich, Jülich, Germany, in 2020 and 2023, respectively.

Since 2017, he has been a Software Engineer with the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich.

Jan Vogelbruch received the Dipl.Ing. and Dr.-Ing. degrees in electrical engineering from RWTH Aachen University, Aachen, Germany, in 1994 and 2003, respectively.

In 1995, he joined Parsytec Computer GmbH, Aachen, as a Technical Project Manager for European cooperations. His focus has been on high-performance computing and image processing solutions, where he has been the technical leader for the company's part in several European-Commission-funded projects. Since late 1998, he has been with the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich, Germany. His research interests include parallel computing, signal and 3-D image processing, fast reconstruction methods for high-resolution computer tomography, and automated defect detection, as well as the automatic tuning of semiconductor quantum dots.

Lotte Geck received the B.Sc. degree in electrical engineering and M.Sc. degree in information technology and computer engineering from RWTH Aachen University, Aachen, Germany, in 2013 and 2015, respectively, and the Dr.-Ing. degree in electrical engineering and information technology from RWTH Aachen University, Aachen, Germany, in 2021.

In 2016, she joined the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich, Germany. Since 2022, she has been a Junior Professor with Forschungszentrum Jülich and RWTH Aachen University. Her research interests include scalable electronic system solutions for quantum computing.

Stefan van Waasen received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from Gerhard-Mercator University, Duisburg, Germany, in 1994 and 1999, respectively.

The topic of his doctoral thesis was optical receivers up to 60 GB/s based on traveling wave amplifiers. In 1998, he joined Siemens Semiconductors/Infineon Technologies AG, Düsseldorf, Germany. His responsibility was to develop BiCMOS and CMOS RF systems for highly integrated cordless systems, such as Digital Enhanced Cordless Telecommunications and Bluetooth. In 2001, he moved to the IC development of front-end systems for high-data-rate optical communication systems. From 2004 to 2006, he was with the Stockholm Design Center, Stockholm, Sweden, and was responsible for the short-range analog, mixed-signal, and RF development for system-on-chip (SoC) CMOS solutions. From 2006 to 2010, he was responsible for wireless RF system engineering in the area of SoC CMOS products at headquarters in Munich, Germany, and later at the Design Center Duisburg, Duisburg. Since 2010, he has been the Director of the Peter Grünberg Institute—Integrated Computing Architectures, Forschungszentrum Jülich GmbH, Jülich, Germany. In 2014, he became a Professor in Measurement and Sensor Systems with the Communication Systems Chair, University of Duisburg-Essen, Duisburg. His research interests include complex measurement and detector systems, particularly on electronic systems for quantum computing.