Additive Manufacturing 36 (2020) 101453

Contents lists available at ScienceDirect

Additive Manufacturing

journal homepage: www.elsevier.com/locate/addma

Layer-wise anomaly detection and classification for powder bed additive manufacturing processes: A machine-agnostic algorithm for real-time pixel-wise semantic segmentation

Luke Scime[a,*], Derek Siddel[b], Seth Baird[c], Vincent Paquit[a]

[a] Electrical & Electronic Systems Research Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830, United States
[b] Energy & Transportation Science Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830, United States
[c] Reactor & Nuclear Systems Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830, United States

ARTICLE INFO

Keywords:
Additive manufacturing
Machine learning
Convolutional neural network
In-situ anomaly detection
Semantic segmentation

ABSTRACT

Increasing industry acceptance of powder bed metal Additive Manufacturing requires improved real-time detection and classification of anomalies. Many of these anomalies, such as recoater blade impacts, binder deposition issues, spatter generation, and some porosities, are surface-visible at each layer of the building process. In this work, the authors present a novel Convolutional Neural Network architecture for pixel-wise localization (semantic segmentation) of layer-wise powder bed imaging data. Key advantages of the algorithm include its ability to return segmentation results at the native resolution of the imaging sensor, seamlessly transfer learned knowledge between different Additive Manufacturing machines, and provide real-time performance. The algorithm is demonstrated on six different machines spanning three technologies: laser fusion, binder jetting, and electron beam fusion. Finally, the performance of the algorithm is shown to be superior to that of previous algorithms presented by the authors with respect to localization, accuracy, computation time, and generalizability.

1. Introduction

1.1. Metal powder bed additive manufacturing

Metal Additive Manufacturing (AM) encompasses a wide range of developing technologies which promise the ability to create complex three-dimensional components through layer-wise addition of material. Initially used solely for prototyping, recent advances and market conditions have allowed metal AM to begin producing near-net-shape parts for end-use [1,2]. In particular, metal AM is nominally well-suited for production of highly engineered or customized parts in low to medium volumes [3]. However, many of the applicable industries (e.g. aerospace, medical implants, nuclear, and automotive) require production of parts with higher levels of quality, better pedigree traceability, and lower rejection rates than can be achieved using only open loop control schema and ex-situ inspection [4].

Additively manufactured components are particularly prone to deterministic (but difficult to model) anomalies such as sub-optimal thermal histories [5], stochastic defects such as porosity [6], process stability issues which cause build failures [7], and general machine errors. Some of these anomalies occur on such short time and length scales that they must be detected during fusion or binding of the feedstock material. However, many anomalies are persistent and large enough for layer-wise detection to be possible. The focus of this work is the development of an algorithm for the layer-wise detection of these surface-visible anomalies via the real-time automated analysis of imaging data collected from powder bed metal AM processes.

All powder bed metal AM processes share several fundamental precepts. In these processes a thin layer (20 μm – 200 μm) of powdered metal feedstock is spread across a substrate using a recoating mechanism. A two-dimensional cross-section of the desired component is then either fused using a targeted heat source or bound together using a binder. Next, the substrate is lowered by the height of a powder layer and the process is repeated until the "build" is complete. This work presents anomaly detection results for three distinct powder bed technologies:

1 In Laser Powder Bed Fusion (L-PBF) one or more focused laser beams are used to selectively melt the metal powder [8].
2 In Electron Beam Powder Bed Fusion (EB-PBF) a magnetically-

* Corresponding author.
E-mail address: scimelr@ornl.gov (L. Scime).

https://doi.org/10.1016/j.addma.2020.101453
Received 2 March 2020; Received in revised form 25 June 2020; Accepted 13 July 2020
Available online 16 July 2020

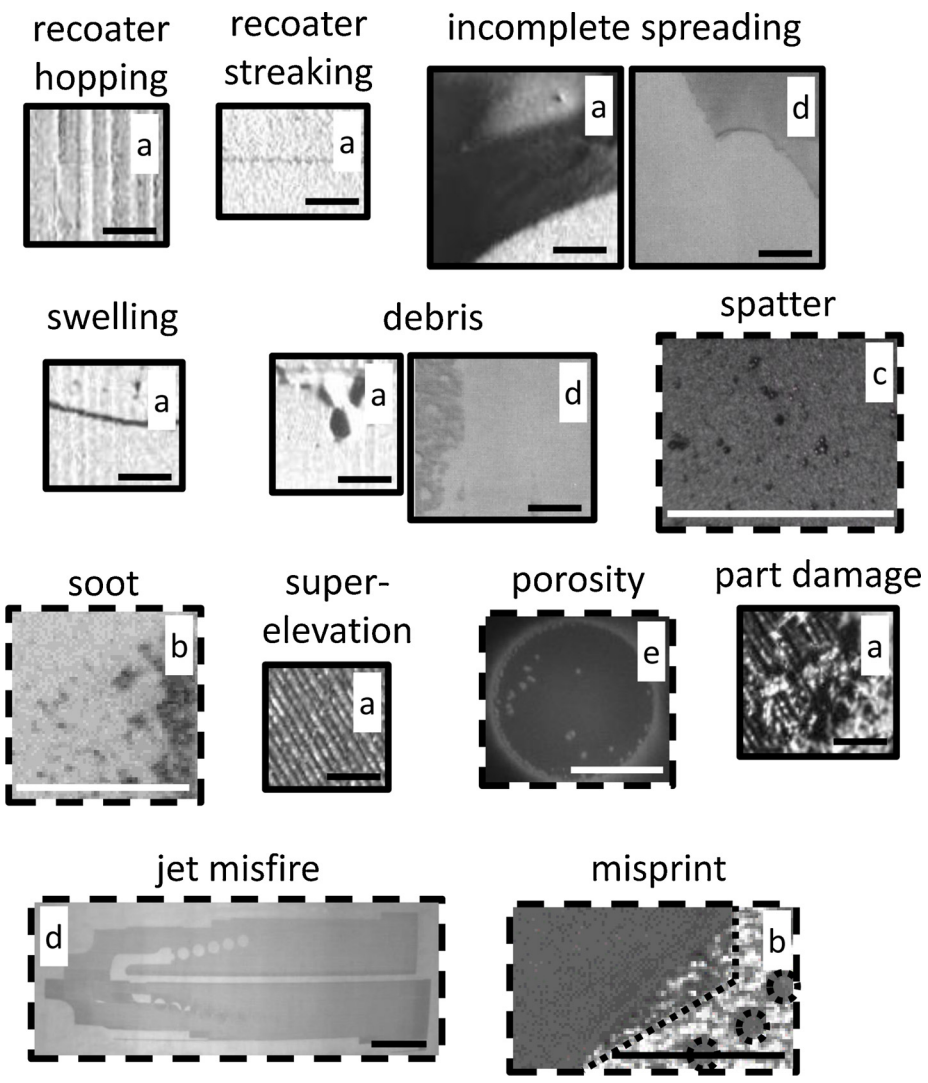

Fig. 1. Representative examples of twelve different surface-visible anomalies detected in this work. The examples are sourced from (a) EOS M290 (L-PBF), (b) ConceptLaser M2 (L-PBF), (c) Renishaw AM250 (L-PBF), (d) ExOne M-Flex (binder jetting), and (e) Arcam Q10 (EB-PBF) data. Solid borders denote images taken immediately after powder spreading while dashed borders denote images taken immediately after powder fusion or binder deposition. All scale bars are 10 mm in length. The dotted lines on the *misprint* example indicate the boundaries of the intended geometry. Note that some of these images have also appeared in the lead author's previous work [18,19].

steered electron beam is used to selectively melt the powder bed [8]. These processes typically operate at much higher background temperatures than L-PBF machines [8].

3 In binder jetting machines melting does not occur in-situ. Instead, a binder is deposited on the powder bed, using an inkjet printer head, to temporarily hold the powder in place [9]. At this stage the part is considered "green" and is placed in a furnace for either sintering or infiltration [9].

1.2. Surface-visible powder bed anomalies

Fig. 1 shows examples (from each relevant technology) of each of the layer-wise surface-visible anomalies detected in this work. *Recoater hopping* occurs when a rigid recoater blade impacts a part just below the powder surface [7] and is characterized by repeated lines perpendicular to the recoater travel direction. *Recoater streaking* occurs when a recoating mechanism is either damaged or dragging a relatively large contaminant across the powder bed [10] and is characterized by a line parallel to the recoater travel direction. *Incomplete spreading* (short feed) is present when an insufficient amount of powder is spread across the powder bed in one or more layers [11]. *Debris* is a broad classification used to describe small-to-medium scale disturbances on the powder bed. *Spatter* is generated during melting of the powder bed and typically consists of ejecta from the molten pool or melted but unconsolidated powder particles [12]. In this context, *soot* is *spatter* imaged at a lower effective resolution. *Swelling* and *super-elevation* are evident when a part warps up above the powder bed [13] either as the result of residual thermal stresses [14] or swelling from excessive energy input [15]. Porosity generally encompasses a range of small-scale defects formed by a variety of mechanisms [6]. For the purposes of this work, *porosity* refers only to lack-of-fusion porosity [16,17] in the EB-PBF process. This porosity is surface-visible primarily because of a geometry-induced change in emissivity in the Near Infrared (NIR) wavelength range.

