MACHINE LEARNING FOR DEFECT DETECTION FOR PBFAM USING HIGH RESOLUTION LAYERWISE IMAGING COUPLED WITH POST-BUILD CT SCANS

Jan Petrich[1], Christian Gobert, Shashi Phoha, Abdalla R. Nassar, and Edward W. Reutzel
Applied Research Laboratory at the Pennsylvania State University, University Park, PA 16804

Abstract

This paper develops a methodology based on machine learning to detect defects *during* Powder Bed Fusion Additive Manufacturing (PBFAM) processes using data from high resolution images. The methodology is validated experimentally using both a support vector machine (SVM) and a neural network (NN) for binary classification. High resolution images are collected each layer of the build, and the ground truth labels necessary for supervised machine learning are obtained from a 3D computed tomography (CT) scan. CT data is processed using image processing tools—extended to 3D—in order to extract *xyz* position of voids within the component. Anomaly locations are subsequently transferred from the CT domain into the image domain using an affine transformation. Multi-dimensional features are extracted from the images using data surrounding both anomaly and nominal locations. Using cross-validation strategies for machine learning and testing, accuracies of close to *90%* could be achieved when using a neural network for in-situ anomaly detection.

1. Introduction

This investigation extends initial work conducted in [1][2] which was built upon the hypothesis that visual process sensors monitoring a PBFAM process, specifically high resolution *in situ* imaging of build surfaces, can capture visible features on build surfaces of individual layers that can be linked to discontinuities or indications of discontinuities in the resultant component. Discontinuities can be powder-induced or process-induced and commonly arise as a result of part solidification, from process errors in a nominal build environment or whenever process parameters deviate too much from their nominal settings [3]. Additionally, porosities can form from entrapped gas pores, appearing spherical in shape and with diameters in the order of ten of microns, or from elongated voids, arising from a lack of fusion between powder particles that can extend to several hundred microns in length [4].

In this work, ground truth regarding the presence and exact location of discontinuities is established using an automated discontinuity detection methodology for CT scans developed by PSU/ARL. As a result, each three-dimensional (3D) neighborhood is labeled as either anomalous or nominal. Given the known part geometry and the introduction of unique reference points, the 3D locations of anomalous and nominal neighborhoods can then be transferred from the CT domain into the *in situ* imaging domain, i.e. the stack of images collected during the build process. Using the associated labels “anomalous” and “nominal”, supervised machine learning can be employed in order to attempt and verify the detection of flaws using only layerwise images of the build surface. A high-level schematic of the overall process is shown in Figure 1.

[1] Corresponding author, jup37@arl.psu.edu, (814)-863-2493

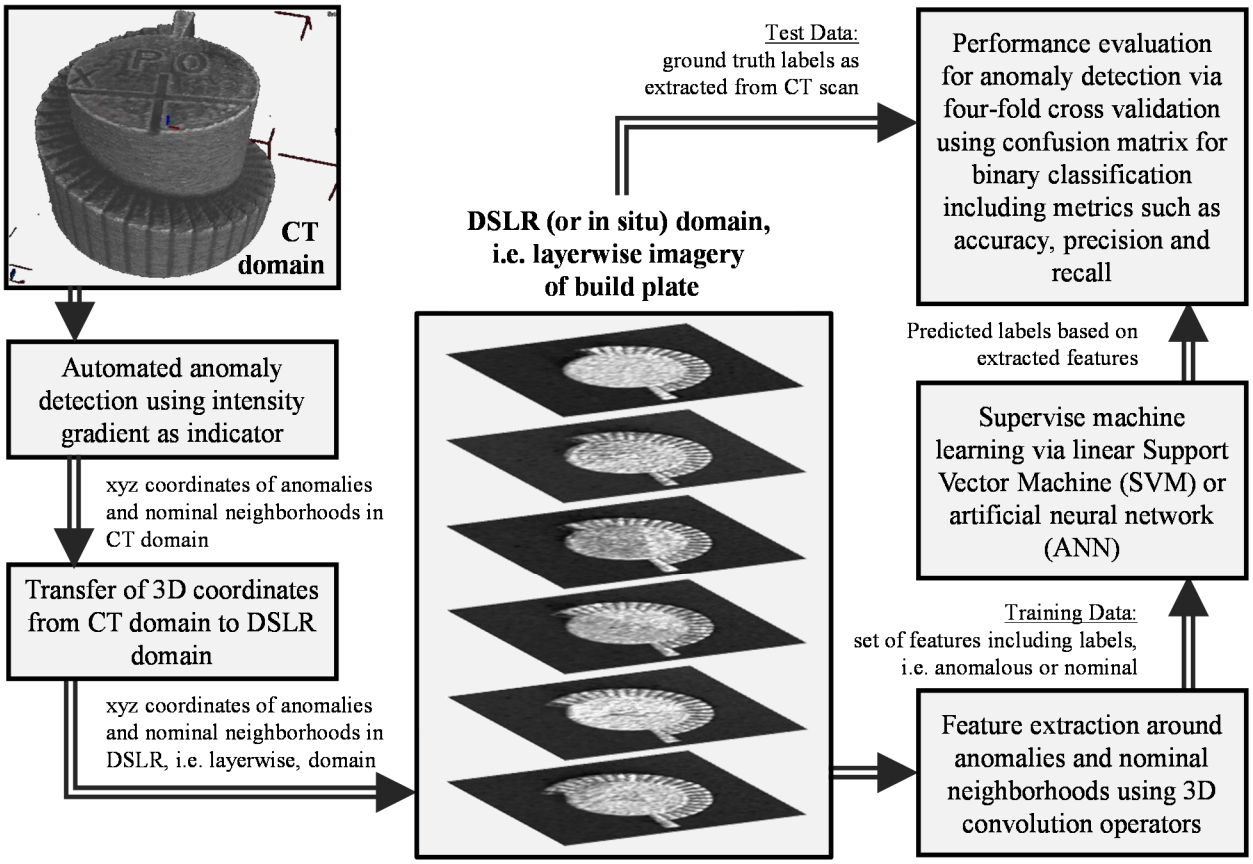

Figure 1: High-level process schematic showing (i) extraction of anomalies from CT scan including transfer of coordinates to the DSLR domain on the left, (ii) in situ, layerwise imagery at the center, and (iii) feature extraction, supervised machine learning and performance evaluation on the right.

Specific contributions of this work and extension to prior work presented in [1][2] include the development of:

1. an updated anomaly detection algorithm for CT scans using separable convolution operators that are highly parallelizable and thus provide significantly shorter execution times when deployed on hardware accelerated platforms
2. a streamlined clustering algorithm that groups adjacent, anomalous CT voxels into anomaly clusters without pre-defining the resulting number of clusters,
3. an updated feature extraction methodology for layerwise imagery that extracts gradients rather than absolute grayscale intensity, thereby providing additional robustness to varying lighting conditions, and
4. an extension of supervised machine learning concepts from a linear SVM toward a neural network architecture

The paper is organized as follows. Section 2 describes the experimental setup for this work including the data collection process. Because anomalies are not detectable for a human observer within the layerwise imagery collected during this build process, post-build CT scans are utilized to detect anomalies and infer their xyz location in the part. This process is outlined in Section 3. Section 4 discusses the transfer of coordinates from the CT domain to the layerwise image domain. Feature extraction in the layerwise image domain is described in Section 5. Section 6 outlines machine learning and anomaly classification via linear SVM and NN and presents preliminary results. Section 7 concludes the paper and outlines future work.

2. Experimental Setup

The PBFAM test build process was conducted in an EOS M280 AM system [5] and monitored by a 36.3-megapixel digital single-lens reflex (DSLR) CCD camera (Nikon D800E) mounted inside the build chamber. The DSLR camera captured a total of eight images of the build platform for each build layer using five light sources, with each different lighting condition designated as a flash module. The set-up including the test part used for this investigation, a step cylinder, are shown in Figure 2. Each number next to a light source indicates the designated flash module, with some light sources employed both after the powder recoating operation and again after the laser fusion step. The variation in lighting angle and characteristics enhances contrast in different ways, and reveals additional information that could not be provided by a single lighting source alone. Timing of image capture was triggered via proximity-sensor monitoring of the recoater blade, with images (1) - (3) captured immediately following the powder recoating operation, images (4) - (8) were captured immediately following the laser fusion step. Additional details about the camera and light system setup in the EOS M280 AM system were reported by the PSU/ARL team in [6].

