

**SANDIA REPORT**

SAND20XX-XXXX

Printed Click to enter a date

**Sandia  
National  
Laboratories**

# Super-Resolution Approaches in Three-Dimensions for Classification and Screening of Commercial-Off-The-Shelf Components

Andrew T. Polonsky, Carianne Martinez, Catherine A. Appleby, Sylvain R. Bernard,  
James J.M. Griego, Philip J. Noell, and Priya R. Pathare

Prepared by  
Sandia National Laboratories  
Albuquerque, New Mexico  
87185 and Livermore,  
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831

Telephone: (865) 576-8401  
Facsimile: (865) 576-5728  
E-Mail: [reports@osti.gov](mailto:reports@osti.gov)  
Online ordering: <http://www.osti.gov/scitech>

Available to the public from

U.S. Department of Commerce  
National Technical Information Service  
5301 Shawnee Rd  
Alexandria, VA 22312

Telephone: (800) 553-6847  
Facsimile: (703) 605-6900  
E-Mail: [orders@ntis.gov](mailto:orders@ntis.gov)  
Online order: <https://classic.ntis.gov/help/order-methods/>



## **ABSTRACT**

X-ray computed tomography is generally a primary step in characterization of defective electronic components, but is generally too slow to screen large lots of components. Super-resolution imaging approaches, in which higher-resolution data is inferred from lower-resolution images, have the potential to substantially reduce collection times for data volumes accessible via x-ray computed tomography. Here we seek to advance existing two-dimensional super-resolution approaches directly to three-dimensional computed tomography data. Multiple scan resolutions over a half order of magnitude of resolution were collected for four classes of commercial electronic components to serve as training data for a deep-learning, super-resolution network. A modular python framework for three-dimensional super-resolution of computed tomography data has been developed and trained over multiple classes of electronic components. Initial training and testing demonstrate the vast promise for these approaches, which have the potential for more than an order of magnitude reduction in collection time for electronic component screening.

This page left blank

## CONTENTS

Abstract.....	3
Acronyms and Terms .....	7
1. INTRODUCTION .....	9
2. DATA COLLECTION.....	11
2.1. XCT Methodology.....	11
2.2. XCT Datasets .....	14
3. SUPER-RESOLUTION METHODS.....	19
3.1. Background Information .....	19
3.1.1. Deep Learning Methods .....	19
3.2. Baseline Approach.....	19
3.3. Deep Learning Models.....	20
4. EXPERIMENTS .....	23
4.1. Data preparation .....	23
4.2. Implementation details.....	24
4.3. Diode experiments.....	26
4.4. Capacitor experiment .....	26
5. RESULTS.....	27
5.1. Diode .....	27
5.2. Capacitor .....	30
6. CONCLUSION.....	31
7. References .....	33
Appendix A. NN Architecture details .....	35
A.1. ResNet Generator.....	35
A.2. Discriminator.....	39
A.3. Feature Extractor.....	40
Distribution.....	41

## LIST OF FIGURES

Figure 1 – Five Ta capacitors mounted to toothpick and Al rod .....	12
Figure 2 – Xradia 620 Versa front view .....	13
Figure 3 – Zeiss Xradia 620 Versa with all 4 cabinet doors open. Orange circled region is the X-ray tube/source and rotary beam filter setup. Blue circled region is where the sample is loaded. Red circled region is the camera objectives and detector. ....	13
Figure 4 – Ta capacitor #1 with a range of voxel resolutions of (a) 4.7 $\mu\text{m}$ (b) 7.0 $\mu\text{m}$ (c) 9.4 $\mu\text{m}$ and (d) 23.4 $\mu\text{m}$ . Dark regions of the image correspond to either air surrounding device or low Z materials in device packaging. Medium pixel brightness values match the wire connections. The transition between medium and the brightest contains the bulk Ta material. Brightest pixels are the internal electrode connection points. ....	14
Figure 5 – Resistor #1 with a range of voxel resolutions of (a) 3.2 $\mu\text{m}$ (b) 6.3 $\mu\text{m}$ and (c) 16 $\mu\text{m}$ . ...	15
Figure 6 – MOSFET #1 with a range of voxel resolutions of (a) 7.1 $\mu\text{m}$ (b) 10.7 $\mu\text{m}$ and (c) 14.2 $\mu\text{m}$ . ....	15

Figure 7 – TVS Diode capacitor #1 with a range of voxel resolutions of (a) 4.2  $\mu\text{m}$  (b) 5.6  $\mu\text{m}$  and (c) 14  $\mu\text{m}$ .....16

Figure 8 – Normalized data collection time per part for XCT characterization for each studied part at various reductions in resolution. ....17

Figure 9 - Comparison of the output from the 2D baseline interpolation method (right) with the ground truth high resolution scan (center), and the low-resolution input (left) of a diode. Interpolation is unable to overcome noise and artifacts associated with low resolution scans.....20

Figure 10 - 2D deep learning training example: SRGAN is used to learn a function that produces a clearer diode image slice (right) from a low-resolution scan (left) that qualitatively captures the details of the high-resolution scan (center). This 2D version of the model served as a sanity check that the SRGAN was amenable to processing CT scan data. ....21

Figure 11 – Average, maximum, and minimum greyscale values for each slice of a low-resolution XCT scan of a TVS diode. ....23

Figure 12 – A sequence of the first 15 slices of the low-resolution XCT scan of the TVS diode. The first 10 slices (top 2 rows) are not within the part of interest and must be removed for training the super-resolution network. ....24

Figure 13 - Slice through a TVS diode 3D training example with the low-resolution scan (left), the ground truth high-resolution scan (center) and VNet output (right).....28

Figure 14 - Slice through a diode held-out test example with the low-resolution scan (left), the ground truth high-resolution scan (center) and VNet output (right). The VNet captures some of the high-resolution details but produces an image with approximately 4x error compared to the training set error. ....29

Figure 15 - Qualitative comparison of VNet fit to produce a high-resolution image of a capacitor. The difference between the low-resolution and high-resolution scans used for training is subtle, and we used only one capacitor example for both training and testing to demonstrate the feasibility of our workflow on an alternate electronic part.....30

**LIST OF TABLES**

Table 1 – Completed XCT scans with resolution, X-ray tube voltage, and beam filter.....11

Table 2 - VNet Generator architecture .....25

Table 3 – Mean absolute error over diode sets with respect to ground truth high-resolution scans. .27

## ACRONYMS AND TERMS

Acronym/Term	Definition
3D	three-dimensional
XCT	X-ray computed tomography
COTS	commercial-off-the-shelf
CNN	convolutional neural network
GAN	generative adversarial network
2D	two-dimensional
TVS	transient-voltage-suppressor
MOSFET	metal-oxide-semiconductor field-effect transistors
GPU	graphics processing unit
NN	neural network
SRGAN	super-resolution generative adversarial network

This page left blank

## 1. INTRODUCTION

Three-dimensional (3D) characterization approaches enable the collection of rich, multi-modal datasets that can provide new insight to complex material systems. As compared to conventional characterization, where only a handful of cross-sections can reasonably be inspected for any given component, three-dimensional characterization interrogates much larger volumes of material, and so can capture rare events, including defects or microstructural aberrations at the tails of feature distributions. Three-dimensional characterization, therefore, has the potential to act as a screening tool for quality control that exceeds the capabilities of other testing methods.

Particularly practical techniques such as X-ray computed tomography (XCT) offer a non-destructive evaluation methodology that would allow for inspection of components without affecting their performance or utility. XCT is an essential tool for understanding defective behavior in faulty commercial-off-the-shelf (COTS) electronics, and is commonly employed as a first step prior to more involved destructive physical analysis (DPA) of individual components. The non-destructive nature of XCT makes the approach broadly applicable across many component classes, which enables three-dimensional analysis of a faulty component without physically altering the part. XCT therefore can provide critical information on the location and nature of the defect, which can inform more targeted and specialized testing methodologies that are generally destructive in nature. However, XCT is generally ill-suited for large-scale screening and inspection activities, as the high-resolution data typically required to identify microstructural defects or individual components within electronic devices necessitates long data collection times, reducing the potential throughput of these approaches.

The ability to infer higher resolution data from low-resolution collection, a process known as super-resolution imaging, has received significant interest from the computer vision and machine learning community in the last decade. Using deep learning methods such as convolutional neural networks (CNNs) or generative adversarial networks (GANs), a machine learning framework can be developed to improve the resolution of an image as if it was collected at higher fidelity. Application of such methods to XCT data could enable the collection of low-resolution datasets without loss of data quality, dramatically improving throughput of this method for component screening. Due to material interactions with the X-ray source used to generate tomograms, certain types of defects, especially near material interfaces in multi-material components, can be difficult to identify reliably in XCT data. Serially sectioning approaches, such as those using optical or electron imaging, generally offer better fidelity at material interfaces and allows for easier identification of internal components. However, these techniques suffer from the fact that they are generally destructive in nature. Development of a mapping between different types of three-dimensional characterization data, based on the same framework used for super-resolution imaging, could provide the benefits of both characterization techniques without the need for destructive characterization. A super resolution approach solely focusing on XCT data is therefore a first step to bridging the gap between mixed modalities of high fidelity 3D data.

In this project, we seek to apply state of the art machine learning techniques to expand the applicability of computed tomography as a production level tool by enabling large reductions in data acquisition rates via deep learning super-resolution approaches previously applied to two-dimensional (2D) data alone.

This page left blank

## 2. DATA COLLECTION

In this section, we present XCT characterization work performed on four classes of COTS electronics components.