The *part damage* classification is often visually indistinct from *debris* and therefore requires additional information regarding the intended part geometry for differentiation. For the purposes of this manuscript, areas of powder bed with no anomalies are classified as *powder* while anomaly-free fused or bound powder regions are classified as *part*. Finally, unlike the other anomalies, *jet misfires* and *misprints* are generally not visually distinct from the *part*. *Jet misfires* and *misprints* are respectively defined as scenarios in which powder is bound or fused in unintended locations. Therefore, *jet misfires* and *misprints* must be detected either through comparison of *part* classifications to the intended geometry (Section 3.8) or as described in Section 5.4. *Jet misfires* occur exclusively in binder jetting processes and are typically caused by one of the following: (1) a loss of information during data transmission from the user interface computer to the printhead controller, (2) an error in the OEM (Original Equipment Manufacturer) part model slicing software, or (3) a mechanical or sensor failure which affects the positioning of the printhead with respect to the powder bed. *Misprints* occur in L-PBF systems and appear to be caused by errors in either the OEM laser path planning software or the interpretation of the laser paths by the

mirror galvanometer controllers.

1.3. Existing work and limitations

In the last decade significant research efforts have been undertaken on the topic of layer-wise anomaly detection for powder bed AM processes. Many of these efforts focus on developing improved sensing modalities such as using flash bulbs to better illuminate the powder bed [7,20], fringe projection techniques to recover depth information [21,22], and high resolution scanners [23]. However, the development of optimal imaging systems is only part of the challenge. Regardless of the data source, anomalies must be autonomously detected for any of these approaches to meet the quality and process control needs of the industry.

On this data analysis front, a substantial portion of the effort has focused on comparison of the printed part geometry to the intended part geometry [7,20,24,25]. Seminally, *recoater streaking* was detected by Craeghs et al. [10] using hand-crafted feature extractors. Indeed, much of the early powder bed anomaly detection efforts relied upon human-designed detection algorithms such as line profiles [10] and segmentation based on a global or local intensity thresholds [7,20,24,26]. While simple to implement, such approaches are severely limiting, often detecting only a single type of anomaly or classifying all off-nominal regions as a generic "flaw." As a result, modern Computer Vision (CV) techniques driven by Machine Learning (ML) models have recently become popular for powder bed anomaly detection.

In their previous work, the authors achieved multi-class patch-wise anomaly classification using a Bag of Words methodology [18] and a transfer-learned Multi-scale Convolutional Neural Network (MsCNN) [19]. *Super-elevation* was detected by Jacobsmühlen et al. [13] using various ML models (support vector machines and random forests) operating on extracted DAISY local features. Aminzadeh et al. [27] used Bayesian statistical models operating on hand-crafted features extracted from intensity-thresholded post-fusion images. In addition to relying on human judgment to select the features, this Bayesian approach also assumes that the population of all powder bed imaging data is highly similar to the training data (i.e. the learned model may not be generalizable) and it is restricted to labeling relatively large regions as a given class (i.e. localization is coarse). Gobert et al. [28] first extracted features from stacks of layer-wise data (images taken under different lighting conditions [20]) using a set (all linearly-independent permutations spanning a discretized space) of three-dimensional convolution filters. Once the features were extracted, an ensemble of support vector machine models were used to classify voxels as either anomalous or nominal. This approach relies on brute-force for feature extraction – limiting the filter size and risking overwhelming any learned model with non-salient features. Furthermore, classification is based entirely on highly local information which may not be sufficient for classification of some anomalies. Imani et al. [29] used both spectral graph theory and multifractal analyses to extract features describing the layer-wise porosity of a given part. These descriptors could then be used to broadly predict the processing parameters used to fuse that part, assuming that the training database contained examples of layers produced using that processing parameter combination. Finally, in recently submitted work, Kirka et al. [30] used a U-Net to robustly detect porosity in NIR EB-PBF images.

1.4. Goals and novelty of the current work

The powder bed anomaly detection and classification approaches extant in the literature have significant restrictions including at least one of the following: coarse anomaly localization resolution, a small number of discriminative anomaly classes, or a lack of generalizability to other powder bed processes. In this work a novel Convolutional Neural Network (CNN) architecture, referred to by the authors as a Dynamic Segmentation CNN (DSCNN), is developed specifically to address the following key requirements:

1. Model predictions (anomaly classifications) must be returned quickly enough to provide real-time information, even at high camera resolutions.
2. Semantic segmentation must always be achieved pixel-wise at the native resolution of the imaging sensor.
3. Because training data for any individual machine may be limited, knowledge learned on data from one AM machine must be rapidly transferrable across technologies between entirely different powder bed machines and imaging systems.
4. The algorithm must naturally support fusion of data from multiple imaging systems or other spatially correlated sensing modalities.

Semantic segmentation refers to labeling pixels or groups of pixels (super-pixels) in an image as belonging to one or more human-interpretable classes [31]. Over the years, a host of successful CNN architectures have been developed, including AlexNet [32], VGG [33], and GoogLeNet (Inception) [34]. However, these algorithms focus on classifying entire images (e.g. "there is a cat in this picture") while there may be hundreds of distinct anomaly occurrences visible in a single build layer. While semantic segmentation has been achieved using patch-wise application of such conventional CNNs (as in the authors' MsCNN [19]), Fully Convolutional Neural Networks [31] and related extensions such as U-Nets [35] currently lead this field. These existing approaches, however, struggle to meet the second and third key requirements in most implementations.

The MsCNN can only return segmentation results patch-wise without implementing a computationally-expensive sliding window approach [36]. In the case of Fully Convolutional networks, the computational tradeoff between decreasing the pooling strides and learnable kernel size limits the output resolution of the segmentation as discussed by Long et al. [31]. Furthermore, dense predictions are returned via upsampling using up-convolution layers and these would need to be relearned for different input camera resolutions [31]. Similarly, the skip layers which transfer the multi-scale contextual (semantic) information to the pixel-scale may need to be architecturally modified in situations where the receptive fields observe substantially different types of information than provided during training. Finally, the U-Net architecture addresses the challenge of arbitrarily-sized input data via patch-wise application of the entire network [35]. While effective, this approach places an effective limit (imposed by current GPU memory capacities) on the largest size scale from which context can be carried down to the pixel-scale. Furthermore, transferring knowledge from one image size to another assumes that features learned with the initial receptive fields will be applicable to differently-scaled receptive fields. The novel DSCNN approach presented in this work mitigates this tradeoff between high resolution image segmentation and capture of large-scale contextual information. Built around a U-Net core, the DSCNN uses parallel CNNs to capture global context and translate the U-Net's feature maps into higher resolution pixel-wise classifications.

The explicit inclusion and representation (i.e. encoding) of multi-scale contextual information in the authors' DSCNN shares concepts with several other network architectures used for image segmentation. Shen et al. [37] utilized three parallel CNNs to leverage information at discrete size scales to improve patch-wise classifications of medical imagery; an approach similar to the authors' MsCNN [19]. Bertasius et al. [38] used high-level context to inform canonically low-level tasks (contour detection) in their DeepEdge algorithm. Finally, and perhaps most similarly to the DSCNN, Raj et al. [39] used the extremely deep VGG16 [33] to share large-scale contextual information with a much shallower and finer-detailed CNN to return localized segmentation results for stereo depth images.

Details on the experimental setup and data collection methodology are discussed in Section 2. Additional background on ML as well as all details of the DSCNN algorithm are covered in Section 3. The

performance of the DSCNN across all six powder bed machines is quantified and compared to the authors' previous work in Section 4. Brief case studies demonstrating the usage of the DSCNN on multiple machines are presented in Section 5 with a discussion following. The AM terminology used herein complies as closely as possible with ISO/ASTM 52900:2015 [40].

2. Experimental procedure and methods

2.1. Programming environment

All software was developed in Python 3.6. Relevant libraries used include TensorFlow 1.12.0, OpenCV 3.3.1, and Open3D 0.4.0.0. No high-level APIs were used on top of TensorFlow.

2.2. Powder bed machines, imaging configurations, and camera calibration

In this work, the DSCNN is successfully demonstrated on data from three L-PBF machines, two binder jetting machines, and one EB-PBF machine. While the primary focus of this manuscript is on the design and capabilities of the algorithm, the imaging configurations and experimental setups for each machine type are briefly described below and summarized in Table 1. Example powder bed images from each machine type are shown in Fig. 2.