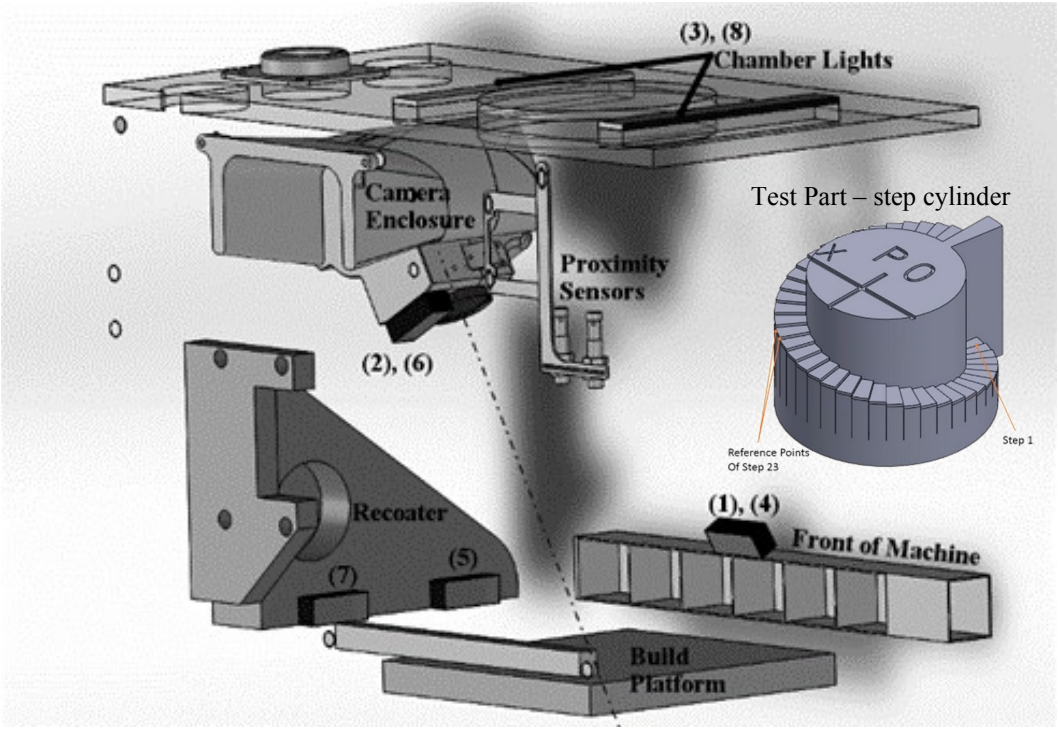

Figure 2: Location of camera system and light sources within the build chamber. (1) - (8) sequence and locations of flash modules. The test part, here a step cylinder, is also shown on the right.

A single stainless steel part, denoted as a step cylinder and shown on the right in Figure 2, was built with EOS Stainless Steel GP-1 powder using the EOS standard exposure parameters and processing strategy for 20 µm layers. The 10 mm tall × Ø10 mm cylinder was encircled by a 39-step staircase starting near the base and ending at the top of the part, with a step height of 200 µm (10 layers). The small size of the test part was chosen to enable high resolution in the post-build CT scans (i.e. a larger test part would limit the scanning resolution). The 39-step staircase was incorporated into the design to provide unique reference points (i.e. corners of a staircase) in order

to estimate an affine transformation that enables the transfer of 3D coordinates from the CT domain into the DSLR (or in situ) layerwise domain. Then, neighborhoods of interest can be identified in the DSLR layerwise domain and properly labeled, e.g. anomalous or nominal, allowing for supervised machine learning.

DSLR camera images of the entire build plate were cropped to a 400 x 400 pixel image centered on the step cylinder and RGB channels were combined to produce grayscale representations with 8 bit intensity values. For each flash module, DSLR images were stacked to create a 3D representation of the build process. Voxels in the DLSR domain were defined based upon image resolution and build layer height (~50 μm/pixel xy image resolution, 20 μm/layer). For comparison, CT scan resolution is 15 μm/voxel in x, y and z direction. Then, the grayscale intensity I_{DSLR} is a function of the voxel location x_I, y_I, z_I in the image stack as row, column, and layer as well as the flash module index for each image $f_I \in \{1, 2, .., 8\}$. We find

$$I_{DSLR}(x_I, y_I, z_I, f_I) \in [0,255] \tag{1}$$

In this investigation the dimensions of the domain, denoted as the DSLR domain, was [400, 400, 555, 8]. Further processing of the DSLR data primarily for the purpose of feature extraction is detailed later where appropriate. Figure 3 displays one layer image for each of the eight flash modules taken during a single layer of the step cylinder build.

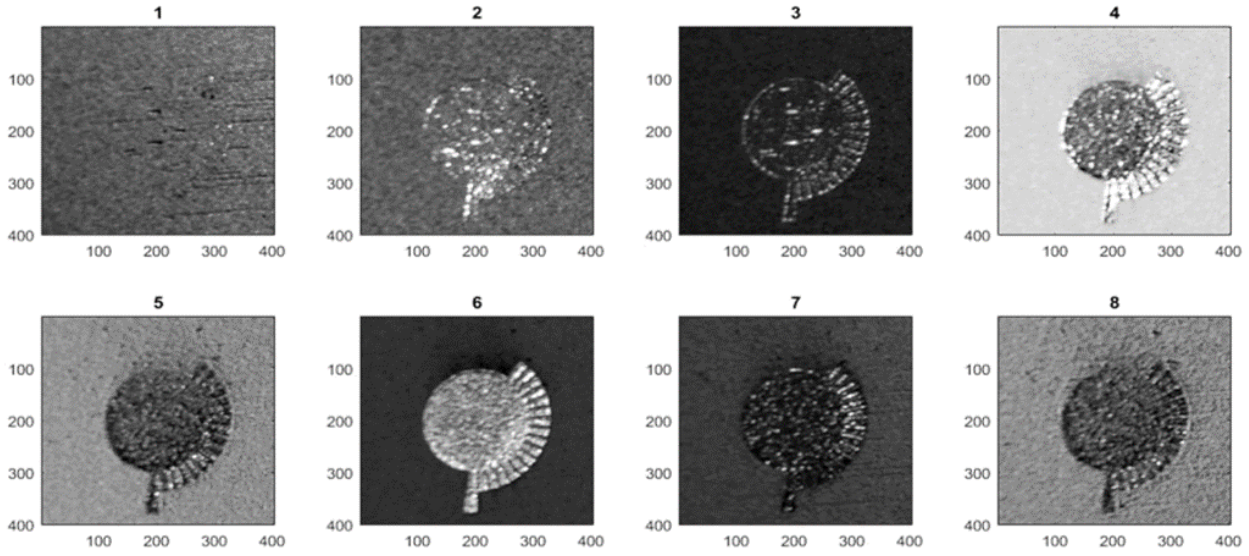

Figure 3: Eight flash module images of step 23: (1)-(3) post powder recoating flash modules and (4)-(8) post fusion flash modules.

3. Detection of Anomalies in CT Scan

A post-build CT scan, shown in the upper left in Figure 1, was obtained to create a 3D representation of the part. Voxels in the CT images are discretized at 15 μm/voxel resolution with 8 bit intensity values ranging from 0 to 255. In the CT domain, gray scale intensity is a function of voxel locations x_{CT}, y_{CT} and z_{CT} representing row, column, and layer, i.e.

$$I_{CT}(x_{CT}, y_{CT}, z_{CT}) \in [0,255] \tag{2}$$

For this experiment, the complete CT domain dimensions for the step cylinder were [1009, 1103, 715] (or ~800 million voxels).

Discontinuities (or anomaly voxels) in a part manifest themselves in CT scan images through noticeable spatial intensity gradients, i.e. irregular intensity values as compared to their surroundings. Anomaly CT voxels can be divided into two classes: (i) voids or pores which are low-intensity (low density) CT voxels surrounded by higher intensity (higher density) CT voxels, and (ii) super-densities (perhaps as the result of powder contamination) which are high-intensity (high density) CT voxels surrounded by lower intensity (lower density) CT voxels. Nominal CT voxels, characterized by discontinuity and indication-free regions of a test part, are observed as CT voxels that are surrounded by similar intensity CT voxels, thereby generating a near zero intensity gradient.

In order to detect intensity gradients in any direction, a digital 3D Gaussian kernel was constructed and convolved with the CT image stack. Over a normalized 3D domain with $x_k, y_k, z_k \in [-1, +1]$, we select a kernel of the form

$$K_n(x_k, y_k, z_k) = \exp\left(-\frac{x_k^2 + y_k^2 + z_k^2}{\sigma^2}\right) - \gamma_0 \tag{3}$$

with standard deviation $\sigma = 0.5$ and offset parameter $\gamma_0 \in \mathbb{R}$ to yield a zero mean kernel, i.e.

$$0 = \iiint_{-1}^{+1} K_n(x_k, y_k, z_k)\, dx_k\, dy_k\, dz_k \tag{4}$$

As compared to [1][2] in which a Gabor wavelet was used, a Gaussian kernel allows for a separation of dimensions during the convolution, which significantly reduces the computational complexity of the convolution operation. In fact, it can be shown that any 3D convolution using the Gaussian kernel (3) can be separated using the three 1D kernels

$$\begin{aligned} k_x(:,1,1) &= \exp(-x_k^2 / \sigma^2) \\ k_y(1,:,1) &= \exp(-y_k^2 / \sigma^2) \\ k_z(1,1,:) &= \exp(-z_k^2 / \sigma^2) \end{aligned} \tag{5}$$

representing 1D tensors along different spatial directions. Then, for any CT image I_{CT}, we find the convolution response to be separable into

$$\begin{aligned} I_{CT} * K_n(x_k, y_k, z_k) = {} & I_{CT} * k_x(:,1,1) * k_y(1,:,1) * k_z(1,1,:) \\ & - \gamma_0\, I_{CT} * 1_x(:,1,1) * 1_y(1,:,1) * 1_z(1,1,:) \end{aligned} \tag{6}$$

where * denotes the convolution operator. Here, $1_x(:,1,1)$, $1_y(1,:,1)$ and $1_z(1,1,:)$ are all-ones 1D tensors in row, column and layer direction, respectively, and of dimensions equal to those of $k_x(:,1,1)$, $k_y(1,:,1)$ and $k_z(1,1,:)$. Note, that due to the zero mean requirement and the addition of the γ_0 parameter in (3), the 3D convolution between CT image I_{CT} and kernel $K_n(x_k, y_k, z_k)$, is broken down into six 1D convolutions in (6).