### 2.1. XCT Methodology

Four families of COTS electronic parts were characterized using XCT. These four families included Ta-polymer capacitors (T541X336M050BH6710 22uF 50V), transient-voltage-suppressor (TVS) diodes (3.3 V 19 V CDSOD323-T03CT0-ND), metal-oxide-semiconductor field-effect transistors (MOSFETs) (SIC MOSFET N-CH 3), and resistors (RES SMD 10M Ohm). Multiple devices from each family were characterized. The exact number of devices from each family which were characterized using XCT, as well as the resolution, X-ray tube voltage, and beam filter are summarized in Table 1.

**Table 1 – Completed XCT scans with resolution, X-ray tube voltage, and beam filter**

Device Name	Resolution 1	Resolution 2	Resolution 3	Resolution 4	Resolution 5
Ta capacitor (10 devices)	4.7 $\mu\text{m}$ 160 kV / HE18 4.2 hr per device	7 $\mu\text{m}$ 160 kV / HE18 2.8 hr per device	9.4 $\mu\text{m}$ 160 kV / HE18 2.6 hr per device	23.4 $\mu\text{m}$ 160 kV / HE18 0.6 hr per device	--
TVS diode (15 devices)	2.8 $\mu\text{m}$ 50 kV / no filter 2.6 hr per device	4.2 $\mu\text{m}$ 50 kV / no filter 0.7 hr per device	5.6 $\mu\text{m}$ 50 kV / no filter 0.3 hr per device	14 $\mu\text{m}$ 50 kV / no filter 0.2 hr per device	28 $\mu\text{m}$ 50 kV / no filter <0.1 hr per device
MOSFET (4 devices)	7.1 $\mu\text{m}$ 100 kV / LE6 0.9 hr per device	10.7 $\mu\text{m}$ 100 kV / LE6 0.8 hr per device	14.2 $\mu\text{m}$ 100 kV / LE6 0.8 hr per device	35.5 $\mu\text{m}$ 100 kV / LE6 0.2 hr per device	--
Resistor (5 devices)	3.2 $\mu\text{m}$ 60 kV / LE2 1.4 hr per device	4.7 $\mu\text{m}$ 60 kV / LE2 1 hr per device	6.3 $\mu\text{m}$ 60 kV / LE2 1 hr per device	16 $\mu\text{m}$ 60 kV / LE2 0.4 hr per device	--

Electronic devices were glued to a toothpick or plastic rod using Bondic liquid plastic (ultraviolet light curable). An example of the mounted Ta capacitors is shown in Figure 1. Depending on the individual device size, 4 to 15 separate devices can fit onto the toothpick or plastic rod. The mounted setup was then affixed to an aluminum rod XCT sample holder. Scans were performed using a lab scale XCT instrument (Zeiss Xradia 620 Versa, Carl Zeiss XRM, Pleasanton, CA) equipped with a tungsten X-ray tube, see Figure 2 and Figure 3 below. The scan voxel size and accelerating voltage for each specimen is noted in Table 1. Initial scans of each part were performed to determine the working voltage that best characterized the features of interest within the part. This consisted of scanning a device from each family of parts at multiple working voltages, ranging from 40 kV to 160 kV. These datasets were subsequently examined to determine which conditions best revealed the features of interest.

Scanning was done with a Flat Panel detector with pixel binning of 1. Beam filtering varied depending on the device type, the details of which are also noted in Table 1. Radiographic reconstructions were performed within the Zeiss Reconstructor utilizing 3201 projections. Post reconstruction analysis with cropping and view alignment was completed with Dragonfly 3D

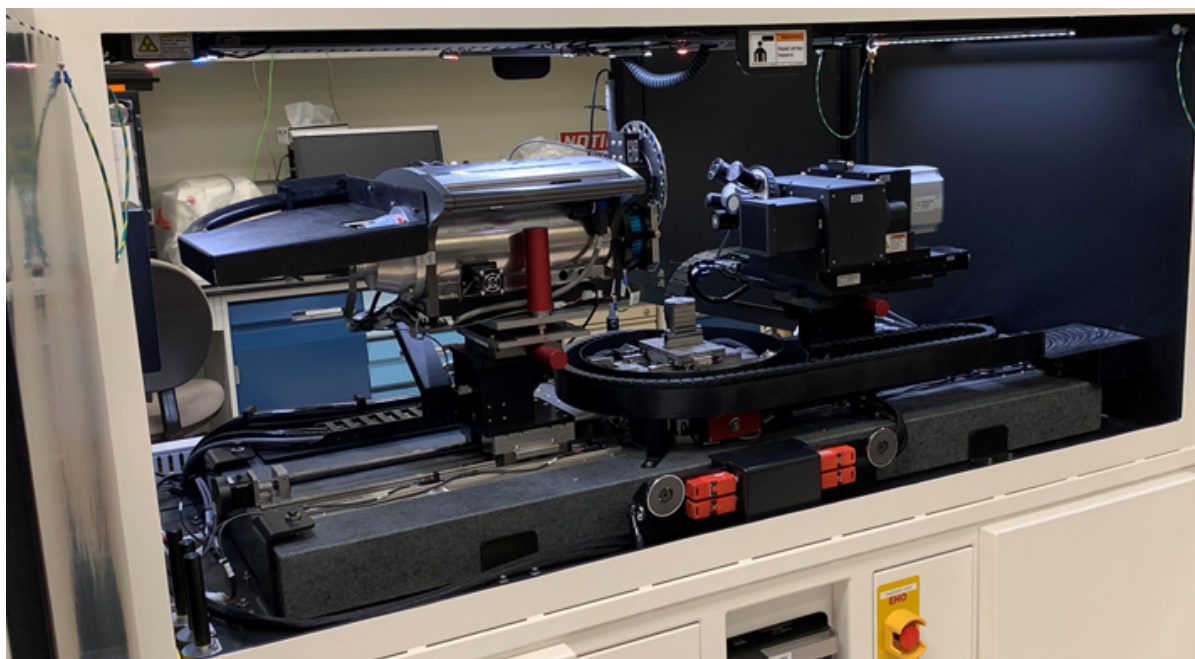
software v3.6.1 or newer (Object Research Systems (ORS) Inc, Montreal, Canada, 2020) and FIJI v1.53q.



**Figure 1 – Five Ta capacitors mounted to toothpick and Al rod**



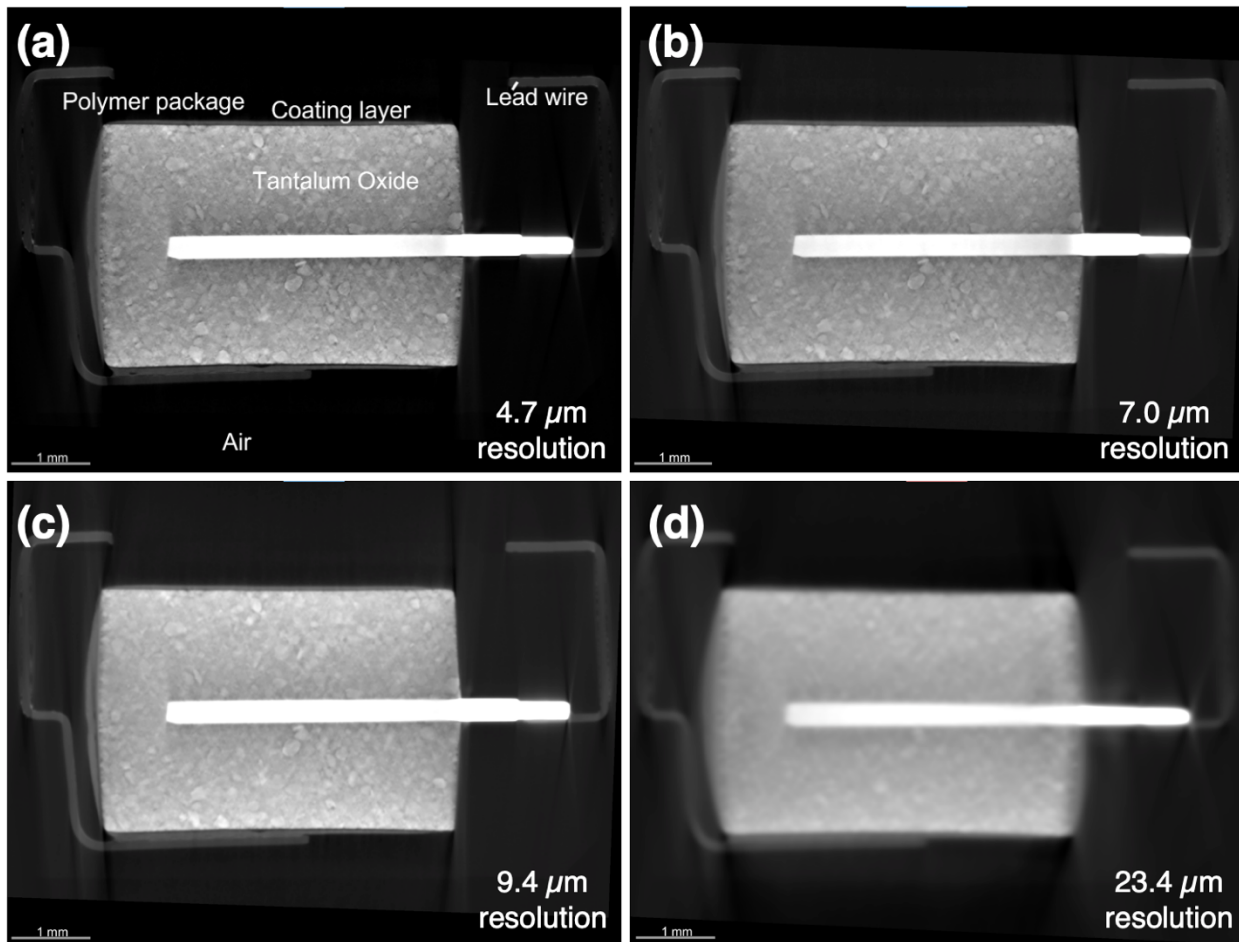
**Figure 2 – Xradia 620 Versa front view**