The EOS M290 (EOS GmbH, Germany) is an L-PBF machine located at Carnegie Mellon University's (CMU) NextManufacturing Center. Data are collected using the manufacturer-provided 1 Mega Pixel (MP) visible-light camera with a Field of View (FoV) covering the entire 250 mm × 250 mm powder bed. Illumination is provided by a bank of white LEDs located on the right side of the build chamber and oriented parallel to the powder bed. All analyses are performed off-line after completion of the build. Data were collected and analyzed for builds using Ti-6Al-4 V (Ti64), AlSi10Mg, Inconel 718, Inconel 625, stainless steel 316 L, stainless steel 17 − 4 PH, and bronze as the feedstock.

The ConceptLaser M2 (ConceptLaser GmbH, Germany) is an L-PBF machine located at Oak Ridge National Laboratory's (ORNL) Manufacturing Demonstration Facility (MDF). Data are collected using the manufacturer-provided 5 M P visible-light camera with a Field of View (FoV) covering the entire 240 mm × 240 mm powder bed. Illumination is provided by several sets of white LEDs located at the top of the build chamber and oriented perpendicular to the powder bed. All analyses are performed off-line after completion of the build. The analyzed builds used stainless steel 316 L as the feedstock.

The Renishaw AM250 (Renishaw® plc, United Kingdom) is an L-PBF machine located at the MDF. Renishaw AM250 data are collected using a custom-mounted 15 M P Pixelink PL-D7715CU-T visible-light camera with a relatively small in-focus FoV (approximately 30 mm × 30 mm) due to the shallow angle between the camera axis and the powder bed. Illumination is provided by a ring of white LEDs located at the top of the build chamber. All analyses are performed off-line after completion of the build. The analyzed build used Fe-3Si as the feedstock.

The Arcam Q10 (Arcam AB, Sweden) is an EB-PBF machine located at the MDF. Arcam Q10 data are collected using the manufacturer-provided 5 M P NIR camera with a FoV covering the entire 200 mm × 200 mm powder bed. Unlike the visible imagers, the illumination is not reflected light from an external source; instead, it is emitted light radiated from the high-temperature powder bed itself. All analyses are performed off-line after completion of the build. The analyzed build used Ti64 as the powder feedstock.

The ExOne M-Flex (ExOne®, United States of America) is a binder jetting machine located at the MDF. ExOne M-Flex data are collected using a custom-mounted 10 M P Basler acA3800 − 14um visible-light camera with a FoV covering the entire 250 mm × 400 mm powder bed and a 0.3 M P Boson 20640A050 − 6ACAAX Mid-Wave Infrared (MWIR) camera with a FoV covering approximately 70 % of the powder bed. Illumination is provided by several sets of white LEDs located at the top of the build chamber and oriented perpendicular to the powder bed. All analyses are performed on-line, in real-time as will be described in future publications by the authors. Data were collected and analyzed for builds using H13 stainless steel, boron carbide (B4C), and silicon carbide (SiC) as the feedstock.

The ExOne Innovents (ExOne®, United States of America) are binder jetting machines located at the MDF and the Reactor & Nuclear Systems Division (R&NSD) at ORNL. In total there are two Innovent machines and one very similar Innovent + machine included in this work. ExOne Innovent data are collected using a custom-mounted 20 M P Basler acA5472 − 17um visible-light camera with a FoV covering the entire 65 mm × 160 mm powder bed. Illumination is provided by a set of white LEDs located at the top of the build chamber and oriented perpendicular to the powder bed. All analyses are performed on-line, in real-time. Data were collected and analyzed for builds using B4C and SiC as the feedstock.

Powder bed imaging data from an AddUp FormUp 350 (a joint-venture by Fives Group and Michelin), a ConceptLaser X-Line 2000R (ConceptLaser GmbH, Germany), and an SLM 280 (SLM Solutions GmbH, Germany) have also been analyzed by the authors but those results are not included in this work for brevity.

TAcross all machine types, two 8-bit images per camera are captured for each layer: one immediately following powder fusion or binder deposition and another immediately following powder spreading. Note that the post-spreading image is always linked to the previous post-fusion/binder-deposition image. While initially counter-intuitive, this may be thought of as using the powder spread to "probe" the topology of the just-fused layer. As a preprocessing step, both images are corrected for FoV, perspective distortion, and lighting artifacts using methods functionally similar to those described in Section 2.1 of Scime et al. [18].

Approximate effective camera resolutions (i.e. the FoV of individual pixels) are reported in Table 1 but should be interpreted with the caveat that the off-axis mounting of the cameras results in perspective distortion. Therefore, true resolution will vary over the powder bed. True determination of the resolving power (smallest detectible anomaly)

Table 1
Summary of imaging configurations and experimental setups.

AM Machine	Facility	Material System	Print Area (mm)	Camera Type	Raw Size (pixels)	Calibrated Size (pixels)	Spat. Res. (μm/pixel)
EOS M290	CMU	Ti64, AlSi10Mg, 718, 625, 316 L, 17 − 4 PH, & bronze	250 × 250	visible	1024×1280	875×875	290
ConceptLaser M2	MDF	316 L	240 × 240	visible	2048×2560	2325×2325	110
Renishaw AM250	MDF	Fe-3Si	30 × 30	visible	3288×4608	1530×1530	20
Arcam Q10	MDF	Ti64	200 × 200	NIR	2050×2448	1632×1632	120
ExOne M-Flex	MDF	H13, B4C, & SiC	250 × 400	visible & MWIR	2748×3840 640×512	1581×2346	130
ExOne Innovent	MDF & R&NSD	B4C & SiC	65 × 160	visible	3672×5496	1922×4731	30

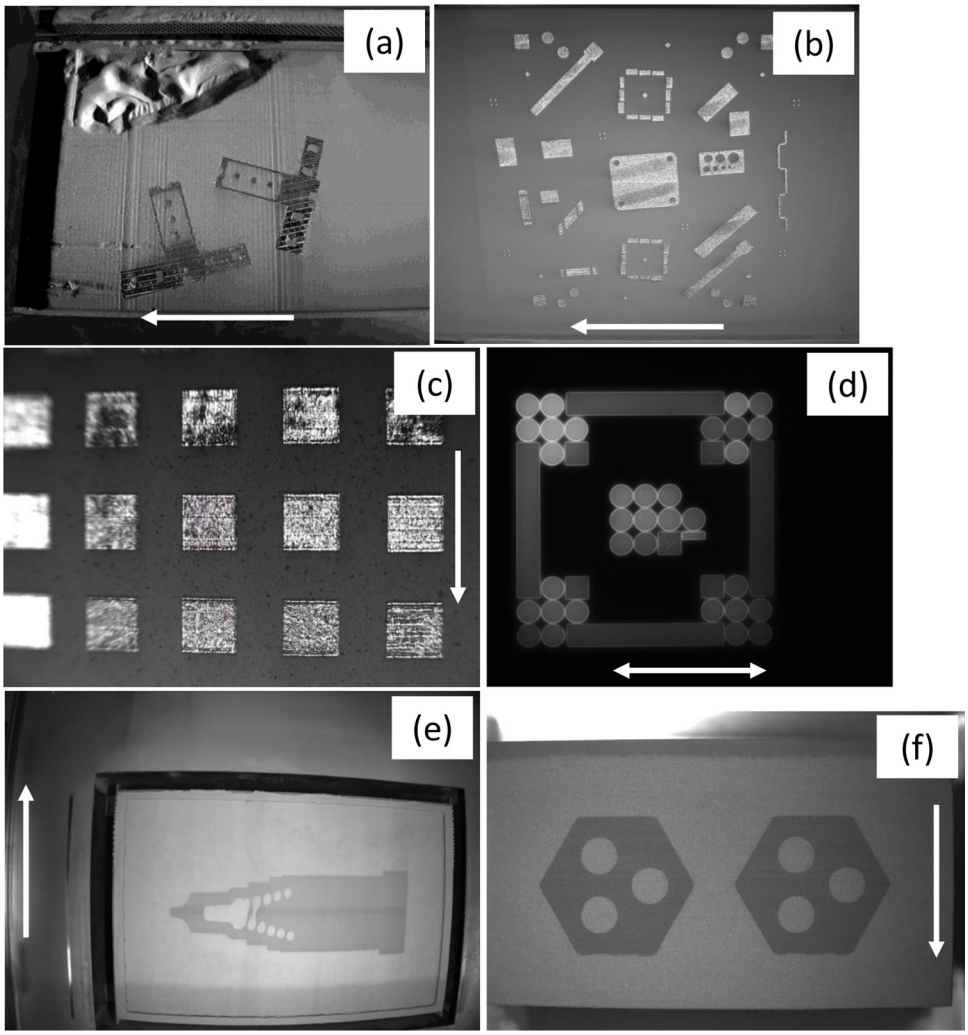

Fig. 2. Examples of uncalibrated, post-fusion/binder-deposition images for the (a) EOS M290, (b) ConceptLaser M2, (c) Renishaw AM250, (d) Arcam Q10, (e) ExOne M-Flex, and (f) ExOne Innovent. The arrows indicate the recoater travel direction during powder spreading or smoothing. Because these images are uncalibrated (distorted), scale bars have not been included; the sizes of the corresponding powder beds are listed in Table 1.

would require imaging a calibration target [41] in multiple locations across the powder bed. This procedure has not been performed for any of the imaging configurations and is immaterial to the performance of the algorithm itself. Also note that for the ExOne M-Flex, the much lower resolution MWIR camera image is artificially resized to match the resolution of the visible-light camera reported in Table 1. Its true spatial resolution is approximately 400 μm/pixel.