The actual size of the normalized kernel (3) can be scaled in order to extract anomalies of different physical sizes within the CT imagery. Therefore, the individual domains for x_k, y_k and z_k will be discretized and subdivided into the appropriate number of CT voxels n_k, which then defines the physical size of the 3D Gaussian kernel. The left plot in Figure 4 shows a sliced view of such a kernel for $n_k = 17$. Given a CT resolution of $15\mu m$ per voxel, this kernel size corresponds to a volume of $(255\mu m)^3$. The right side of Figure 4 depicts the Gaussian envelop in 1D, which also illustrates the approximate size of the anomaly that would be extracted by virtue of 3D image convolution. The lower plot shows the approximate size of extracted anomalies when using kernel sizes 3 to 31.

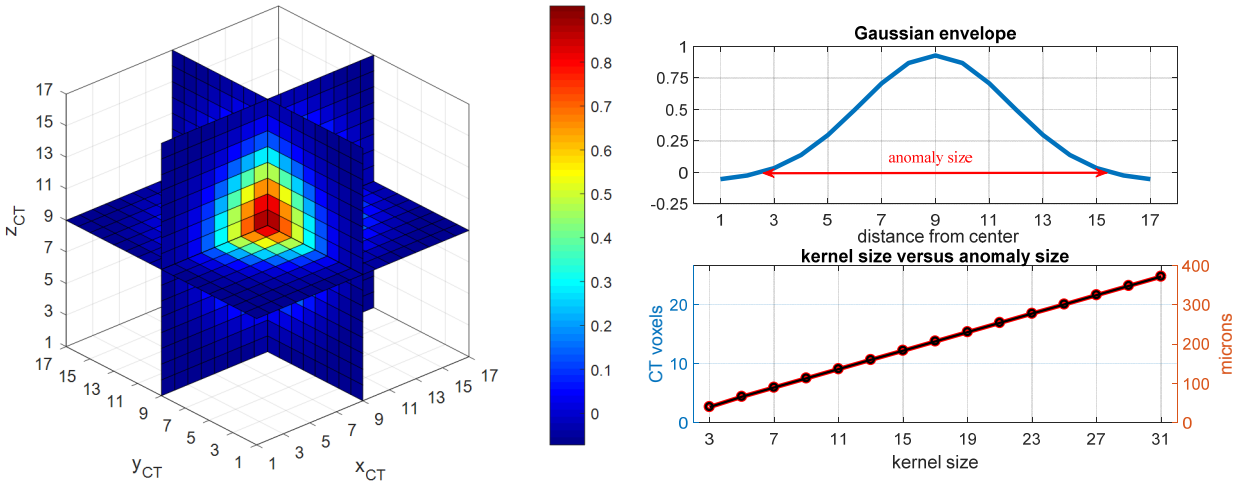

Figure 4: Left - Sliced view of 3D Gaussian kernel of size $n_k = 17$ used for anomaly detection in CT. Right – Gaussian envelope in 1D (top) and approximate anomaly size versus kernel size (bottom).

Given a kernel size n_k, the computational requirements to generate the convolution response reduce from n_k^3 multiplications per voxel in the CT image to $6n_k$ multiplications per voxel. Additionally, the values for minuend and subtrahend in (6) can be computed in parallel. For the CT image used in this work, which consist of ~800 million voxels, the time to generate a convolution response for a 3D Gaussian kernel with size $n_k = 17$ was reduced from *155sec* (*1200sec* when using a CPU only) to *35sec* using a Tesla C2070 GPU. This reduction in processing time further allows the user to generate convolution responses for multiple kernel sizes in a reasonable amount of time, thereby allowing for the extraction of anomalies of various sizes as illustrated in the bottom right in Figure 4.

For ***anomaly*** detection in CT, convolution responses for eight kernel sizes from $n_k = 3$ to $n_k = 17$ in steps of 2 were generated. A voxel is labeled anomalous if its convolution response exceeds a threshold of t_A standard deviations from the mean response for *any* of the eight convolution responses. Typical values for t_A range from 5 to 10, which allows the user to manually increase/decrease the number of detected anomalies. Note that max values within the convolution response will be associated with super-densities, while min value (for which the convolution response is below t_A standard deviations from the mean) will be associated with voids or pores.

A voxel is labeled ***nominal*** if its convolution response stays within a threshold of t_N standard deviations from the mean response for *all* of the eight convolution responses. Typical

values for t_N range from 0.1 to 0.5. Again, both t_A and t_N, with $t_A > t_N$ are design parameters dictating the amount of voxels being extracted as either anomalous or nominal.

Locations of anomalies and nominal voxels are subsequently stored in matrix form using $A_{A+} \in \mathbb{R}^{n_A \times 3}$ for n_A anomalies, i.e. super-densities and voids, and $A_N \in \mathbb{R}^{n_N \times 3}$ for n_N nominal CT voxels. After extraction, anomaly voxels are clustered by simply assigning two adjacent anomaly voxels, i.e. voxels of same type and within a distance of d_c voxels from one another, to the same cluster. As compared to [1][2], an iterative search for finding the number of resulting clusters is no longer necessary. An example of a resulting cluster consisting of 669 individual CT voxels is shown in Figure 5. In addition to the cluster configuration on the left, the respective CT scan slices for XY, YZ and XZ planes are shown on the right. Those 40x40 slices correspond to an area of 600µm x 600µm. The apparent grayscale intensity gradient which shows low intensity voxels surrounded by high intensity voxels confirms this anomaly as a 'void'.