**Figure 3 – Zeiss Xradia 620 Versa with all 4 cabinet doors open. Orange circled region is the X-ray tube/source and rotary beam filter setup. Blue circled region is where the sample is loaded. Red circled region is the camera objectives and detector.**

## 2.2. XCT Datasets

Each device was scanned at the best possible resolution, defined as the smallest voxel size that would fit the sample in the field of view. Additional magnifications were then set to 1.5x, 2x, and 5x coarser of this starting resolution. The TVS diodes were additionally scanned at a resolution 10x coarser than the initial dataset. An example set shown in Figure 4 was taken from the Ta capacitor set. These images show the transition from high (smallest voxel size) to low (largest voxel size) resolution for the scan series. Each image is a slice view of the Ta device, with the centering position adjusted so that comparisons can be made stepping from the highest resolution (4.7  $\mu\text{m}$ ) to the lowest resolution (23.4  $\mu\text{m}$ ). The darkest pixels in the images correspond to either low absorbing materials or air regions of the scan. The intensity scale increases as a function of how the materials absorb the X-ray beam. Beam hardening artifacts are seen in some of the regions of the images, they appear as feathered regions when you transition from the brightest features into dark pixel regions. As expected, with decreasing resolution, details of the Ta capacitor become increasingly harder to resolve. For reference, Figure 5, Figure 6, and Figure 7 show 2D representations of 3D XCT data from a representative Resistor, MOSFET, and TVS diode, respectively at various resolutions.



**Figure 4 – Ta capacitor #1 with a range of voxel resolutions of (a) 4.7  $\mu\text{m}$  (b) 7.0 $\mu\text{m}$  (c) 9.4  $\mu\text{m}$  and (d) 23.4  $\mu\text{m}$ . Dark regions of the image correspond to either air surrounding device or low Z materials in device packaging. Medium pixel brightness values match the wire connections. The transition between medium and the brightest contains the bulk Ta material. Brightest pixels are the internal electrode connection points.**

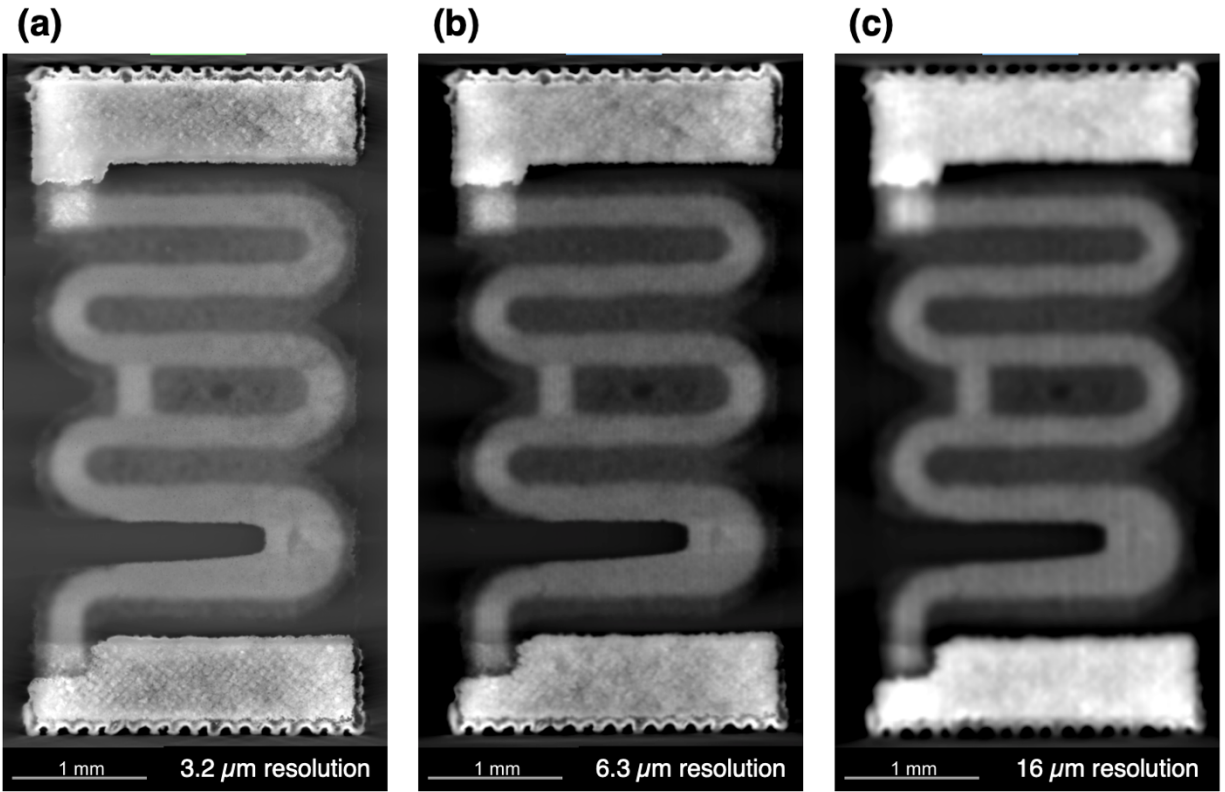


Figure 5 – Resistor #1 with a range of voxel resolutions of (a) 3.2 μm (b) 6.3 μm and (c) 16 μm.

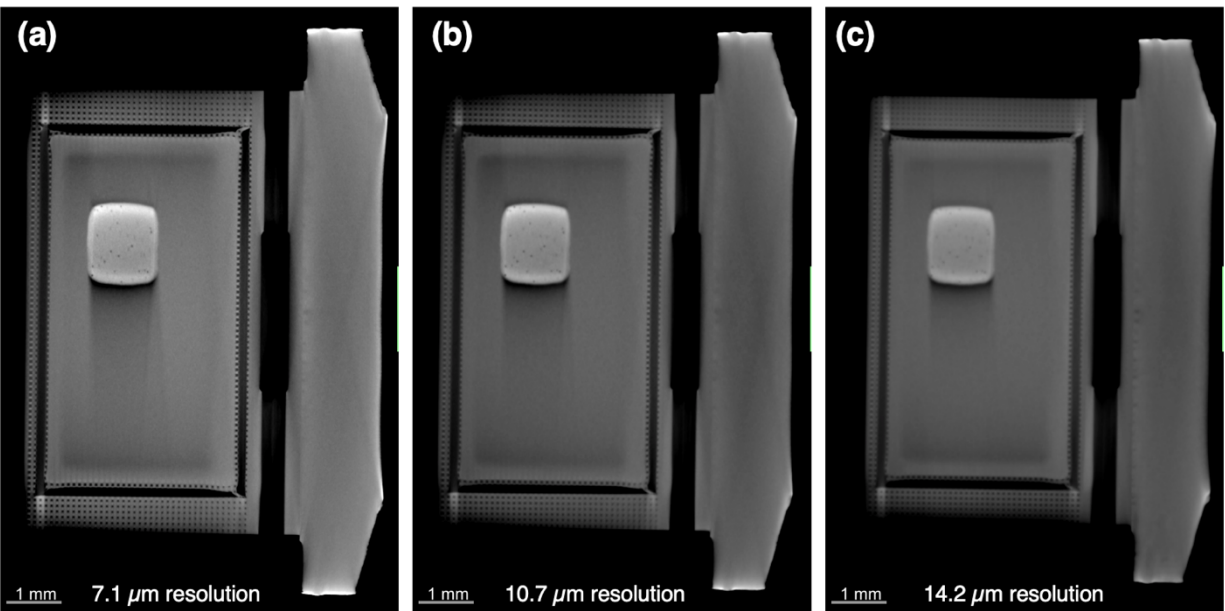
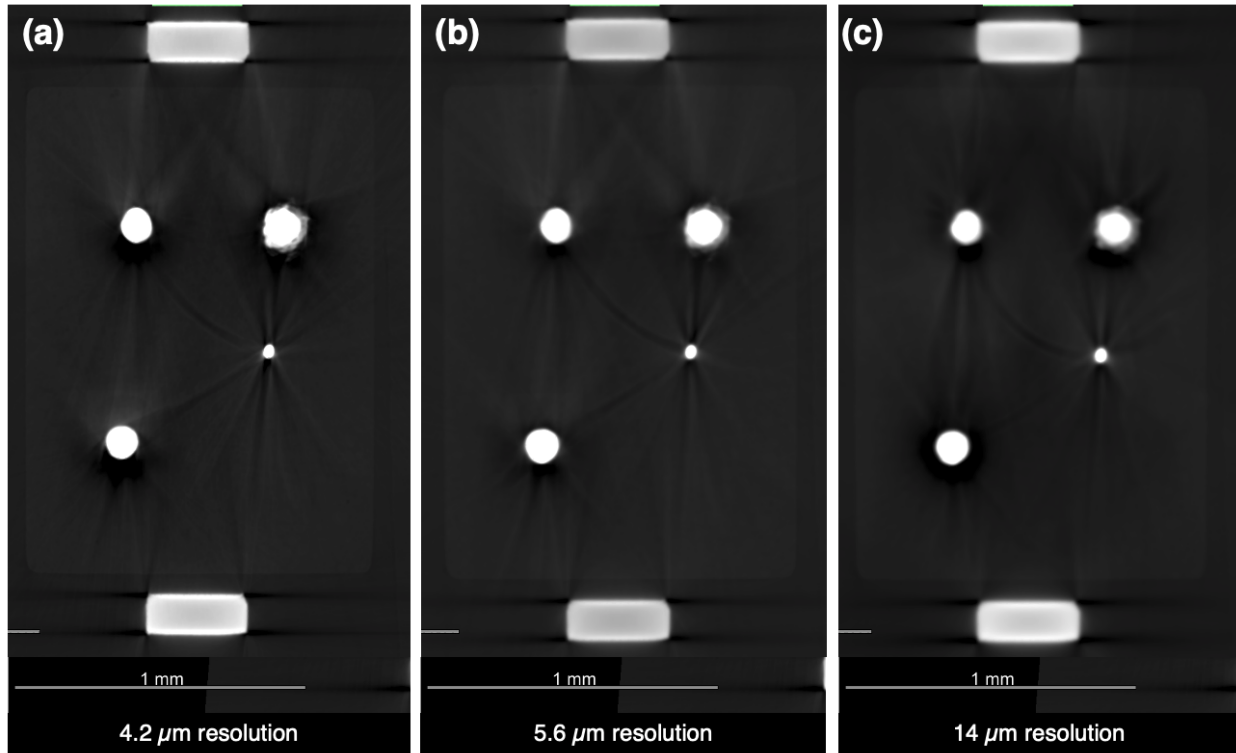