Finally, the authors would like to note that there is no implication that any of the imaging and lighting configurations are "optimal." Indeed, the optimal camera resolution and FoV are highly dependent upon the anomalies of interest and other work has demonstrated the benefits of more advanced structuring of the illumination sources [20,22]. The primary purpose of this work is to present an algorithm capable of robustly returning pixel-wise anomaly classifications independent of any specific imaging solution.

2.3. Design intent integration

If available, the nominal Computer Aided Design (CAD) geometry of the printed parts can be incorporated into the DSCNN for visualization purposes and as described in Section 3.8. Such CAD data are extracted from layer-wise slices of the 3D geometries using CV techniques functionally similar to those described in Section 2.4 of Scime et al. [19].

3. Theory

3.1. Overview and intuition

At their core, machine learning algorithms autonomously generate models (within human-defined constraints) to predict information of value (e.g. the presence of an anomaly) based on *features* extracted from provided data (e.g. images of a powder bed) [42]. Training such models requires "ground truth" information, i.e. representative data which have been labeled by a human expert (e.g. an AM machine operator) or other independent process. Deep Learning algorithms go further, simultaneously learning both a model and an optimal set of *features* which typically outperform human-crafted *features* [42].

In order to address the four stated requirements, the presented algorithm utilizes a highly customized CNN architecture. As a subset of generalized neural networks, CNNs leverage the a priori knowledge that the data of interest are images – largely limiting their calculations to convolution [32] (element-wise multiplication followed by summation) operations [43] which produce *responses* composing the *features*. Note that in this context the term "images" may refer broadly to any data on a regular grid for which the spatial relationships between the data points on the grid encode information.

As introduced in Section 1, segmentation tasks seek to classify each pixel in an image based on the value stored at the given pixel as well as

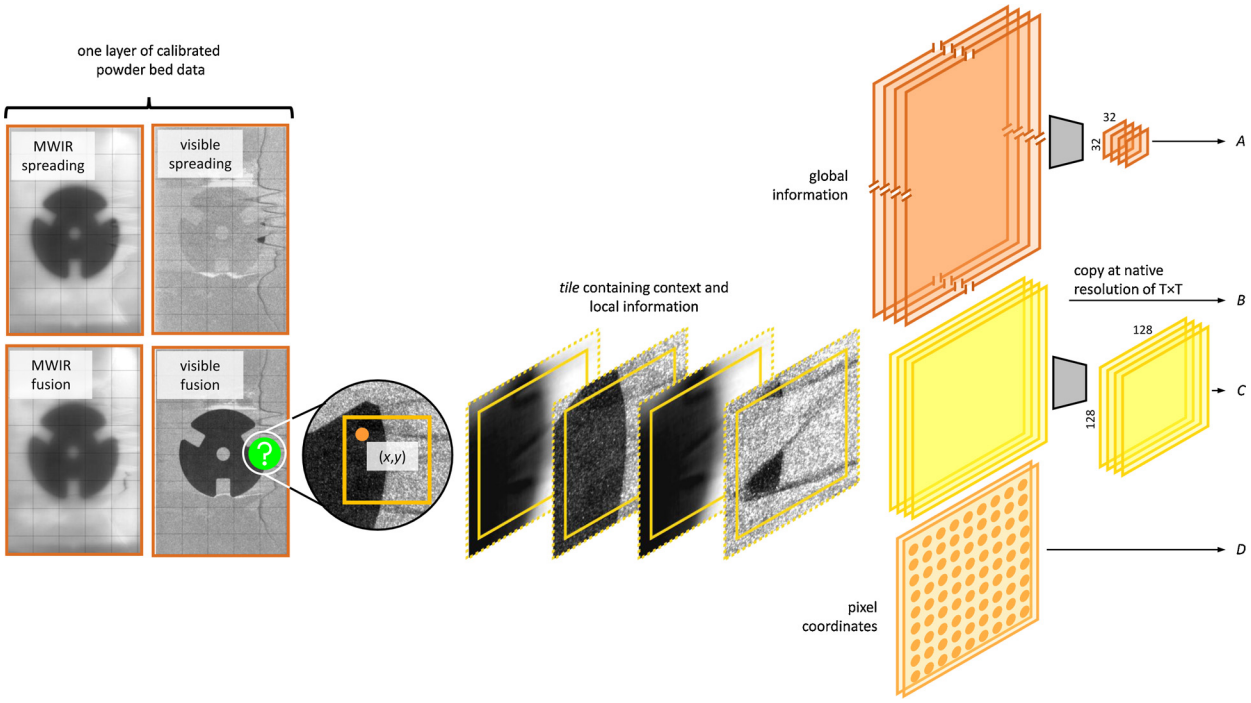

Fig. 3. Graphical representation of the *tile*-wise data ingested by DSCNN. A total of four image stacks, indicated by the *A*, *B*, *C*, and *D* annotations, are extant for each *tile*. (For interpretation of the color in in this figure, the reader is referred to the web version of this article).

information about the area surrounding that pixel. The need to "carry" contextual information from larger size scales down to smaller size scales is well recognized in the literature and was the key insight leading to the authors' previous MsCNN algorithm in which *patches* at three discrete size scales were forced into the three RGB color channels of AlexNet. One limitation of the modified AlexNet CNN is that most of the *features* were learned using data for which a spatial correlation between the three input channels corresponds to a real-world spatial correlation, while this is not the case for the multi-scale data. Therefore, in the DSCNN, three *legs* operate in parallel to capture information at the local, regional, and global size scales.

The most important *leg* of the DSCNN consists of a U-Net [35] designed to extract deep morphological *features* while recalling their approximate spatial locations. Another *leg* consists of a simple CNN to extract *features* at a very large size scale. *Features* at this size scale might encode information about environmental lighting conditions, major powder spreading aberrations, and material feedstock characteristics. Finally, a localization *leg* guarantees pixel-wise classification at the native input resolution by translating the multi-scale contextual information into pixel-wise labels based on local regions around each pixel. The use of a parallel localization network also allows *features* learned for one AM machine to be applied to a different AM machine regardless of the native resolution or physical size of the imager's FoV. Fusion of data from multiple imaging systems (the number of which may vary from machine to machine) is naturally handled via a dynamic resizing of the initial *feature* extractor *kernels* described in Section 3.9.

Like several other implementations of U-Nets, the DSCNN analyzes the input images in discrete, marginally overlapping, *tiles* (fully synonymous with a "patch") for the following reasons:

1 *Tile*-wise delineation of the input data (Section 3.3) prevents the GPU memory from limiting the maximum native powder bed image size as the full image is naturally broken up into mini-batches of *tiles* for evaluation. This becomes a significant concern for desktop-level GPUs as image sizes of 10 M P or greater are considered.
2 By resizing the input *tiles*, the effective receptive fields of the convolutional units deeper in the network can be modified without making structural changes to the network architecture. To make a similar change in [31], for example, *kernel* and/or pooling strides and/or sizes must be modified. Such modifications would likely prevent transfer of learned parameters between different powder bed machines.

Overall, the DSCNN is significantly smaller than AlexNet (14×10^6 vs. 60×10^6 learnable parameters) but it benefits from an architecture designed specifically for segmentation of images for which important context exists at disparate size scales and its extracted *features* are learned on relevant data (i.e. layer-wise powder bed images). The parallelized structure and relatively shallow depth (number of sequential layers) of the network also enables high classification speeds in excess of one million pixel-labels per second. While the DSCNN shares many commonalities with other CNN and U-Net implementations, its unique features include:

1 Three parallel *legs* which explicitly encode information at multiple sizes scales, enabling the use of both global and highly local information for segmentation. The overall network topology is also carefully designed to facilitate transfer learning between different AM machines.
2 The global position of each pixel within the layer images is explicitly encoded and preserved throughout the tiling operation.
3 Implicit acceptance of an arbitrary number of input channels.
4 Non-standard data normalization, upsampling architecture choices, loss functions, and data augmentation to accommodate AM datasets with very high-resolution images, unusual lighting environments, and major class-wise imbalances in the training data.

Throughout this section, aspects of the DSCNN are referred to either as "canonical" or "non-canonical" to highlight the differences between the DSCNN and other networks known from the literature. The inputs to the neural network and the DSCNN architecture are summarized in Fig. 3 and Fig. 4, respectively.