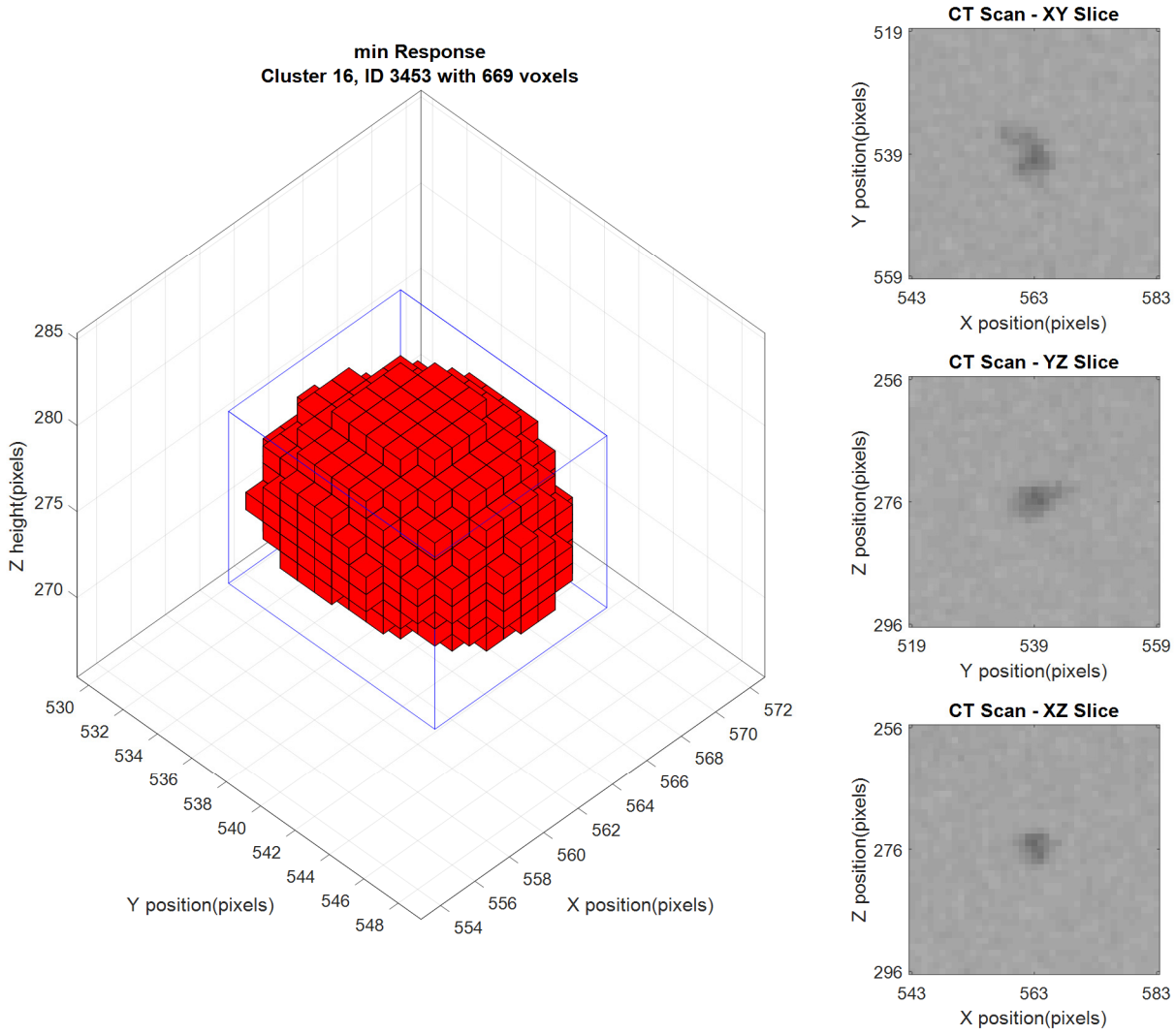

Figure 5: Extracted anomaly cluster in CT domain showing voxel configuration on the left and CT scan slices for XY, YZ, and XZ plane on the right.

4. Mapping Anomalies into DSLR Domain

The transformation of CT anomaly coordinates into DLSR coordinates is based on the work in [1][2]. Given the unique geometry of the step cylinder utilized in this work, distinct reference points, here two corners per step, are extracted in the CT domain as well as the DSLR domain. Using a linear least squares approach, an affine transformation defined by a matrix $T \in \mathbb{R}^{3\times3}$ and a bias vector $b \in \mathbb{R}^{3\times1}$ can be estimated between the two sets of reference points. Then the corresponding coordinates in the DSRL domain denoted $\tilde{B}_A \in \mathbb{R}^{n_A\times3}$ and $\tilde{B}_N \in \mathbb{R}^{n_N\times3}$ are computed via

$$\begin{aligned} \tilde{B}_A^T &= TA_A^T + b\ 1_{n_A} \\ \tilde{B}_N^T &= TA_N^T + b\ 1_{n_N} \end{aligned} \tag{7}$$

Here, 1_{n_A} and 1_{n_N} denote all-ones row vectors of dimensions indicated by the subscript. DSLR reference points as well as CT reference points projected into the DSLR domain are shown on the left in Figure 6. Only a small residual error is noticeable after transformation. Residual errors in the generated coordinate transformation can arise due to the non-exact extraction of reference points, distortion of the component in a nonlinear, non-affine manner during and after the build, or by fluctuations in the melt pool that remain during and after solidification. Therefore, the residual transformation error dictates the size of the DSLR neighborhood in which to expect the anomaly. Consequently feature extraction in the DSLR domain must entail a large enough neighborhood, or convolution kernel, in order to guarantee that the actual anomaly as indicated by the CT scan was captured.

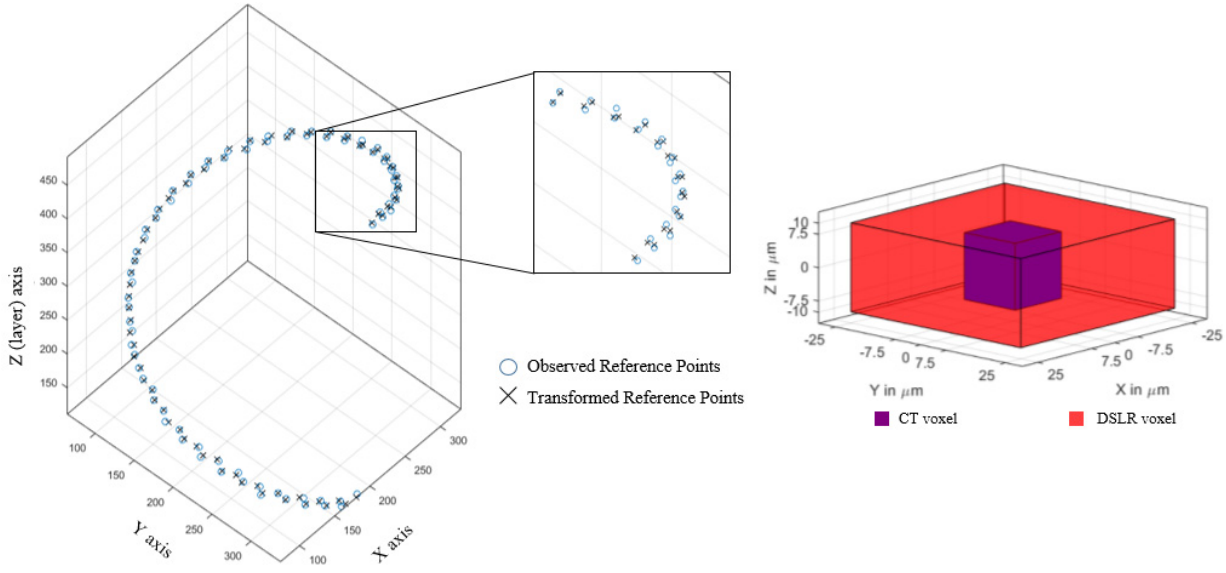

Figure 6: Left - DSLR and transformed CT reference points showing small residual, transformation error. Right - Size comparison showing high resolution CT voxel (blue) and lower resolution DSLR voxel (red).

The right side of Figure 6 depicts the difference in size between the high resolution CT voxels (blue - $15\mu m \times 15\mu m \times 15\mu m$) and the lower resolution DSLR voxels (red - $50\mu m \times 50\mu m \times 20\mu m$). This mismatch in resolution causes a down-sampling between CT and DSLR domains. In other words, several CT voxels will be mapped into the same DSLR voxel, thereby

reducing the number of unique coordinates for anomalies as well as nominal build conditions. Note that the ratio of volumes between DSLR voxels and CT voxels is approximately 15:1. Consequently, duplicate entries in $\tilde{B}_A$ and $\tilde{B}_N$ are removed to generate sets of unique DSLR coordinates, which are then re-captured by the matrices $B_A \in \mathbb{R}^{m_A \times 3}$ and $B_N \in \mathbb{R}^{m_N \times 3}$. Note that, due to down-sampling, the number of entries (or rows) in both matrices reduces to $m_A < n_A$ and $m_N < n_N$, respectively.

5. Feature Extraction in DSLR Domain

As discussed in the previous Section, an error in the coordinate transformation from CT domain into DSLR domain may be present and thus ground truth labels may be offset slightly in the DSLR domain without accountability or certainty. On one hand, narrow feature extraction in the DSLR domain would potentially miss voxels of interest, and discontinuity detection would not be successful. On the other hand, a large feature extraction filter would increase the overall confidence in capturing the voxels of interest, while simultaneously expanding the dimensionality of the underlying classification problem. The latter not only requires additional computational bandwidth for training but also presents concern for overfitting. Therefore and as described in [1], the size of the feature extraction filter is directly based on the residual error of the coordinate transformation. In [1], the root-mean-square error (RMSE) in *x*, *y* and *z* direction was ~1.75, ~1.5, and ~0.75 DSLR voxels, respectively, corresponding to 87.5, 75, and 37.5 µm. Then, a feature extraction filter of size 7 x 7 x 3 DSLR voxels provides a 95% confidence (corresponding to ±2 standard deviations) that the extracted CT anomaly including label is properly captured in the DSLR neighborhood spanned by the filter.

Figure 7 depicts an outline of the feature extraction filter (or 3D convolution kernel) expanding through three layers of the DSLR domain. An illustration of the 3D convolution over the stack of layerwise DSLR images is shown on the left and over actual DSLR imagery on the right. The red outline indicates the filter, 7 x 7 x 3 DSLR voxels (350 x 350 x 60 µm), which is centered on the labeled DSLR voxel.