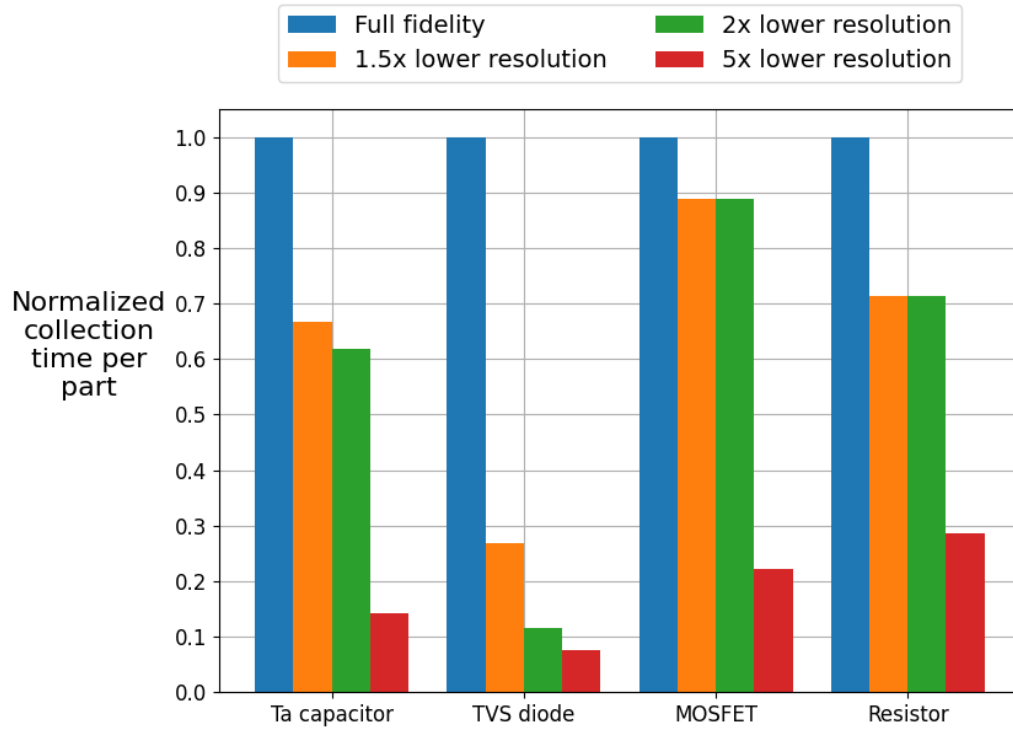
Figure 6 – MOSFET #1 with a range of voxel resolutions of (a) 7.1 μm (b) 10.7 μm and (c) 14.2 μm.



**Figure 7 – TVS Diode capacitor #1 with a range of voxel resolutions of (a) 4.2  $\mu\text{m}$  (b) 5.6  $\mu\text{m}$  and (c) 14  $\mu\text{m}$ .**

As Table 1 highlights, scan time significantly varied part to part. This was a function of the amount of time necessary to collect XCT data for a given part at a given resolution. For example, XCT data collection at a 7  $\mu\text{m}$  voxel size took  $\approx 3\text{X}$  longer for Ta capacitors than for MOSFETS. Additionally, for a given family of devices, the scan time decreased significantly with decreasing scan resolution. The relative times to collect XCT data for various part families, normalized to the full fidelity, highest resolution scan, are shown in Figure 8.

Scan time decreased by between 70 and 90% for the TVS diode components, while the same range or relative resolutions gave a reduction of between 10 and 75% for the MOSFET devices. These differences in absolute normalized collection time per device are subject to the specific dimensions of a particular device and its apparent size in the effective field of view as resolution is decreased. However, in all, cases a 5X reduction in resolution, a reduction well-within the range of most super resolution approaches, led to a reduction of data collection time by no less than 70%. This dramatic reduction in collection time forms the basic motivation for seeking super resolution approaches to reduce data collection time for characterization of COTS components, which will be discussed in more detail in the following section.



**Figure 8 – Normalized data collection time per part for XCT characterization for each studied part at various reductions in resolution.**

This page left blank

### 3. SUPER-RESOLUTION METHODS

In this section, we present the state of the art in super-resolution methods. We discuss our modifications to these methods, and we document the details of data preparation and model architectures.

#### 3.1. Background Information

Several approaches for super-resolution have been proposed in the literature. The goal of super-resolution is to take a low-resolution image as input and learn a function that maps the low-dimensional image to an approximation of a high-resolution version of the input image. Most of these tools have been developed for two-dimensional (2D) images, and the extension to 3D is a focus of our work.

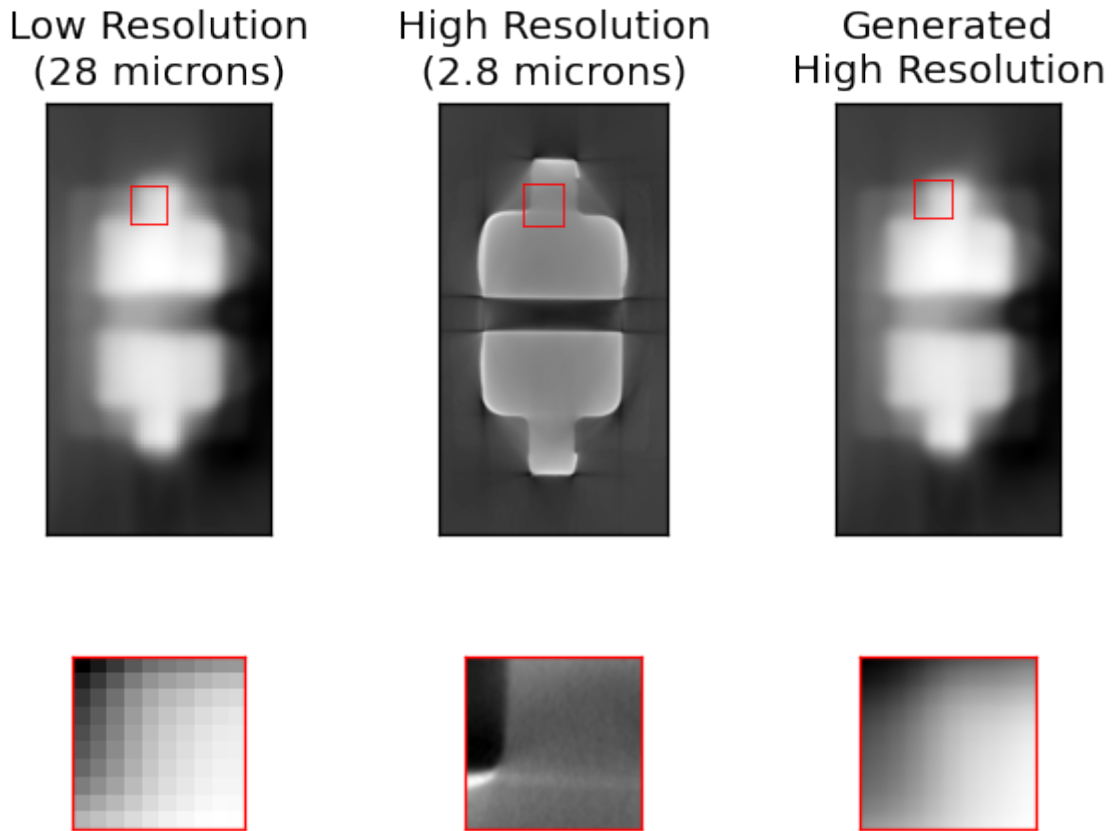
##### 3.1.1. Deep Learning Methods

Deep learning has become the dominant approach to many computer vision and image processing problems. Several super-resolution approaches have emerged in recent years, primarily using generative models that use convolutional neural networks (CNNs) (LeCun, et al., 1989) to create a higher resolution output image given a lower resolution input image. In contrast to traditional computer vision methods that rely on expert-engineered features to detect regions of interest in images, CNNs learn the features most relevant to a downstream task by training directly on image data.