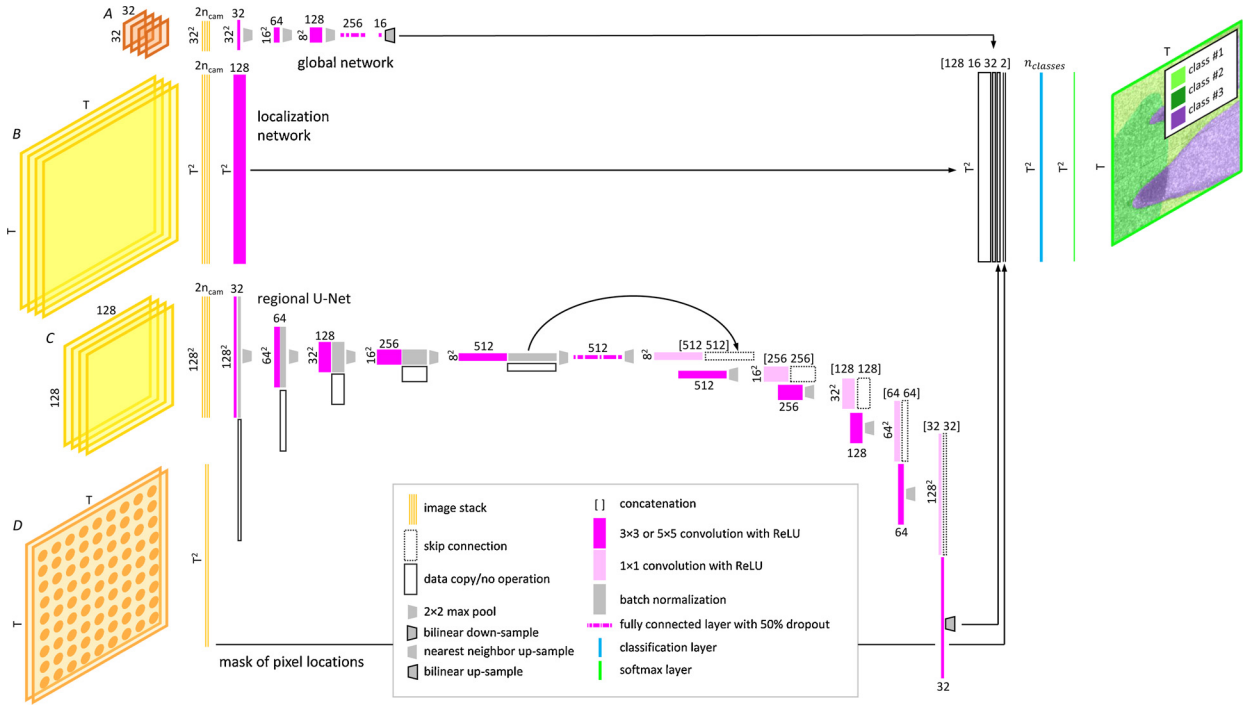

Fig. 4. Graphical representation of the DSCNN architecture. Data volumes are rendered approximately to scale with the channel dimensions compressed by a relative factor of ten. The numbers of channels are indicated horizontally, and the square spatial dimensions are indicated vertically. (For interpretation of the color in in this figure, the reader is referred to the web version of this article).

3.2. Manual ground truth data labeling

Training a CNN (Section 3.7) requires a significant amount of ground truth data, typically on the order of 10^5 – 10^7 *targets* (labeled samples) [44]. This problem was avoided in the authors' previous work [19] through the use of transfer learning [45] – a process by which the low-level *feature* extractors are learned using a large but generic dataset (e.g. pictures of cats and dogs) and only the final layers of the CNN are learned using the smaller dataset of interest. While highly effective, the unusual architecture of the DSCNN precludes transfer learning from such a generic dataset. Fortunately, because the *targets* are now pixels instead of *patches*, collection of millions of labeled samples is achievable with a manageable labor cost.

All ground truths were collected through manual pixel-wise annotation of powder bed images from multiple layers, multiple builds, and multiple AM machines. The authors used a custom graphical interface to label layers of data, with each layer-wise dataset consisting of a calibrated (Section 2.2) image taken after powder-spreading and a calibrated image taken after powder-fusion or binder-deposition for each camera. Where possible, the authors adhered to the following guidelines while labeling the training data for each machine type:

1 Representation from all of the anomaly classes is critical. While the anomaly labels do not need to be class-wise balanced (Section 3.7), at least 100,000 pixels were labeled for each class.
2 Variety is more important than quantity. Because only a finite number of person-hours were available to label data, the available data were prioritized for inclusion during training. While merely increasing the number of labeled pixels is generally beneficial, it is critical that "rare" events are well-represented in during training (Section 4.3). Such variety might include capturing both thin- and thick-wall *part* geometries, unusual *incomplete spreading* behavior, and atypical *debris* morphologies.
3 To enhance the variety seen during training, only labeling data from a single build was avoided where possible. Labeling data from multiple builds is particularly important given the nature of the global network (Section 3.4).
4 Some anomaly classes are often co-located. In situations where multiple classes might legitimately apply to a single pixel, a consistent set of rules were used to determine the appropriate label. These rules prioritized the more detrimental anomaly classes over comparatively minor issues. For example, the authors considered *super-elevation* to be more serious than *incomplete spreading*, as is it indicates a direct interaction between the anomaly and the part geometry.
5 As the DSCNN leverages the full stack of input images (Section 3.3) to classify each pixel, all of the images for a given layer (i.e. the post-spreading and post-fusion images for each camera) were considered during annotation.
6 Because the physical location of each pixel is an input to the DSCNN, the authors were careful to label data from all regions of the powder bed.
7 At least three layers of data were labeled for each machine type before training was attempted. While this was sufficient for reasonable model accuracies when combined with transfer learning (Section 3.9), the robustness and accuracy of the DSCNN models improved with increasing dataset size and compositional variety.

Table 2 reports the total number of pixels labeled for each AM machine and a class-wise breakdown of the training data. For all the machine types, the relative occurrences of the different anomaly classes within the training database varied substantially; with some classes accounting for as much as 83.2 % or as little as 0.1 % of the labeled data. As discussed in Section 3.7, if a *tile* is only partially labeled, it may still be considered during training, with the unlabeled pixels excluded from the loss calculation.

3.3. Construction of the input layer

Canonically, all CNNs perform a sequence of operations on data stored within an input layer. In the trivial case, a CNN's input layer may consist of a resized grayscale image of width W and height H. Similarly,

Table 2
Accounting of ground truth data used for training. The balance of the training data consists of unlabeled pixels.

	EOS M290	ConceptLaser M2	Renishaw AM250	Arcam Q10	ExOne M-Flex	ExOne Innovent
Number of Builds	8	12	1	1	10	10
Number of Layers	37	23	3	4	14	13
Number of Pixels (10^6)	33	90	8	7	36	132
Anomaly Class						
Powder	83.2 %	71.8 %	73.5 %	44.7 %	63.5 %	80.5 %
Part	3.1 %	12.8 %	25.2 %	41.1 %	17.6 %	13.1 %
Recoater Hopping	1.3 %	–	–	–	–	–
Recoater Streaking	0.7 %	0.2 %	–	–	0.6 %	0.5 %
Incomplete Spreading	5.2 %	3.0 %	–	–	4.0 %	5.1 %
Swelling	0.1 %	0.1 %	0.1 %	–	–	–
Spatter	–	–	1.2 %	–	–	–
Soot	–	0.2 %	–	–	–	–
Debris	2.7 %	0.4 %	–	–	3.1 %	–
Super-Elevation	1.2 %	0.4 %	–	–	–	–
Part Damage	0.0 %[a]	–	–	–	–	–
Porosity		–	–	0.2 %	–	–

[a] The *part damage* class is not directly learned, it is only predicted according the heuristics outlined in Section 3.8.

the input layer of a CNN designed for RGB color images may be a stack of three images of size $H \times W$ with each image of the stack considered a "channel." In the case of the DSCNN, the input layer is substantially more complex as it must capture information from multiple camera images of each layer while architecturally ensuring that contextual information at larger size scales can be effectively leveraged during pixel-wise classification. The components of the input layer for a given *tile* are identified in Fig. 3 as four image stacks *A*, *B*, *C*, and *D*.

First, each layer of powder bed data, consisting of both a calibrated post-spreading and a calibrated post-fusion/binder-deposition image for each camera, is broken up into a fixed grid of *tiles* which are sized to capture important contextual information at the regional size scale. As a result, the appropriate *tile* sizes for each AM machine were determined with the assistance of domain experts. If layer images are not evenly divisible by the *tile* size *T*, the final *tiles* (on the right and bottom edges of the image) will be pinned to the image borders and will overlap some of the previous *tiles*. When the *tile*-wise DSCNN classifications are reconstructed into the full image, these "edge" *tiles* overwrite any overlapping regions.