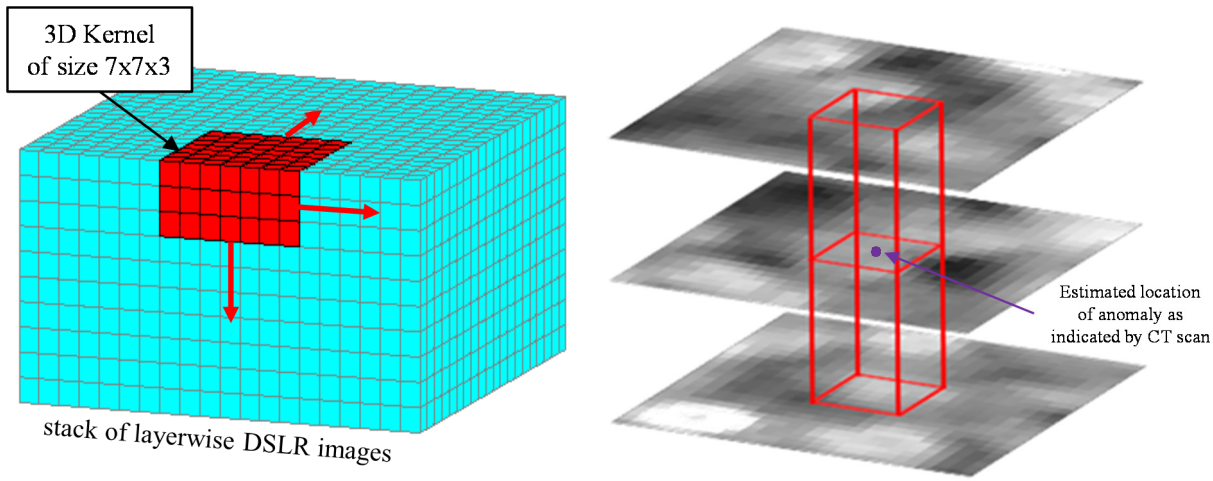

Figure 7: Left – Illustration of 3D convolution using 7x7x3 filter over stack of layerwise DSLR images. Right - Outline of feature extraction filter over a ground truth anomaly in DSLR domain.

Given any filter size (x_f, y_f, z_f), $n_f = x_f y_f z_f$ linearly-independent filters for feature extraction can be created, where one filter extracts one feature. In [1], this was done by setting

only one filter element to 1 at a time, while the remaining $n_f - 1$ elements in a filter are set to zero. For the selected filter size, we find $n_f = 7\ x\ 7\ x\ 3 = 147$ and each of the 147 filters then extracts an intensity value from a DSLR voxel as one feature dimension. The entire set of filters then extract the neighborhood of DSLR voxel grayscale intensities surrounding the estimated DSLR coordinate of the labeled CT voxel. Any additional filter constructed would be a linear combination of the first n_f filters and thus contain no additional information for linear machine learning classification.

In order to relax the reliance on absolute intensity, which may be fluctuating as the process evolves and lighting conditions vary, and to enforce anomaly detection based on intensity gradient information, all filters have been modified in this study to reflect zero mean values. In other words, all filter elements are offset by $-1/147$ such that for each filter the sum of all filter elements is zero.

Via 3D convolution and as illustrated on the left of Figure 7, an n_f-dimensional feature vector is generated for each voxel of interest, i.e. each voxel that has previously been labeled 'anomalous' or 'nominal'. In the following, we denote the feature vector as $X_{k,i} \in \mathbb{R}^{n_f}$ where i indicates the incremental index of the voxel (or data sample) and k represents the flash module, see Figure 3, from which the feature vector was extracted. The same sized feature extraction filter is used in each of the eight individual DSLR flash module domains.

6. Machine Learning and Anomaly Detection

This section outlines a linear support vector machine (SVM) and neural network (NN) architecture for in situ anomaly detection. Both architecture will be outlined and formally compared. Following feature extraction in the DSLR domain discussed in Section 5, a feature matrix is generated for each flash module $k = 1,..,8$ containing n_s data samples (or voxels of interest). Here, n_s denotes the total number of 3D neighborhoods, anomalous and nominal, and each data sample contains n_f features extracted from the specific 3D neighborhood. The feature matrix for flash module k is then given by

$$X_k = [X_{k,1} \quad X_{k,2} \quad \cdots \quad X_{k,n_s}] \in \mathbb{R}^{n_f \times n_s} \tag{8}$$

where $X_{k,i} \in \mathbb{R}^{n_f}, i = 1..n_s$, i.e. the columns of X_k, are the individual, n_f-dimensional feature vectors. The associated vector of labels is denoted $Y \in \{0,1\}^{n_s}$, with elements $Y_i = 1$ or $Y_i = 0$ denoting that data sample $X_{k,i}$ describes an anomaly or nominal condition, respectively.

In the following, Section 6.1 discusses and compares in-situ anomaly detection via linear Support Vector Machine (SVM) and Neural Network (NN) classifier using *single* flash modules only. Anomaly detection using fused information from *multiple* flash modules is presented in Section 6.2.

6.1 Anomaly Detection using Single Flash Module

In this section, anomaly detection is carried out using only single flash imagery information X_k. Sections 6.1.1 and 6.1.2 introduce classification architectures using the linear SVM and the

NN approach, respectively. Overall classification results including a formal comparison between both approaches using standard classification performance metrics is presented in Section 6.1.3.

6.1.1 Linear SVM Architecture using Single Flash Module

SVMs are discriminative classifiers that utilize a hyperplane to separate data samples with different labels within the feature space [7]. For an n_f-dimensional feature space with $x \in \mathbb{R}^{n_f}$, a hyperplane is defined by

$$\beta^T x - \beta_0 = 0 \tag{9}$$

where $\beta \in \mathbb{R}^{n_f}$ is the hyperplane's normal vector, and $\beta_0 \in \mathbb{R}$ is a bias parameter. As an example, Figure 8 shows such a hyperplane in two dimensions separating two classes of data points. The "best" separating hyperplane is defined by a subset of data points, called support vectors, and provides an optimal solution for a user-defined trade-off between in-sample classification accuracy and a wide margin of separation between the two classes. The margin of separation given by $d = 2/\|\beta\|$. The trade-off between minimizing in-sample misclassifications and maximizing the margin of separation is commonly resolved via soft-margin classification [7].

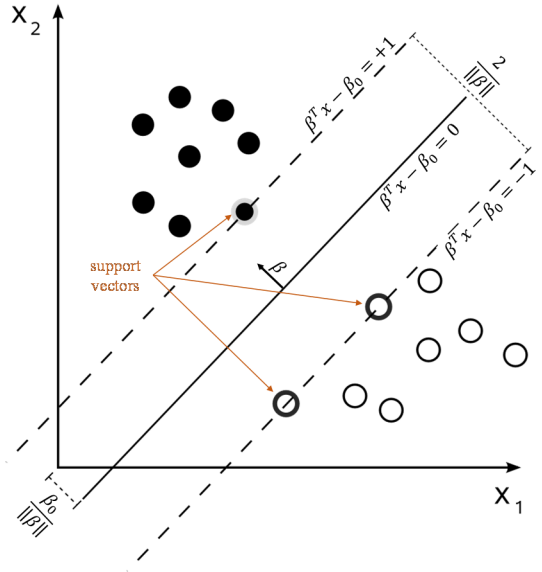

Figure 8: Example of separating hyperplane in two dimensions (original source Wikipedia).

With the feature matrix from (8), one can construct a linear SVM classifier for each flash module k that predicts labels $\tilde{Y}_{k,i}$ via

$$\begin{aligned} \tilde{Y}_{k,i} &= 1 \quad \text{if} \quad \beta_k X_{k,i} - \beta_{k,0} > 0 \\ \tilde{Y}_{k,i} &= 0 \quad \text{if} \quad \beta_k X_{k,i} - \beta_{k,0} \leq 0 \end{aligned} \tag{10}$$

Four-fold cross validation is used to find the hyperplane parameters β_k and $\beta_{k,0}$ that provide the best in-sample classification accuracy between true labels Y_i and predicted labels $\tilde{Y}_{k,i}$ while also maximizing the margin of separation between both classes of data points. Note that predictions for the sample i may differ between flash modules used, and therefore $\tilde{Y}_{k,i}$ is a function of k while the true label Y_i is not. During four-fold cross validation, one classifier is trained on 3/4 of the data and tested for out-of-sample performance on the remaining, unseen 1/4 of the data. Numerical SVM algorithms and software libraries are readily available, and MATLAB's *fitcsvm*

function was utilized for this work. Classification results for all flash modules k are discussed in Section 6.1.3.

6.1.2 Neural Network Architecture using Single Flash Module

The neural network architecture utilized for machine learning and anomaly detection follows the layout of a fully connected two-layer perceptron. It is shown on the left in Figure 9. The feature vectors $X_{k,i}$ constitutes the input to the neural network, while the output is given by the corresponding prediction $\hat{Y}_{k,i} \in [0,1]$.

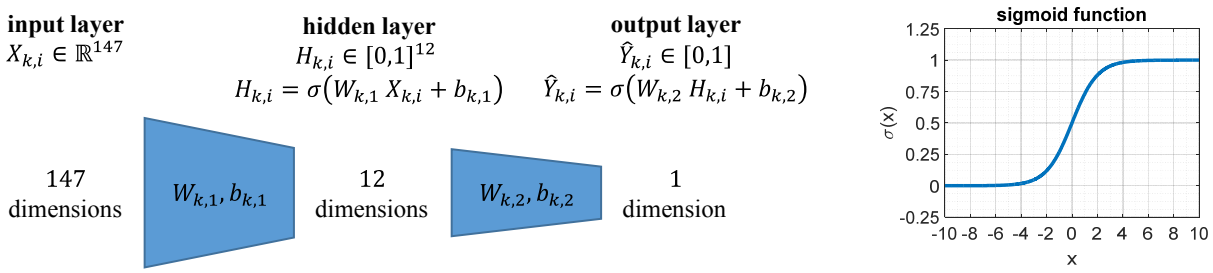

Figure 9: Left - Neural network architecture, i.e. fully connected two-layer perceptron, for single flash module k, mapping the feature vector $X_{k,i}$ into corresponding prediction $\hat{Y}_{k,i} \in [0, 1]$. Right - Sigmoid activation function σ for neural network.