In this work, we base our implementation on the Super-Resolution Generative Adversarial Network (SRGAN), a generative adversarial network for super-resolution that uses a CNN as a generative model to produce high resolution versions of input images (Ledig, et al., 2017). GANs (Goodfellow, et al., 2014) consist of a pair of models that are trained under a game theory paradigm. The generator model creates content with the goal to mimic the training distribution (in our case, high resolution CT scans) while a discriminator model learns to distinguish generator output from real training data. Both models are typically neural networks and compete to outperform one another, with the objective of the generator being to fool the discriminator. The key advance of the SRGAN for super-resolution is to learn both from the typical GAN training loss but also to include a content loss based on activations derived from a feature extractor CNN model pretrained on ImageNet™ (Deng, et al., 2009), a large open-source dataset of approximately 1.3 million images.

#### 3.2. Baseline Approach

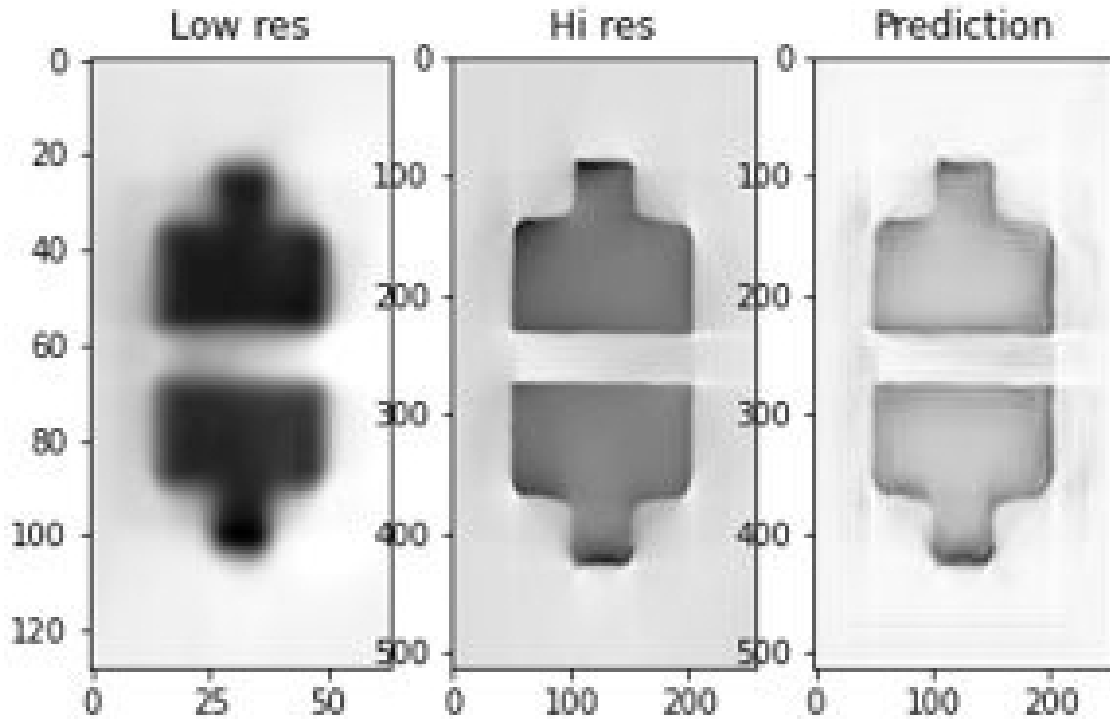
We implemented an interpolation-based method to establish a baseline for comparison with the deep learning super-resolution methods. This method uses the PyTorch (Paszke, Gross, Chintala, & Chanan, 2017) python library to upscale the low-resolution scan to have the same dimensions as its high-resolution counterpart. We experimented with different interpolation algorithms, including comparing 2D to 3D interpolation methods and found that the qualitatively best result came from 2D bicubic interpolation from the python OpenCV library (OpenCV, 2022) while the best quantitative comparison could be calculated against the 3D trilinear interpolation method provided via the `upscale` method available in the PyTorch `nn` module. Figure 9 shows an example of the output of the baseline method compared with the ground truth high resolution scan.



**Figure 9 - Comparison of the output from the 2D baseline interpolation method (right) with the ground truth high resolution scan (center), and the low-resolution input (left) of a diode. Interpolation is unable to overcome noise and artifacts associated with low resolution scans.**

### 3.3. Deep Learning Models

We began by implementing the SRGAN for use with electronic parts CT scans. As a sanity check, we used 2D slices of the scans for training and inspected the output of the generator model that produced 2D slices of a high-resolution image. Figure 10 shows an example of the output from the 2D model from the training set. A major disadvantage of the 2D method lurks in the need to align relevant slices from low-dimensional to high-dimensional scans. This manual effort would be required to produce sufficiently large training sets to assess the 2D model’s generalization capacity and as such, we proceeded to build a 3D model.



**Figure 10 - 2D deep learning training example: SRGAN is used to learn a function that produces a clearer diode image slice (right) from a low-resolution scan (left) that qualitatively captures the details of the high-resolution scan (center). This 2D version of the model served as a sanity check that the SRGAN was amenable to processing CT scan data.**

While some manual data preparation was required in 3D, it was limited to finding an appropriate volumetric bounding box around both the low-resolution and high-resolution images. A key challenge to this project was to extend the SRGAN architecture to support 3D images. We considered two types of generator models for our super-resolution framework: a ResNet (He, Zhang, Ren, & Sun, 2016) similar to the architecture set forth in the SRGAN paper that we extended to include 3D neural network (NN) layers and a VNet, a volumetric model introduced in (Milletari, Navab, & Ahmadi, 2016) for segmentation tasks.

ResNets (He, Zhang, Ren, & Sun, 2016) use residual blocks to incorporate loss resulting from approximating function residuals that enables successful training of deeper NNs with several layers. In the context of computer vision, these deeper NNs provide feature maps at different scales with respect to input images, enabling hierarchical information to be used for downstream tasks such as image classification.

The VNet consists of several down-sampling convolutional layers followed by symmetric upsampling layers that transform the input image to the desired output. Skip connections allow information from the downsampling layers to pass to corresponding upsampling layers along with feature maps calculated from the previous upsampling layer that identify features important for super-resolution conversion. In contrast to the VNet in the literature that predicts a class for each voxel, we architect our VNet as a generative model to produce output at the desired high-resolution scale.

In addition to the 3D SRGAN model, we experimented with the VNet alone as a straightforward supervised model using the mean absolute error as a loss function.

This page left blank

## 4. EXPERIMENTS

In this section, we provide details of our experimental setup and implementation.

### 4.1. Data preparation

For the two parts, diodes and capacitors, used to train the 3D models, some data preparation was required. XCT scans were collected in a tagged image file format (TIFF) image format and opened in the Python environment using Pillow (Clark, 2015), stored as 16-bit big endian unsigned integer pixels. They were then converted to greyscale NumPy arrays, with each voxel, or pixel within a slice, represented by a big-endian integer. In visualizing the data, we discovered that some data cleansing was required before training to ensure better results.

Visualizing the CT scans of the diodes in individual slices revealed that the removal of some slices at the beginning and ends of the 3D scans was required, effectively cropping the scan closer to the part. This removed excess slices on which we did not want the model to learn, as these regions do not correspond to data containing the part of interest. Plotting the minimum, maximum, and average of the arrays used to represent each individual slice over a sample and comparing visually to the slices, it was clear that a threshold would be useful in determining which slices to keep. For all samples at both resolutions studied here, 2.8 and 28  $\mu\text{m}$ , the maximum value increased drastically in slices necessary for training. Creating a threshold at 10,000 and removing slices with maximums under that threshold resulted in a reliable method for automated cropping of each sample. Figure 11 depicts the statistical values we used to crop a low-resolution diode scan. Figure 12 shows the first 15 slices of the dataset, 10 of which contain information which must be removed prior to training.