The native square sizes of the *tiles* vary by AM machine as follows: EOS M290 (128), ConceptLaser M2 (225), Renishaw AM250 (200), ExOne M-Flex (400), ExOne Innovent (450), and Arcam Q10 (200) with all values in pixels. One copy of each *tile* remains at the native resolution (Fig. 3-*B*) while another copy is always resized (using bilinear interpolation) to 128 pixels × 128 pixels (Fig. 3-*C*). Each native-resolution *tile* is padded with five surrounding pixels (or symmetric pixels if at the true edge of the powder bed image) to mitigate artifacts at the *tile* boundaries. This results in a final size of $T+10$ pixels × $T+10$ pixels. The resized *tile* is padded with only two pixels resulting in a final size of 132 pixels × 132 pixels. The amount of padding required to maintain layer outputs of identical size to the layer inputs is equal to half of the corresponding initial *kernel* widths listed in Tables 3–5. The *tiles* from both the post-spreading and post-fusion/binder-deposition images are stacked together into two channels per camera, resulting in a number of channels given by $2n_{cam}$. This approach to sensor fusion allows the DSCNN to learn relationships between the data modalities. For example, the DSCNN may learn that a fused part is extant at a given pixel if there is a significant difference in the pixel intensities between the post-spreading and post-fusion images. Similarly, some anomalies may only be apparent in an infrared wavelength while others may be most prominent in the visible range.

Next, global information is captured by resizing (via bilinear interpolation) the full images associated with each layer to an image stack 32 pixels × 32 pixels in size (Fig. 3-*A*). This image stack is padded with two symmetric pixels to a size of 36 pixels × 36 pixels. Global information of interest may include extremely large-scale disturbances to the powder bed as well as global lighting levels (e.g. if the room lights were turned on or off during data collection) or powder feedstock characteristics.

Finally, the normalized $\in =[-1,1]$ (x, y) coordinates, relative to the center of the powder bed, for each pixel are also captured as a two channel image stack (Fig. 3-*D*). The relevant intuition being that certain powder bed anomalies (e.g. *incomplete spreading*) are more common in some locations of the powder bed than others. That is, some anomalies have prior probability distributions which are sensitive to their location on the powder bed and this information should be capturable by the

Table 3
Regional U-Net architecture. All units in pixels.

Layer	W_i	C_i	F	S	W_o	C_o
Down-Sample						
CONV + ReLU + BN	132	$2n_{cam}$	5	1	132	32
Crop	132	32			128	32
Max Pool	128	32	2	2	64	32
CONV + ReLU + BN	64	32	5	1	64	64
Max Pool	64	64	2	2	32	64
CONV + ReLU + BN	32	64	5	1	32	128
Max Pool	32	128	2	2	16	128
CONV + ReLU + BN	16	128	5	1	16	256
Max Pool	16	256	2	2	8	256
CONV + ReLU + BN	8	256	3	1	8	512
Max Pool	8	512	2	2	4	512
FC	4	512			1	512
Drop50	1	512			1	512
Up-Sample						
Up-sample NN	1	512			8	512
CONV + ReLU	8	512	1	1	8	512
Skip Concat	8	[512 512]			8	1024
CONV + ReLU	8	1024	3	1	8	512
Up-sample NN	8	512			16	512
CONV + ReLU	16	512	1	1	16	256
Skip Concat	16	[256 256]			16	512
CONV + ReLU	16	512	3	1	16	256
Up-sample NN	16	256			32	256
CONV + ReLU	32	256	1	1	32	128
Skip Concat	32	[128 128]			32	256
CONV + ReLU	32	256	3	1	32	128
Up-sample NN	32	128			64	128
CONV + ReLU	64	128	1	1	64	64
Skip Concat	64	[64 64]			64	128
CONV + ReLU	64	128	3	1	64	64
Up-sample NN	64	64			128	64
CONV + ReLU	128	64	1	1	128	32
Skip Concat	128	[32 32]			128	64
CONV + ReLU	128	64	3	1	128	32
Up-sample BL	128	32			T	32

Table 4
Global network architecture. All units in pixels.

Layer	W_i	C_i	F	S	W_o	C_o
CONV + ReLU	36	$2n_{cam}$	5	1	36	32
Crop	36	32			32	32
Max Pool	32	32	2	2	16	32
CONV + ReLU	16	32	5	1	16	64
Max Pool	16	64	2	2	8	64
CONV + ReLU	8	64	5	1	8	128
Max Pool	8	128	2	2	4	128
FC	4	128			1	256
Drop50	1	256			1	256
FC	1	256			1	16
Drop50	1	16			1	16
Up-sample BL	1	32			T	32

Table 5
Localization network architecture. All units in pixels.

Layer	W_i	C_i	F	S	W_o	C_o
CONV + ReLU	T + 5	$2n_{cam}$	11	1	T + 5	128
Crop	T + 5	128			T	128

DSCNN.

Non-canonically, the pixel-wise intensity u of each *tile* is centered according to Eq. (1) using a global mean intensity $\bar{u}$ determined during calibration (Section 2.2); in contrast to canonical image-wise normalization. This choice was made because, unlike in many image classification exercises, the absolute intensity of an individual pixel encodes valuable information. Effectively, this approach only centers the intensity values around zero for improved numerical stability during training. Note that (1) assumes an 8-bit image as an input; images with larger bit depths were not explored. The output of the normalization operation is stored as a 32-bit floating point number.

$$u_{normalized} = \frac{u - \bar{u}}{255} \tag{1}$$

3.4. DSCNN architecture

After *tiling*, three of the image stacks (A, B, and C) are fed into parallel neural networks. Eventually, the parallel *legs* are recombined to enable pixel-wise classification informed by the multi-scale context. For brevity, the individual hidden layers are not discussed here in detail. Instead the authors refer the reader to Section 3.4 of their previous work [19] or these original sources [32,46–48] for discussion of the different types of CNN layers.

Operating on image stack C, the regional U-Net aims to extract *features* at a medium size scale. Because much of the information at this size scale is morphological in nature (e.g. part or defect geometries), this is the deepest *leg* of the DSCNN. Table 3 enumerates the layers composing the regional U-Net and is followed by a brief discussion on the chosen hyperparameters and design choices. The design of the U-Net is heavily inspired by the architecture presented in the seminal work by Ronneberger et al. [35].

In Table 3, the columns, from left-to-right, indicate the type of network layer, the input spatial size W_i, the number of input channels C_i, the spatial size of the *kernel* F, the stride of the *kernel* S, the output spatial size W_o, and the number of output channels C_o. The number of data sources (e.g. cameras) is represented by n_{cam}. As an example, the regional U-Net begins by operating on image stack C with a convolution (CONV) layer composed 32 *kernels* (a.k.a. filters) of size 5 pixels × 5 pixels × 2 n_{cam} pixels. The CONV layer is followed by a cropping operation which removes the explicit padding added during the *tiling* operation. This was empirically found to remove edge effects at the borders between *tiles* in the final segmentation results by ensuring that none of the *features* leaving the first CONV layer are influenced by TensorFlow's implicit padding. This crop is functionally equivalent to setting the "valid" padding flag in TensorFlow. In Ronneberger et al. [35] a similar cropping operation follows every convolution layer, however this was not found to be necessary in this work.

Following the CONV layer, canonical activation (ReLU) and Batch Normalization (BN) [49] layers are applied as implemented in TensorFlow. Max Pooling is then used to reduce the dimensionality of the model and increase the effective size of the receptive fields of subsequent convolution *kernels* while preserving the strongest *kernel responses*. The output spatial sizes of the of the CONV and Max Pooling layers can be calculated according to Eq. (2); where the implicit padding P is handled by TensorFlow such that $W_o \equiv W_i$ for all strides equal to one.

$$W_o = \left(\frac{W_i - F + 2P}{S}\right) + 1 \tag{2}$$

Where W_o, W_i, F, S, and P are defined above.

The first portion of the U-Net collapses the multi-mode spatially-defined information contained within the *tile* into a single *feature* vector using a Fully Connected (FC) layer. The FC layer is followed by a canonical dropout layer (Drop50) which randomly disables different 50 % sets of the inputs into the FC layer during training to improve the robustness of the learned *features*. The second portion of the U-Net expands this *feature* vector back out to each pixel in the original 128 pixels × 128 pixels *tile*. Each expansion step is similar, but not identical, to the corresponding steps in [35] and consists of the following operations which are also shown graphically in Fig. 4.

1 The input data volume is up-sampled using a nearest-neighbor (NN) operation.
2 A CONV layer with 1 pixel × 1 pixel *kernels* is applied to reduce the number of *feature* channels in order to conserve memory on the GPU while allowing the network to preserve the encoded information in a set of more compressed/complex *features*. This operation is non-canonical, but necessary for the larger resolution tiles.
3 The data volume is then channel-wise concatenated with a copy of the corresponding data volume from the *feature* encoding side (first portion) of the U-Net. Where the corresponding data volume has the same spatial size as the up-sampled data volume. These skip connections are shown in Fig. 4 as hollow data volumes with dotted borders linked to hollow data volumes with solid borders.
4 A CONV layer with 3 pixels × 3 pixels *kernels* is applied to map the deeper *features* spatially by considering the spatial information preserved during the first portion of the U-Net. These CONV layers are also designed to collapse the number of *feature* channels after the concatenation operation increased them.
5 An activation function (ReLU) is applied at each CONV layer, but batch normalization is not used.