The selected architecture utilizes a single hidden layer with states $H_{k,i} \in [0,1]^{12}$. The dimension of $H_{k,i}$ is a design parameter trading computational complexity for overall fidelity of the underlying data representation. For the application at hand, the dimension for the hidden layer was chosen to be 12 in order to achieve similar reduction in dimensionality for the signal path from input to hidden layer (i.e. 147:12) and hidden to output layer (i.e. 12:1). Assuming a fully connected neural network, the signal path between input layer and hidden layer is then given by

$$H_{k,i} = \sigma(W_{k,1} X_{k,i} + b_{k,1}) \tag{11}$$

using the weight matrix $W_{k,1} \in \mathbb{R}^{12\times 147}$, bias term $b_{k,1} \in \mathbb{R}^{12}$, and the activation function σ. The sigmoid function, shown on the right in Figure 9, represents a common choice for neural network applications [8]. Similar to the hidden layer (11), the prediction at the output layer can now be constructed via

$$\hat{Y}_{k,i} = \sigma(W_{k,2} H_{k,i} + b_{k,2}) \in [0,1] \tag{12}$$

using the weight matrix $W_{k,2} \in \mathbb{R}^{1\times 12}$, and the bias term $b_{k,2} \in \mathbb{R}$. Note that for the weight matrices $W_{k,1}, W_{k,2}$ and bias parameters $b_{k,1}, b_{k,2}$, the first subscript denotes the flash module, while the second subscript denotes the layer in the neural network.

With the vector of predictions $\hat{Y}_k = [\hat{Y}_{k,1} \quad \dots \quad \hat{Y}_{k,n_s}] \in [0,1]^{n_s}$ (one element for each of the n_s data samples) generated at the output of the NN, the objective for machine learning is to train the NN, i.e. estimate weight matrices and bias terms, in order to best approximate the set of true labels Y. Following guidelines from [8], a cross entropy cost function is utilized to quantify

the difference between Y and $\hat{Y}_k$. The cross entropy for the collection of true and predicted labels is defined as

$$H\left(Y,\hat{Y}_k\right) = -\frac{1}{n_s}\sum_{i=1}^{n_s}\left[Y_i \ln \hat{Y}_{k,i} + (1-Y_i)\ln\left(1-\hat{Y}_{k,i}\right)\right] \tag{13}$$

Figure 10 displays cross entropy (13) as function of the prediction $\hat{Y}_i \in [0,1]$ for the cases $Y_i = 1$ (blue) and $Y_i = 0$ (red). For the comparison, the squared error $\left(Y_i - \hat{Y}_i\right)^2$ is also plotted for both cases using dashed lines. The cross entropy effectively resembles the loss function between two probability distributions, and it is thus frequently used for optimization and probability estimation. Despite being highly nonlinear, the cross entropy improves and accelerates achievable learning rates during NN training. As shown in [9], this is especially the case when using gradient-descent-based optimization techniques and sigmoid-activation functions.

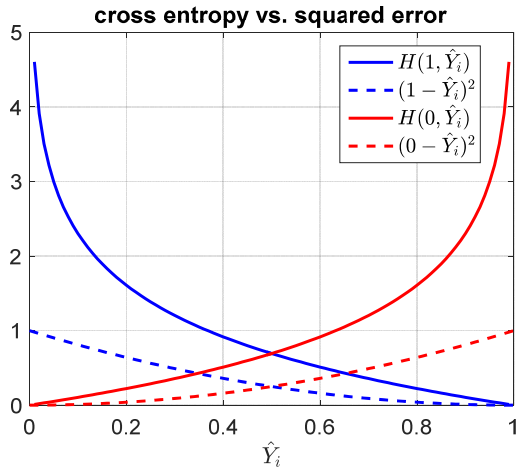

Figure 10: Cross entropy (solid) compared to squared error (dashed) as function of predicted label $\hat{Y}_i \in [0,1]$ for true label $Y = 1$ (blue) and $Y = 0$ (red).

NN training is carried out using back propagation [8]. Back propagation is an iterative, gradient-descent-based optimization scheme to successively update the weight matrices $W_{k,1}$ and $W_{k,2}$ as well as bias vectors $b_{k,1}$ and $b_{k,2}$ such that a pre-defined cost function, here $H\left(Y,\hat{Y}_k\right)$ from (13) will be minimized over time. Regularization terms may be added to prevent network parameters from growing too large during training. Similar to the linear SVM, NN training is carried out using a four-fold cross validation scheme.

6.1.3 Classification Results using Single Flash Module

Adhering to standard performance metrics for binary classification, a confusion matrix as shown in Table 1 can be generated for each flash module k and for each classification approach, i.e. linear SVM with predictions $\tilde{Y}_{k,1}$ and NN with predictions $\hat{Y}_{k,i}$.

Table 1: Confusion matrix including definitions for TNs, FPs, FNs, and TPs.

		Prediction	
		$\hat{Y}_{k,i}, \tilde{Y}_{k,1} = 0$ (nominal)	$\hat{Y}_{k,i}, \tilde{Y}_{k,1} = 1$ (anomaly)
Condition ***(ground truth)***	$Y_i = 0$ (nominal)	True Negatives (TNs)	False Positives (FPs)
	$Y_i = 1$ (anomaly)	False Negatives (FNs)	True Positives (TPs)

From [1][2], the existing data set consists of $n_s = 900$ data samples of which 720 samples are labeled nominal, i.e. $Y_i = 0$, and 180 samples are labeled anomalous, i.e. $Y_i = 1$. A four-fold cross validation scheme is used to train and test both SVM-based or NN-based approaches. During four-fold cross validation, a single classifier is trained on 3/4 of the data (i.e. 540 nominals and 135 anomalies) and subsequently tested for out-of-sample performance on the remaining, unseen 1/4 of the data (i.e. 180 nominals and 45 anomalies). Cycling through all four folds for data partitioning, each data sample will be used three times for training and once for testing. For each 1/4 of data, out-of-sample test results as established by the corresponding classifier are added to the confusion matrix. After the confusion matrix has been constructed, the following *balanced* performance metrics can be computed

$$\begin{aligned} \text{Accuracy} &= \frac{4\text{TPs} + \text{TNs}}{4\text{TPs} + \text{FPs} + 4\text{FNs} + \text{TNs}} \\ \text{Precision} &= \frac{4\text{TPs}}{4\text{TPs} + \text{FPs}} \\ \text{Recall} &= \frac{4\text{TPs}}{4\text{TPs} + 4\text{FNs}} \\ \text{F1} - \text{Score} &= \frac{2 \cdot 4\text{TPs}}{2 \cdot 4\text{TPs} + \text{FPs} + 4\text{FNs}} \end{aligned} \tag{14}$$

It is important to note that when populating the confusion matrix in Table 1, all anomalies, i.e. $Y_i = 1$, will be weighted by a factor of *4* in order to account for the uneven distribution of positive and negative samples (or prevalence of only 0.2). In other words, all TPs and FNs are pre-multiplied by *4* in (14) to generate *balanced* performance metrics.

Figure 11 shows the resulting metrics (14) for all eight flash modules, i.e. $k = 1, \ldots, 8$, using single flash classifiers. Results for linear SVM classification are plotted in red, while NN-based results are plotted in blue. It is clear from Figure 11 that (i) anomaly detection based on single flash information is indeed possible using either approach, as accuracy is consistently above 0.5, and (ii) overall classification performance increases if the neural network architecture from Section 6.1.2 is used.