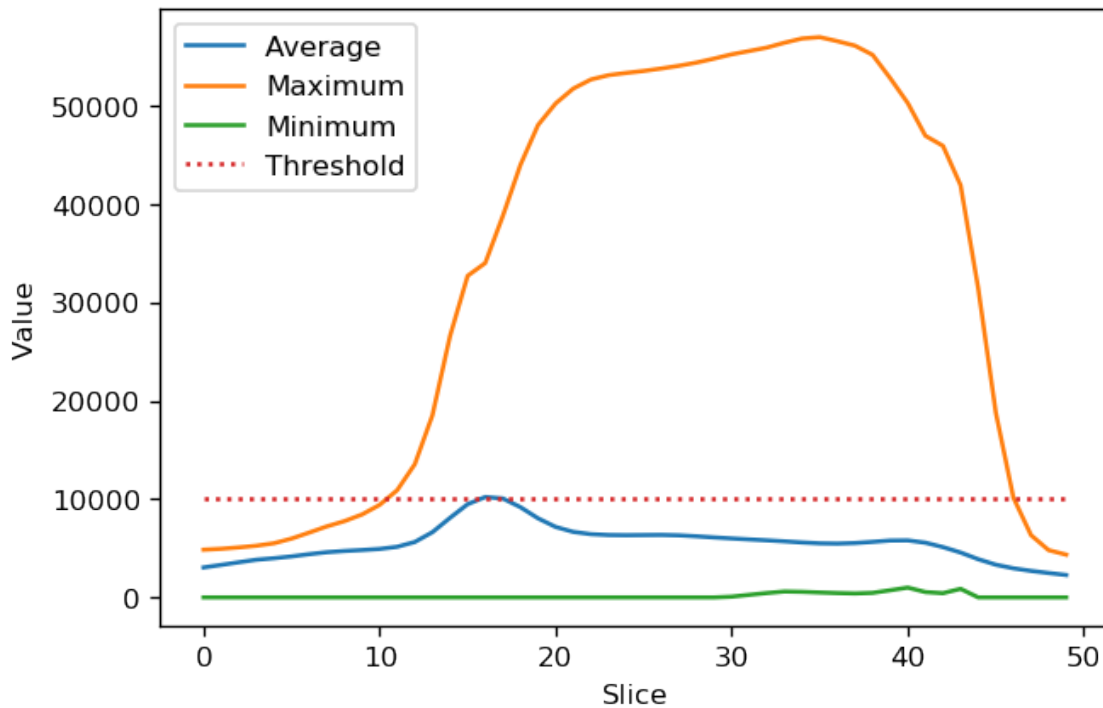
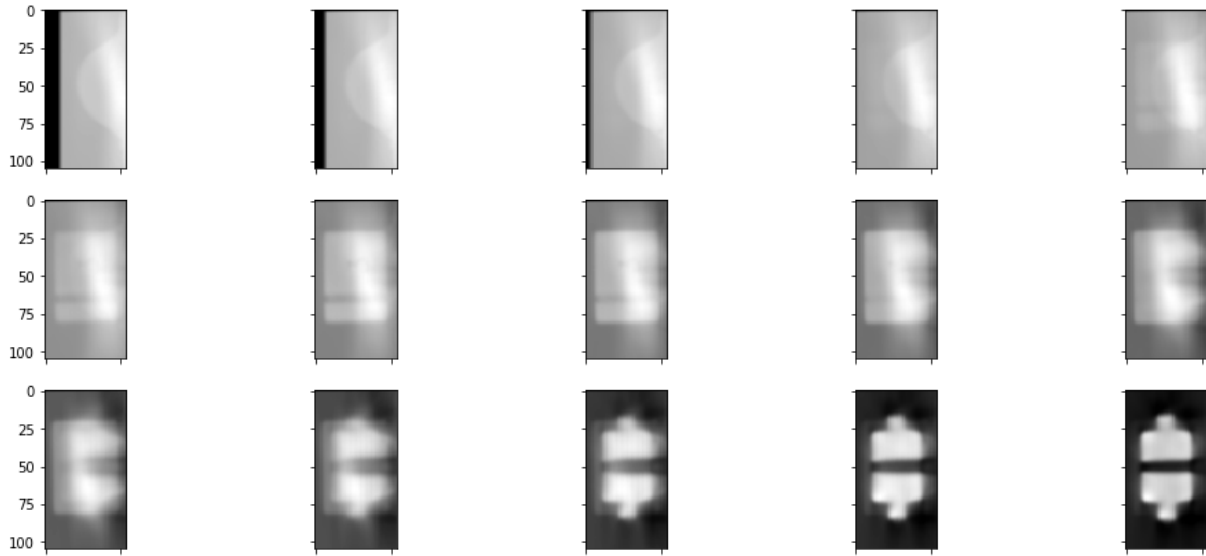


Figure 11 – Average, maximum, and minimum greyscale values for each slice of a low-resolution XCT scan of a TVS diode.



**Figure 12 – A sequence of the first 15 slices of the low-resolution XCT scan of the TVS diode. The first 10 slices (top 2 rows) are not within the part of interest and must be removed for training the super-resolution network.**

The four resolutions of capacitors, 23.4, 9.4, 7.0, and 4.7  $\mu\text{m}$ , did not have the same issue of excessive slices from outside of the part, with a solid black background surrounding the part. This is likely due to the large amount of higher-density materials within the capacitor as compared to the TVS diode. However, these scans did require other additional manual transformations in order to align the different resolution datasets. The 23.4  $\mu\text{m}$  resolution Ta capacitor dataset was rotated along two axes to match the 3D orientation of the other higher resolution datasets. Due to the manual nature of these transformations, the resulting alignment was insufficient for the lowest resolution scan, so the model was instead fit to the next largest resolution jump available from the scanned parts, from 4.7 down to 9.4  $\mu\text{m}$ .

## 4.2. Implementation details

Our super-resolution framework is implemented in PyTorch, and is based on an open-source 2D implementation (Linder-Noren, 2018). The framework consists of modular python code that enables exploration of emerging models. The codebase includes:

A **dataset module** that ingests raw data and prepares it for deep learning by standardizing images to have pixel/voxel values that range [0,1]. The PyTorch **Dataloader** class handles batching and exchanges between system memory and the graphics processing unit (GPU).

A **models module** that includes several NN architectures including the 2D and 3D Resnet and the VNet as well as the CNN discriminator and feature extractor models that serve as part of the GAN.

Configurable **training scripts** and **inference scripts** that instantiate the various models incorporate Weights and Biases (Biewald, 2020) to visualize training progress and ensure reproducibility.

For the volumetric version of the SRGAN, we used a VNet model trained to predict Equivalent Plastic Strain (EQPS) values in simulation data from prior Laboratory-Directed Research and

Development (LDRD) work related to the results described in (Johnson, et al., 2022) as the feature extractor model. In contrast to (Linder-Noren, 2018), we added additional intermediate feature map outputs from the pretrained model (all of the downsampling layer outputs) to the content loss to provide more information for training.

For the SRGAN generators, we extended an originally 2D ResNet architecture to 3D with full architecture details shown in the Appendix. Alternatively, we developed a VNet generator with layer details shown in Table 2. We used the same VNet architecture for the supervised training approach. Inspired by (Saharia, et al., 2021), when we used the VNet model as a generator, we first upsampled the low-resolution scan using the baseline Pytorch **upscale** method before ingestion into the model.

**Table 2 - VNet Generator architecture**

Layer Type	PyTorch Module	Parameters
Padding	ReplicationPad3d	padding=1
Convolution	Conv3d	in_channels=1, out_channels=8, kernel_size=(3, 3, 3), stride=(2, 2, 2)
Convolution	Conv3d	in_channels=8, out_channels=16, kernel_size=(3, 3, 3), stride=(2, 2, 2)
Convolution	Conv3d	in_channels=16, out_channels=32, kernel_size=(3, 3, 3), stride=(2, 2, 2)
Convolution	Conv3d	in_channels=32, out_channels=64, kernel_size=(3, 3, 3), stride=(2, 2, 2)
Deconvolution	ConvTranspose3d	in_channels=128, out_channels=64, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(2, 2, 2), output_padding=(1, 1, 1)
Deconvolution	ConvTranspose3d	in_channels=64, out_channels=32, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(2, 2, 2), output_padding=(1, 1, 1)
Deconvolution	ConvTranspose3d	in_channels=64, out_channels=256, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(1, 1, 1), output_padding=(1, 1, 1)
Deconvolution	ConvTranspose3d	in_channels=48, out_channels=128, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(1, 1, 1), output_padding=(1, 1, 1)
Deconvolution	ConvTranspose3d	in_channels=24, out_channels=32, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(2, 2, 2), output_padding=(1, 1, 1)
Convolution	Conv3d	in_channels=256, out_channels=32, kernel_size=(3, 3, 3), stride=(1, 1, 1)
Convolution	Conv3d	in_channels=32, out_channels=4, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1)
Convolution	Conv3d	in_channels=5, out_channels=1, kernel_size=(1, 1, 1), stride=(1, 1, 1)
Activation	Tanh	N/A

The discriminator model is a 9-layer CNN with full architecture details included in the Appendix.

Exploratory analysis was performed to tune the developed models with the Weights & Biases experiment tracker tool (Biewald, 2020). We set the number of training epochs for each model to

100,000 and we used early stopping with the mean absolute error of the validation example to select the best model for evaluation. For the SRGAN, we used the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.00001 for the generator model and 0.0001 for the discriminator, both with betas = (0.5, 0.999). For the VNet, we used the Adam optimizer with a learning rate of 0.0001 and betas = (0.9, 0.999). We ran all experiments on a 32 Gigabyte graphics processing unit (GPU) on an Nvidia DGX-2 machine. Due to GPU memory constraints, we downsample and resize both the low-resolution and high-resolution images to fit onto a single GPU, thereby constraining the problem to attempt an 2x increase in resolution in each of the 3 dimensions (total 8x) for each experiment.

### **4.3. Diode experiments**

For the TVS diode, we prepared 10 examples, and we used 8 for training, holding out one for validation that we used for early stopping during training to select the best model and one example completely held out for testing. We downsample and resize the low-resolution images to (12,48,24) for the SRGAN and (32,128,64) for the VNet and the high-resolution images to (48, 192, 96) for the SRGAN and (64,256,128) for the VNet.

### **4.4. Capacitor experiment**

For the capacitor, we prepared one example to demonstrate that our super-resolution framework is capable of learning to generate high resolution images of an alternate part. We downsample and resize the low-resolution images to (24,24,16) for the SRGAN and (48,48,32) for the VNet and the high-resolution images to (96,96,64) for the SRGAN and (192,192,128) for the VNet.

## 5. RESULTS

In this section, we report the results from each 3D super-resolution experiment. We found that the best model for super-resolution with respect to mean absolute error to the ground truth high-resolution scan quality is the supervised 3D VNet model. Both the SRGAN and the VNet approaches show promise but require more training data and a more powerful 3D feature extractor model to effectively generalize, as we observe overfitting from the VNet and insufficient content training for the SRGAN.

### 5.1. Diode

The diode experiments represent the full extent of the methods we have developed. We used a full training set and tested the generalizability of each model. Figure 13 shows an example of output from the VNet model used to generate a high-resolution slice through a diode from the training set and demonstrates the model’s ability to sharpen the image. Figure 14, in contrast, is an output example from the held-out test set. The blurriness of the edges of the diode is reduced, but the model’s error increases by approximately 4x. Full quantitative results for each method over the training set, validation example, and test example are reported in Table 3.