Taken together, these operations can be considered a "learned up-sampling operation" and in the literature are variously referred to as "deconvolution" (arguably improperly), "fractionally-strided convolution," and "up-convolution" [35]. Non-canonically for a U-Net, the DSCNN collapses all the way down to a single *feature* vector in the first portion instead of a data volume with a non-unity spatial size. This choice was made to ensure that contextual information from the entire region covered by the *tile* can be used to inform the pixel-wise classifications. While equivalent results could be achieved with the canonical implementation, because the *tile* size is one of the hyperparameters exposed to the user, this design decision improves the overall interpretability of the algorithm's behavior. Finally, the output of the U-Net is up-sampled using bilinear interpolation (BL) to the native *tile* size T pixels × T pixels.

Operating on image stack A, the global network aims to extract *features* at a large size scale which nominally describe the state of the

entire powder bed. Because the number of truly unique input datums for this size scale number in the tens to hundreds (the number of annotated powder bed layers) instead of tens of millions (the number of pixels), it is critical that this *leg* have relatively few parameters to protect its generalizability. In other words, there is a risk that insufficient variation will be captured in the training sample to ensure robustness of the network to the variations extant in the population. Similarly, reduction of the variation possible within the population drove the decision to resize this input image stack to only 32 pixels × 32 pixels. Table 4 enumerates the layers composing the global network.

The output of the global network is up-sampled from a single *feature* vector to the native *tile* size T pixels × T pixels. Just prior to up-sampling, the number of channels is reduced from 256 to 16 using a fully connected layer with the goal of emphasizing the more local information in the final pixel-wise classifications. Two dropout layers are included in this *leg* to mitigate overfitting of this portion of the model.

Operating on image stack B, the localization network aims to map the information captured and encoded at the larger size scales to the original image pixels. This *leg* of the DSCNN, combined with the ability to resize the input *tiles* in their native resolution, is responsible for ensuring the model's agnosticism to the AM machine and imaging system. Table 5 enumerates the layers composing the localization network.

The localization network is composed of a single CONV layer followed by a ReLU activation function. Because this *leg* of the network must be able to operate on input data of arbitrary spatial dimensions, no pooling operations or deeper layers are implemented. The output of the localization network is a set of *feature vectors* – one for each pixel in the original (i.e. native resolution) *tile*. Each of these *feature vectors* encodes information about the local region around each pixel as well as relationships between the post-spreading and post-fusion/binder-deposition images from one or more cameras. The following subsection describes merging of all three of the parallel *legs* of the DSCNN.

3.5. Pixel-wise classification

The fundamental intuition behind the DSCNN is to capture contextual information at multiple size scales and combine it with highly local information to enable pixel-wise classifications. To achieve this, the outputs of the three *legs* are concatenated together along with the (x, y) coordinates of each pixel (image stack D). Table 6 describes the concatenation process as well as the subsequent layers required for pixel-wise classification.

The result of the concatenation is a 178-element *feature* vector capturing context at multiple size scales for each pixel in the original *tile*. The classification layer is essentially a channel-wise *1* pixel × *1* pixel convolution operation which collapse the number of *feature* channels from 178 to the number of anomaly classes ($n_{classes}$) for the given AM machine. No ReLU activation function is applied to the output of the classification layer.

Canonically, the $n_{classes}$ long vector at each pixel may be considered a distribution of *responses* – one for each anomaly class. Ideally, if the fully trained DSCNN observes a pixel which should be labeled as anomaly class #5, it will produce a high *response* value in the element of this final vector corresponding to anomaly class #5, and low *responses* in all other elements. This distribution of class-wise responses $\{p\}$ (of arbitrary absolute magnitude) are rescaled as pseudo-probabilities $\{q\} \in (0,\ 1]$ such that $\Sigma\{q\} = 1$ using the softmax function given in Eq. (3).

$$q_j(\{p\}, j) = \frac{e^{p_j}}{\sum_{k=1}^{n_{classes}} e^{p_k}} \tag{3}$$

Where q, p, and $n_{classes}$ have been previously defined and $j \in [0,\ n_{classes})$.

Table 6
Final combination of multi-scale *features*. All units in pixels.

Layer	W_i	C_i	F	S	W_o	C_o
Concat	T	[128 16 32 2]			T	178
Classification	T	178	1	1	T	$n_{classes}$
Softmax	T	$n_{classes}$			T	$n_{classes}$

3.6. Performance optimization

The specifics of the DSCNN architecture were chosen considering factors such as the effective receptive fields of the *kernels*, layer classification time, classification performance metrics, available GPU RAM, and the total number of learnable parameters. At 13,996,612 + 179($n_{classes}$ + 1) learnable parameters (*kernel weights* [46], *biases* [46], and batch normalization distribution statistics [49]) the network is small by many modern standards and a deeper network may indeed produce superior classification results. However, it is important to keep the number of learnable parameters smaller, or at least on the same order, as the number of available labeled *targets* (Table 2). For ML algorithms, generally, any model with substantially more degrees of freedom than available training data is at risk of overfitting and losing its ability to generalize to other data sets [45,50].

GPU memory restrictions have been a particular concern throughout the DSCNN design process. For reference, each layer of ExOne Innovent data is approximately two orders of magnitude larger than the standard ImageNet image size of 256 pixels × 256 pixels × 3 pixels [44]. Finally, observe that because the *feature* vector output by the global network is identical for all *tiles* in a powder bed layer, it is only computed once per layer in the actual implementation of the algorithm.

3.7. Training

For the sake of brevity, reader-familiarity of backpropagation is assumed. A more complete discussion of this topic is available in Section 3.6 of the authors' previous work [19] as well as these original sources [32,48,51]. During training of the DSCNN all of the *kernel weights* in the CONV layers are initialized randomly from a zero-centered normal distribution with a standard deviation of 0.05 while the all of the *kernel weights* in the FC layers are initialized from zero-centered normal distributions with a standard deviation of 0.004; note that the performance of the DSCNN is not particularly sensitive to the choice of the parameters of the initialization distributions. All of the *kernel biases* are initialized with a constant value of 0.1. The totality of the training database (Section 3.2) is broken up into a set of "mini-batches," each containing a fixed number of *tiles* and a corresponding number of labeled pixels (*targets*) given by $n_t = n_{batch} \cdot T^2$. Note that not all the pixels within a training *tile* are required to be labeled. Handling of the unlabeled pixels is discussed at the end of this subsection.

During the "forward pass," a mini-batch of data are analyzed by the DSCNN – producing n_t softmax vectors $\{q\}$ each of length $n_{classes}$ and containing class-wise pseudo-probabilities. For example, consider the minimal case of a three-class problem for which the ground truth anomaly label at a particular pixel is class #2. The one-hot-encoded *target* would be $\{t\} = \{0,1,\ 0\}$ while the predicted pseudo-probabilities may be $\{q\} = \{0.3,0.3,0.4\}$. The disagreement between $\{t\}$ and $\{q\}$ is expected to initially be high. The goal of backpropagation is to reduce this disagreement in successive mini-batches by modifying the *kernel weights* and *biases* during the "backward pass." Specifically, the error E between $\{t\}$ and $\{q\}$ can be defined as a function of the set of learnable parameters Ω, and those parameters should be modified in the direction of the negative gradient of the error - $\nabla E(\Omega)$. In this work two different loss functions are implemented. Training was primarily performed using the cross-entropy loss function implemented by TensorFlow [52]. However, initial investigation of the "skeptical" loss function described by Reed

et al. [53] was also performed.

Training ML algorithms on noisy labels, that is, ground truth labels which may not always be "truly" correct, is a contemporary problem in the ML field. Reed et al. [53] offer several potential mitigation techniques from which the authors selected "hard-bootstrapping." Intuitively, this loss function considers both the noisy (potentially incorrect) ground truth label and the current (forward pass) predictions of the model when calculating the error – with more weight given to the model's prediction if the model is "confident" in that prediction. Continuing with the minimal case introduced previously, a prediction of $\{q\} = \{0.1,0.1,0.8\}$ is relatively more confident than $\{q\} = \{0.3,0.3,0.4\}$ despite both predicting class #3 as the most likely class. Generally, the model predictions at the start of training are not confident (i.e. the pseudo-probabilities for each class are similar in magnitude) and the confidence increases as training progresses. The hard-bootstrapping loss function is given by Eq. (4).

$$E(\{q\},\{t\}) = \sum_{k=1}^{n_{classes}} (\lambda t_k + (1-\lambda) z_k) \log_e(q_k + \varepsilon) \tag{4}$$

Where $\lambda \in [0,1]$ is a trust coefficient, $\{z\}$ is equal to $\{q\}$ after one-hot-encoding, and $\varepsilon = 1 \times 10^{-8}$ is needed for numerical stability as the model produces increasingly confident predictions.