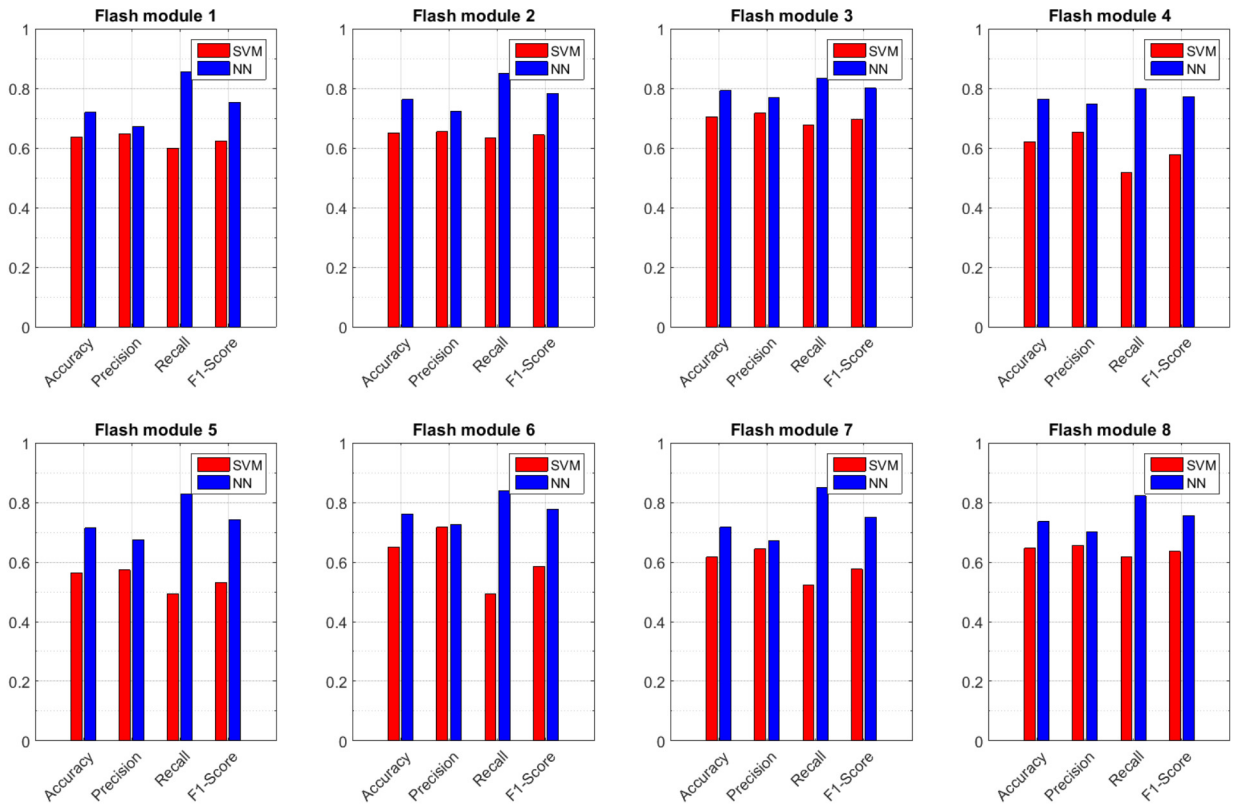

Figure 11: Classification performance for each flash module using linear SVM (red) and neural network (blue) showing accuracy, precision, recall, and F1-score.

As with any classification scheme, one can usually trade FPs (or Type I misclassifications) for FNs (or Type II misclassifications). Depending on the application, it may be more severe to miss an anomaly than to falsely identify a nominal build condition as anomaly. For both approaches, the bias terms, specifically $\beta_{k,0}$ in (10) for SVM and $b_{k,2}$ in (12) for NN, can be manually modified to achieve such a trade-off.

Figure 12 on the next page displays classification performance using receiver operating characteristic (ROC) curves that track the true positive rate (TPR) and false positive rate (FPR) as the bias term is modified. By definition, the upper right corner of the ROC curve represents the case for with $\beta_{k,0},\ b_{k,2} \to -\infty$, i.e. all predictions are positive (anomalies), while the lower left corner represents the case for which $\beta_{k,0},\ b_{k,2} \to +\infty$, i.e. all predictions are negative (nominal build). TPR (also known as Recall) and FPR (also known as Fall-out) are defined as follows

$$\begin{aligned} \text{TPR} &= \frac{\text{TPs}}{\text{TPs} + \text{FNs}} \quad \in [0,1] \\ \text{FPR} &= \frac{\text{FPs}}{\text{FPs} + \text{TNs}} \quad \in [0,1] \end{aligned} \tag{15}$$

ROC curves are plotted in red for the SVM classifier and in blue for the NN classifier. Cumulative performance is measured by the area under the curve (AuC), satisfying $0 \leq AuC \leq 1$. The performance gain when using the NN approach over the SVM classifier is highlighted in yellow, showing an average increase of about 20%.

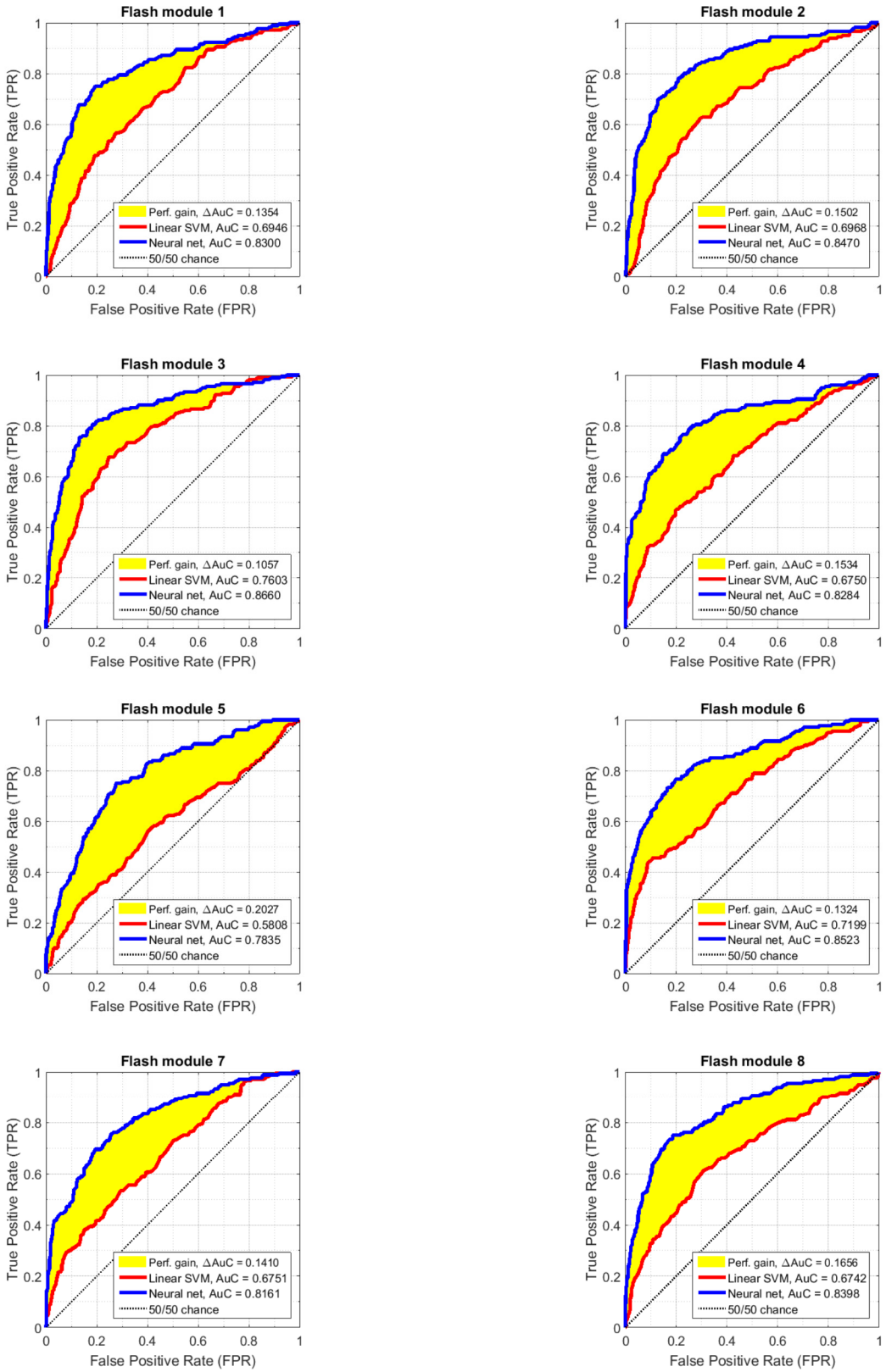

Figure 12: Receiver operating characteristic (ROC) curve for classification performance for each flash module using linear SVM (red) and neural network (blue) classifiers. Performance gain (yellow) when using the neural network is measured by the area under the curve (AuC), satisfying $\mathbf{0} \leq \boldsymbol{AuC} \leq \mathbf{1}$.

From Figure 11 and Figure 12, it can be concluded that some flash modules provide better discriminatory features than others. Best overall classification performance is found for flash module $k = 3$ using accuracy in Figure 11 and AuC in Figure 12 for either classifier. Using the same criteria, flash module $k = 5$ appears to provide the worst classification performance.

6.2 Anomaly Detection fusing Information from Multiple Flash Modules

Both SVM and NN frameworks can be augmented to accommodate multiple flashes and to make prediction based on fused imagery information. Using the feature matrices X_k for each flash module k as given in (8), a concatenated feature matrix X can be constructed

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_8 \end{bmatrix} \in \mathbb{R}^{8n_f \times n_s} \tag{16}$$

The new, concatenated feature matrix now contains $8n_f$ feature dimensions, while the number of samples n_s remains unchanged.