**Table 3 – Mean absolute error over diode sets with respect to ground truth high-resolution scans.**

Model	Training set	Validation example	Test example
Baseline interpolation	0.0561 +/- 0.0100	0.0570	0.0411
<b>3D VNet</b>	<b>0.0039 +/- 0.0004</b>	<b>0.0127</b>	<b>0.0172</b>
SRGAN (VNet generator)	0.0304 +/- 0.0012	0.0167	0.0423
SRGAN (ResNet generator)	0.0347 +/- 0.0009	0.0268	0.0416

# 3D Diode CT

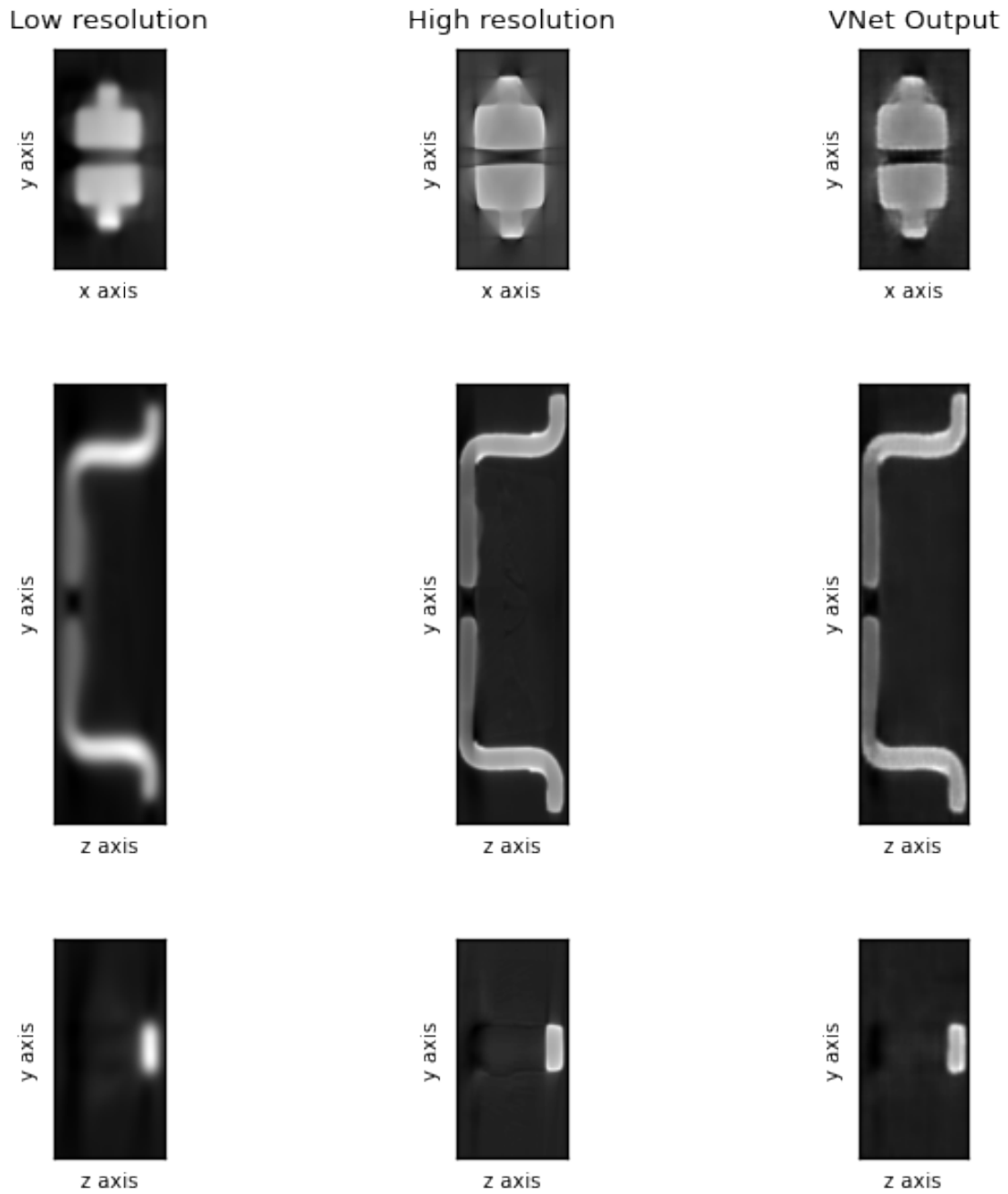


Figure 13 - Slice through a TVS diode 3D training example with the low-resolution scan (left), the ground truth high-resolution scan (center) and VNet output (right).

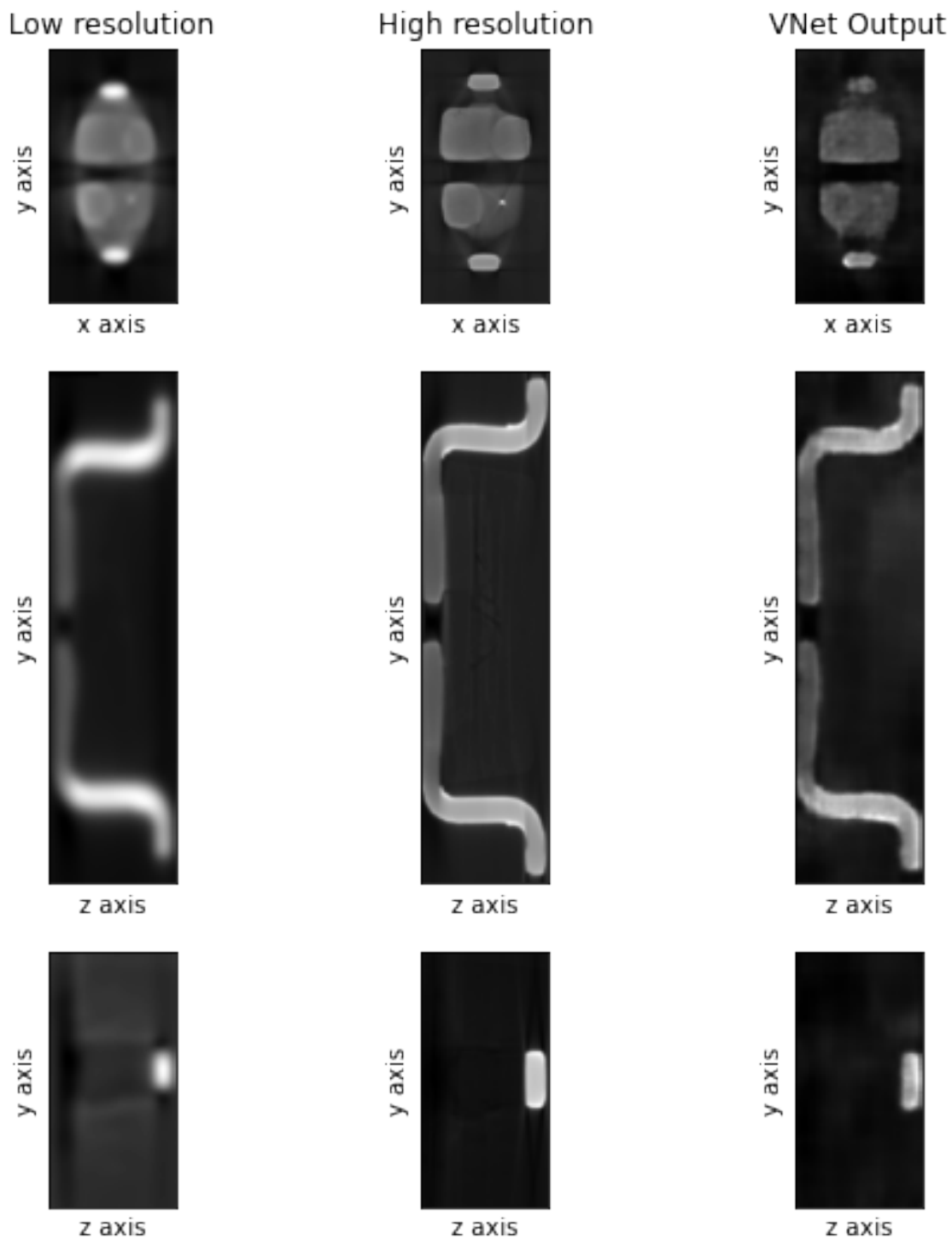
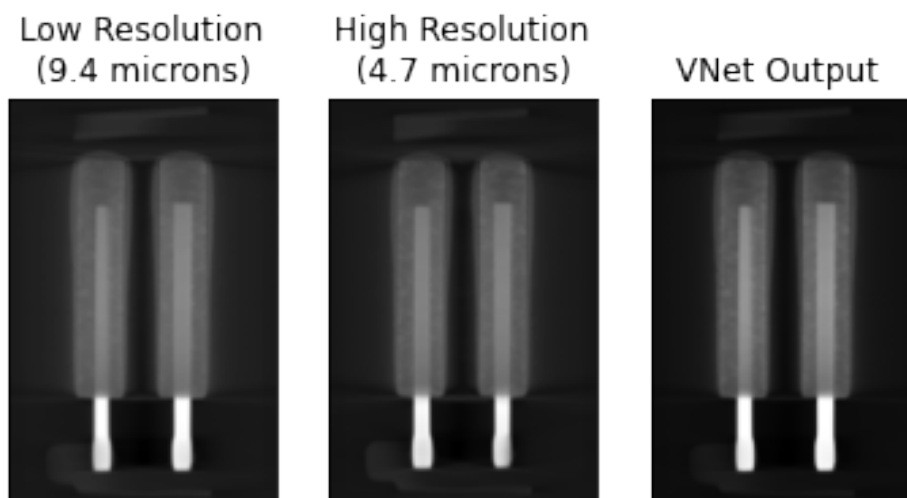


Figure 14 - Slice through a diode held-out test example with the low-resolution scan (left), the ground truth high-resolution scan (center) and VNet output (right). The VNet captures some of the high-resolution details but produces an image with approximately 4x error compared to the training set error.