The linear SVM approach (10) can be modified to accommodate the higher dimensional feature space $\mathbb{R}^{8n_f}$. For the feature matrix (16), we find

$$\begin{aligned} \tilde{Y}_i &= 1 \quad \text{if} \quad \beta\, X_i - \beta_0 > 0 \\ \tilde{Y}_i &= 0 \quad \text{if} \quad \beta\, X_i - \beta_0 \leq 0 \end{aligned} \tag{17}$$

Similar to the single flash SVM approach, four-fold cross validation is used to find the hyperplane parameters $\beta \in \mathbb{R}^{8n_f}$ and $\beta_0 \in \mathbb{R}$ that provide the best in-sample classification accuracy between true labels Y_i and predicted labels $\tilde{Y}_i$. Note that compared to (10), the subscript k has been removed to indicate the concatenation of all flash modules.

The augmentation of the NN-based classification scheme towards multiple flash modules is similar to the SVM approach, and the NN utilizes the same, concatenated feature matrix X from (16) as input. In order to keep the number of weight parameters tractable, feature vectors are propagated individually from input layer into the hidden layer for each flash k. The architecture now represents a partially connected two-layer perceptron. Similar to the single flash case (11), we find

$$H_{k,i} = \sigma\big(W_{k,1}\, X_{k,i} + b_{k,1}\big) \quad \text{with} \quad H_{k,i} \in [0,1]^{12} \tag{18}$$

Data fusion of imagery information across all flashes is carried out at the hidden layer by concatenation all hidden layer states such that

$$H_i = \begin{bmatrix} H_{1,i} \\ \vdots \\ H_{8,i} \end{bmatrix} \in [0,1]^{96} \tag{19}$$

Predictions are then constructed using information from all flashes via

$$\hat{Y}_i = \sigma\big(W_k\, H_i + b_k\big) \in [0,1] \tag{20}$$

with the weight matrix $W_2 \in \mathbb{R}^{1\times 96}$ and bias term $b_2 \in \mathbb{R}$.

In order to accelerate NN training, weight matrices can be initialized using values previously learned for the single flash scenario. In other words, the single flash NN classification may serve as pre-training for the multi-flash case. However, besides weight initialization, the overall scheme for NN training via backpropagation remains unchanged.

Classification metrics (14) can now be recomputed for the multi-flash architectures. The left plot in Figure 13 shows a formal comparison in accuracy, precision, recall and F1-score between linear SVM (red) and NN (blue) *master* classifiers. As expected, overall performance increases significantly as compared to the single-flash classification, now reaching 80% and 90% accuracy for SVM and NN, respectively. Associated ROC curves are plotted on the right in Figure 13. The overall performance increase is again evident for either classification approach. Although the NN-based classification generates a larger area under the curve (AuC), the actual gain over linear SVM-based classification (highlighted in yellow) is now less than 10%.

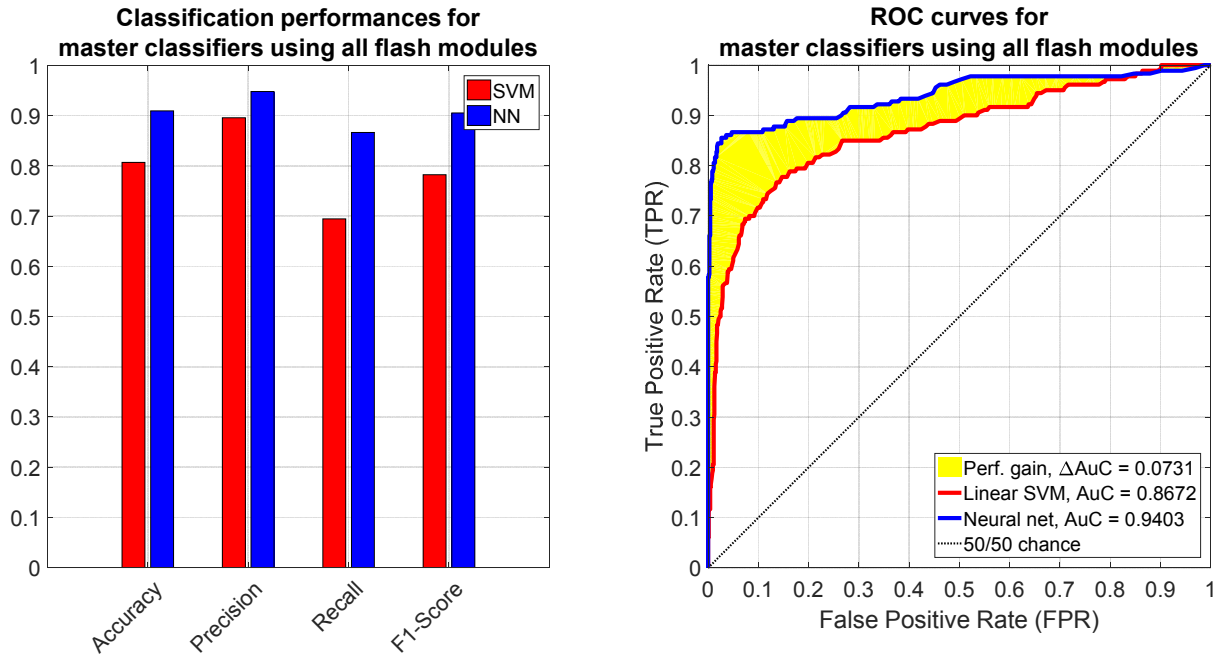

Figure 13: Left - Classification performances of master classifiers with linear SVM (red) and neural network (blue) showing accuracy, precision, recall, and F1-score. Right - Receiver operating characteristic (ROC) curves when using master classifiers.

7. Conclusion and Future Work

This manuscript outlined and validated machine learning strategies for in situ anomaly detection during PBFAM. Computationally efficient and highly parallelizable methods for anomaly detection in CT were presented. Extracted CT anomaly locations including labels were transferred into the in situ layerwise domain, in which visual features were extracted in a neighborhood surrounding the voxels of interest. For the collected data set, both linear SVM and NN-based classification showed adequate performance for anomaly detection when using imagery information from a single flash module. Overall classification performance was further increased by fusing imagery information from all eight flash modules prior to casting a prediction. The NN outperformed the linear SVM. However, the performance gain drops if more features become available, i.e. if image information is fused between flash modules.

Future work will focus on extending the existing framework to accommodate new builds, build environments, lighting conditions and/or imaging sensors. Multiple cross-validation strategies, especially between separate builds, will have to be examined to truly assess the system's overall robustness. In addition, open-source software libraries for machine learning and symbolic arithmetic such as Theano™ or TensorFlow™ will be utilized for future implementations in order to streamline and accelerate training and executing of the algorithm.

Acknowledgments

The authors would like to thank Naval Air Systems Command, Penn State's Applied Research Laboratory and Penn State's College of Engineering for supporting this project. We would also like to thank Ms. Gabrielle Gunderman and Mr. Griffin Jones for their efforts designing the experiments and for performing post-process inspection via 3D CT analysis. Finally, we wish to thank Mr. Don Natale, Mr. Jacob Morgan, and Mr. John Morgan for their contributions in the area of imaging system design and calibration, and for their thoughts on designing the test artifact.

This material is based upon work supported by the Naval Air Systems Command (NAVAIR) under Contract No. N00024-12-D-6404, Delivery Order 0321. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Naval Air Systems Command (NAVAIR).

Bibliography

[1] Christian Gobert, Online Discontinuity Detection in Metallic Powder Bed Fusion Additive Manufacturing Processes using Visual Inspection Sensors and Supervised Machine Learning, MS Thesis, Mechanical Engineering, Penn State University, May 2017.

[2] C. Gobert, E.W. Reutzel, J. Petrich, A.R. Nassar and S. Phoha, "Supervised learning for in-situ visual inspection sensors monitoring metallic powder bed fusion additive manufacturing processes", Elsevier – Additive Manufact. Journal, submitted May 2017

[3] H. Gong, K. Rafi, H. Gu, S. Thomas, B. Stucker, Analysis of defect generation in Ti-6Al-4V parts made using powder bed fusion additive manufacturing processes, Addit. Manuf. 4 (2014) 87-98

[4] S. Everton, M. Hirsch, P. Stravroulakis, R. Leach, A. Clare, Review of *In situ* Process Monitoring and *In situ* Metrology for Metal Additive Manufacturing, Mat. And Des. 95 (2016) 431-445

[5] E-Manufacturing Solutions, EOSINT M 280, "Leading-edge system for the Additive Manufacturing of metal products directly from CAD data", June 2013, http://www.eos.info/systems_solutions/metal/systems_equipment/eos int_m280.

[6] B. K. Foster, E. W. Reutzel, A. R. Nassar, B. T. Hall, S. W. Brown, C. J. Dickman, Optical layerwise monitoring of powder bed fusion, Solid Freeform Fabrication Symposium Proceedings; Austin, TX, 2015.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Second Edition, Springer, 2008

[8] Laurene Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms And Applications*, 1st edition, Pearson Education, 1994

[9] Michael Nielsen, "Neural Networks and Deep Learning', Chapter 3, Determination Press, Jan 2015.