## 5.2. Capacitor

Figure 15 shows a slice through the low-resolution, high-resolution, and VNet output for the capacitor example we used to train the model. Quantitatively, the mean absolute error of the VNet prediction when compared with the ground truth high resolution scan is 0.00737, but this was the sole training example in the set, so we expect no generalization to other capacitors without preparing additional training data.



**Figure 15 - Qualitative comparison of VNet fit to produce a high-resolution image of a capacitor. The difference between the low-resolution and high-resolution scans used for training is subtle, and we used only one capacitor example for both training and testing to demonstrate the feasibility of our workflow on an alternate electronic part.**

## 6. CONCLUSION

We have developed a modular, extensible 3D super-resolution framework. We have shown that our method is capable of improving upon the baseline method both qualitatively and quantitatively; however, we found that our model performs well within the training set but fails to generalize with quality consistent with the training set results. This is unsurprising, as the strength of the SRGAN algorithm lies in its ability to leverage features generated from a NN that was pretrained on millions of images. Advancement of this framework would benefit from study of both better 3D alignment procedures as well as more novel super resolution approaches. Better approaches for automatically cropping and aligning varying resolutions of XCT data would enable more facile configuration of already-collected datasets for inclusion in training sets across all component types investigated. Conversion of convolutional filters from this powerful pretrained NN from 2D to 3D as in (Merino, Azpiaz, Remazeilles, & Sierra, 2021) or to pretrain a VNet on many CT datasets such that the feature maps are rich enough to improve SRGAN generalization. Additionally, an alternative approach to generative modeling should be considered. Recently, denoising diffusion methods (Saharia, et al., 2021) have shown improvement over GAN models and could replace the GAN in the SRGAN architecture. Additionally, our prototype supports a fixed upscaling factor for each electronic part. Future work includes the implementation of software solutions that support variable scale super-resolution models.

This page left blank

## 7. REFERENCES

- Biewald, L. (2020). Experiment Tracking with Weights and Biases. <https://www.wandb.com/>.
- Clark, A. (2015). Pillow (PIL Fork) Documentation. <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition*, (pp. 248-255).
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, (pp. 2672-2680).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE conference on computer vision and pattern recognition*, (pp. 770-778).
- Johnson, K. J., Maestas, D., Emery, J. M., Grigoriu, M. D., Smith, M. D., & Martinez, C. (2022). Failure classification of porous additively manufactured parts using Deep Learning. *Computational Materials Science* 204, 111098.
- Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *ICLR poster*. arXiv preprint arXiv:1412.6980.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 541-551.
- Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., . . . Shi, W. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *IEEE conference on computer vision and pattern recognition*, (pp. 4681-1690).
- Linder-Noren, E. (2018). Retrieved from <https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/srgan>
- Merino, I., Azpiazu, J., Remazeilles, A., & Sierra, B. (2021). 3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts. *Sensors*, 21(4), 1078.
- Milletari, F., Navab, N., & Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE.
- OpenCV. (2022). Retrieved from OpenCV: <http://opencv.org/>
- Paszke, A., Gross, S., Chintala, S., & Chanan, G. (2017). Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration.
- Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., & Norouzi, M. (2021). Image Super-Resolution via Iterative Refinement. *arXiv preprint arXiv:2104.07636*.

This page left blank

## APPENDIX A. NN ARCHITECTURE DETAILS

Here, we provide implementation details for the models that comprise the SRGAN.

### A.1. ResNet Generator

```
GeneratorResNet(  
  (conv1): Sequential(  
    (0): Conv3d(1, 16, kernel_size=(9, 9, 9), stride=(1, 1, 1), padding=(4, 4, 4))  
    (1): PReLU(num_parameters=1)  
  )  
  (res_blocks): Sequential(  
    (0): ResidualBlock(  
      (conv_block): Sequential(  
        (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))  
        (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
        (2): PReLU(num_parameters=1)  
        (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))  
        (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
      )  
    )  
    (1): ResidualBlock(  
      (conv_block): Sequential(  
        (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))  
        (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
        (2): PReLU(num_parameters=1)  
        (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))  
        (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
      )  
    )  
    (2): ResidualBlock(  
      (conv_block): Sequential(  
        (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))  
        (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
        (2): PReLU(num_parameters=1)  
        (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))  
        (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)  
      )  
    )  
    (3): ResidualBlock(  
      (conv_block): Sequential(  
        (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
```

```

(1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
(2): PReLU(num_parameters=1)
(3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
)
(4): ResidualBlock(
(conv_block): Sequential(
(0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
(2): PReLU(num_parameters=1)
(3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
)
(5): ResidualBlock(
(conv_block): Sequential(
(0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
(2): PReLU(num_parameters=1)
(3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
)
(6): ResidualBlock(
(conv_block): Sequential(
(0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
(2): PReLU(num_parameters=1)
(3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
)
(7): ResidualBlock(
(conv_block): Sequential(
(0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
(2): PReLU(num_parameters=1)
(3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
)

```

```

)
)
(8): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(9): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(10): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(11): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(12): ResidualBlock(
  (conv_block): Sequential(

```

```

(0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
(2): PReLU(num_parameters=1)
(3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
)
(13): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(14): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(15): ResidualBlock(
  (conv_block): Sequential(
    (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
  )
)
)
(conv2): Sequential(
  (0): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
  (1): BatchNorm3d(16, eps=0.8, momentum=0.1, affine=True, track_running_stats=True)
)
(upsampling): Sequential(

```

```

(0): Conv3d(16, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(1): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(2): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(3): PixelShuffle(upscale_factor=2)
(4): PReLU(num_parameters=1)
(5): Conv3d(16, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(6): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(7): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(8): PixelShuffle(upscale_factor=2)
(9): PReLU(num_parameters=1)
)
(conv21): Conv3d(16, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(conv22): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(bn0): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(up0): Upsample(scale_factor=2.0, mode=nearest)
(conv23): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(conv24): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(bn1): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(up1): Upsample(scale_factor=2.0, mode=nearest)
(act0): PReLU(num_parameters=1)
(act1): PReLU(num_parameters=1)
(conv3): Sequential(
  (0): Conv3d(64, 1, kernel_size=(1, 1, 1), stride=(1, 1, 1))
  (1): Tanh()
)
)
)

```

## A.2. Discriminator

```

Discriminator(
(model): Sequential(
  (0): Conv3d(1, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
  (1): LeakyReLU(negative_slope=0.2, inplace=True)
  (2): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(1, 1, 1))
  (3): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (4): LeakyReLU(negative_slope=0.2, inplace=True)
  (5): Conv3d(64, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
  (6): BatchNorm3d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): LeakyReLU(negative_slope=0.2, inplace=True)
  (8): Conv3d(128, 128, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(1, 1, 1))
  (9): BatchNorm3d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
)

```

```

(10): LeakyReLU(negative_slope=0.2, inplace=True)
(11): Conv3d(128, 256, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(12): BatchNorm3d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(13): LeakyReLU(negative_slope=0.2, inplace=True)
(14): Conv3d(256, 256, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(1, 1, 1))
(15): BatchNorm3d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(16): LeakyReLU(negative_slope=0.2, inplace=True)
(17): Conv3d(256, 512, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
(18): BatchNorm3d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(19): LeakyReLU(negative_slope=0.2, inplace=True)
(20): Conv3d(512, 512, kernel_size=(3, 3, 3), stride=(2, 2, 2), padding=(1, 1, 1))
(21): BatchNorm3d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(22): LeakyReLU(negative_slope=0.2, inplace=True)
(23): Conv3d(512, 1, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
)
)

```

### A.3. Feature Extractor

```

FeatureExtractor(
  (feature_extractor): VNet(
    (padding): ReplicationPad3d((1, 1, 1, 1, 1, 1))
    (conv0): Conv3d(1, 64, kernel_size=(3, 3, 3), stride=(2, 2, 2))
    (conv1): Conv3d(64, 128, kernel_size=(3, 3, 3), stride=(2, 2, 2))
    (conv2): Conv3d(128, 256, kernel_size=(3, 3, 3), stride=(2, 2, 2))
    (conv3): Conv3d(256, 512, kernel_size=(3, 3, 3), stride=(2, 2, 2))
    (activation): SELU()
  )
)

```

## DISTRIBUTION

### Email—Internal

Name	Org.	Sandia Email Address
Sylvain Bernard	1424	srberna@sandia.gov
Ryan Haggerty	1850	rphagge@sandia.gov
Philip Noel	1851	pnoell@sandia.gov
Andrew Polonsky	1851	apolon@sandia.gov
Coby Davis	1854	cldavis@sandia.gov
Catherine Appleby	5575	caapple@sandia.gov
Carianne Martinez	5575	cmarti5@sandia.gov
Steve Wix	7641	sdwix@sandia.gov
Alex Robinson	7643	arobins@sandia.gov
Tiffany Davie	7645	tjdavie@sandia.gov
Tony Isenberg	7645	taisenb@sandia.gov
Technical Library	1911	<a href="mailto:sanddocs@sandia.gov">sanddocs@sandia.gov</a>

### Email—External

Name	Company Email Address	Company Name
Matt McKown	<a href="mailto:mmckown@kcncsc.doe.gov">mmckown@kcncsc.doe.gov</a>	Kansas City National Security Campus

This page left blank

This page left blank



**Sandia  
National  
Laboratories**

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.