As an example, consider a situation in which a human has mislabeled a pixel. The model's forward pass class prediction is likely substantially different than the ground truth label. If the model is not confident, the error will be large – driving a large update in the learnable parameters. However, if the model is confident in its prediction, the calculated error will be relatively less, and the corresponding parameter update will also be smaller. Essentially, this loss function incorporates a consistency check on the ground truth labels; it gives the algorithm permission to fit more loosely to the *training* data if intra-class inconsistencies are extant. More colloquially, the algorithm can ignore the human labeler if what the human is currently teaching the DSCNN is inconsistent with other information that the human has provided. Importantly, if λ is too small, the model may ignore the ground truths completely, effectively constructing its own self-reinforcing version of reality.

In classical gradient descent the parameter updates at mini-batch $i+1$ are computed as in Eq. (5), with the size of the update "step" controlled by the learning rate η. In this work the DSCNN was trained using TensorFlow's implementation of the Adam optimizer which is well-suited for the sparse gradients common in CV problems [54]. Unlike classical gradient descent, a unique learning rate is maintained for each learnable parameter $\omega \in \Omega$ and it is updated during training based on a moving average of the second derivative of the corresponding parameter $\eta_{i+1,\omega} = f(\omega''_i)$ [54]. Essentially, the learning rate is continually optimized to ensure stable and consistent parameter updates. The Adam step size was set to 0.0001, with the exponential decay rates of the first and second derivatives set to 0.9 and 0.999, respectively; the corresponding epsilon value was set to 1×10^{-4}. Overall training behavior and final algorithm performance appear relatively insensitive to the choice of Adam parameters – another advantage of this optimizer. After every complete cycle through the all the mini-batches (an *epoch*), the data are randomly shuffled amongst the mini-batches.

$$\Omega_{i+1} = \Omega_i - \eta \nabla E(\Omega_i) \tag{5}$$

Where Ω, η, and E have been defined previously.

As mentioned in Section 3.2, the training database is class-wise unbalanced; unmitigated, this can lead to highly detrimental artifacts. For example, if class #1 composes 90 % of the training data, the DSCNN may naively learn to always guess "class #1″ and still achieve an accuracy of 90 % – certainly not the desired behavior. To mitigate this problem, the authors multiplicatively applied class-wise balancing weights (6) to the pixel-wise losses $\{E\}$. If a given pixel is unlabeled, then the losses associated with that pixel are set to zero. To ensure stability during training, at least 10 % of a *tile* must labeled for it to be included during training. Without the class-wise balancing weights, it was observed that the rarer (in terms of number of pixels) anomaly classes (e.g. *recoater streaking* and *porosity*) were simply not learned during training or predicted at test time. With the class-wise balancing weights, even extremely rare anomaly classes, comprising only 0.1 % of the training *dataset*, are effectively learned and predicted by the DSCNN.

$$w_k = \frac{Median(\{f\})}{f_k} \tag{6}$$

Where w_k is the balancing weight corresponding to class k, f_k is the occurrence frequency of class k,and $\{f\}$ is the set of class-wise frequencies.

Recent research by Geirhos et al. [55] suggests that CNNs are vulnerable to learning texture-responsive *features* even when morphological-responsive *features* would be more robust. Indeed, the authors encountered this undesirable behavior in a subset of the ExOne M-Flex data – leading to an initial failure of the DSCNN when altered process parameters significantly modified the textural differences between *powder* and *part* pixels. To combat this bias, Gaussian noise is added to the image stacks A and C as a data augmentation step just prior to training. The noise distributions used are designed to disrupt the model's ability to rely exclusively on the image texture for classification, without altering the mean intensity of the data, and have variances of 0.01 % and 0.1 % of the input data's dynamic range. Noise is never added to image stack B to ensure preservation of some textural *features* (which may be important for classification) while forcing at least part of the DSCNN to learn low-frequency morphological *features*.

The only other data augmentation applied is a mean intensity shift of +/-10 % of the dynamic range across image stacks A, B, and C. Other canonical data augmentation techniques such as image rotation are not applicable for these datasets as they may alter the ground truth classifications. Including the case where no augmentations are applied, there are nine unique combinations of data augmentations; this results in the duplication of each training *tile* a total of nine times during training.

Mini-batch sizes of between 15 *tiles* and 50 *tiles* were used for training. The number of epochs used for training varied but was on the order of 100 for all machine types. Training a DSCNN from scratch required approximately two days whereas transfer learning a DSCNN could yield good results in less than two hours. The learnable parameters saved in the final DSCNNs correspond to the exponential moving average values [56] at the end of training with a decay rate of 0.9999.

3.8. Heuristics and final segmentation results

As in the authors' previous work [18,19], the raw DSCNN predictions can be augmented with heuristics. For the purposes of this paper, the following heuristics are enabled for some figures and are included in the reported *testing* performance numbers for the EOS M290.

1 For the EOS M290, if *debris* detections are directly on top of expected part locations, switch to *part damage*. This is justified by the definition of *part damage* provided in Section 1.2.
2 For the EOS M290, if *part*, *super-elevation*, or *swelling* detections are far away (more than 1100 μm) from expected part locations, switch to *debris*. Because *part*, *super-elevation*, and *swelling* cannot, by definition (Section 1.2), occur in the absence of intentionally fused material (*misprints* have not been observed for the EOS M290) any such DSCNN predictions are incorrect. The heuristic leverages this a priori knowledge about the process to improve the results presented to the user.
3 For the ConceptLaser M2 machine, if *part* detections are far away (more than 350 μm) from expected part locations, switch to *misprint*.

This is justified by the definition of *misprints* provided in Section 1.2. The tighter dimensional tolerance (relative to the EOS M290 heuristics) is enabled by a more accurate registration between the powder bed imaging data and the CAD geometries for this machine.

While explicit inclusion of the CAD geometry as a DSCNN input channel was considered, this idea was rejected due to concerns regarding overfitting during training and a reduction in model generalizability. For example, *misprints* may become difficult to detect if the DSCNN learns to make *part* predictions based primarily on the CAD geometry (as this is almost always positively indicative) as opposed to based on the post-fusion images. Data compression of the layer-wise segmentation results is achieved using a kernel-based row-wise connected-components algorithm that is 10–100 times more memory efficient than the algorithm described in Section 3.7 of Scime et al. [19].

3.9. Transfer learning: capabilities and limitations

Much of the focus to this point has been on how the DSCNN architecture meets the first, second, and fourth goals. Now let us consider how the architecture addresses the third goal: transferal of learned knowledge between machines. Consider machine-type #1 for which a great deal of labeled data is available and machine-type #2 for which a relatively smaller number of labeled *targets* exist. In this case, machine-to-machine transfer learning proceeds as follows.

First, a DSCNN is trained on machine-type #1 data with all learnable parameters initialized randomly. Then, a DSCNN for machine-type #2 is largely initialized using the learned parameters from the machine-type #1 DSCNN. In fact, owing to *tile* resizing and the fully convolutional nature of the localization network, only the final classification layer must be initialized randomly – under the assumption that the two machines will have different anomaly classes. In the current implementation, parameters throughout the entire depth of the DSCNN are free to be re-learned during transfer learning so overfitting remains a theoretical concern. While overfitting was not observed for any of the presented datasets, resistance to overfitting can be improved by freezing the lower DSCNN layers during transfer learning, effectively reducing the number of degrees of freedom of the model. Note that all of the batch statistics learned by the Batch Normalization layers are also reset during transfer learning as they are expected to be substantially different between different machine types.

Additional care must be taken when transferring knowledge between two machines with different numbers of camera systems. As an example, if machine-type #1 has one camera (α) and machine-type #2 has two cameras (α and β), the *weights* $\omega_{2\alpha}$ and $\omega_{2\beta}$ (for both the post-fusion and post-spreading channels) will be initialized by the *weights* $\omega_{1\alpha}$ from machine-type #1. Critically, because the number of channels of the data volume produced by the first CONV layer is fixed by the number of *kernels* (Table 3), it is invariant to the number of cameras – again preserving the machine-agnostic architecture of the deeper layers.

While the presented architecture is nominally transferable to an infinite variety of powder bed AM machines and imaging configurations, certain practical limitations do exist. Specifically, while the native size of the *tiles* can be freely modified to ensure that they contain relevant contextual information, they are always resized at the first CONV layer. During this resizing operation, important small-scale information may be lost. Furthermore, the receptive field of the *kernels* in the CONV layer of the localization CNN may become too small (at the native resolution) to provide meaningful discrimination between pixels. These considerations place a theoretical limit on the maximum difference in native imaging resolutions which can be bridged via transfer learning. While this limit has not yet been identified by the authors, it is anticipated that attempting to transfer a DSCNN from a 1 M P camera to a system with a resolution greater than 50 M P may be substantially challenging.

4. Results: algorithm performance

4.1. Validation and testing accuracy

Canonically, the prediction performance of an ML algorithm is evaluated by splitting the sample of available data (with ground truth labels) into *training*, *validation*, and *testing datasets* [57]. Where the ML model is “fit to” the *training dataset*, the model architecture is tuned by evaluating the *validation dataset*, and the *testing dataset* is used to confirm model generalizability [50]. In this work, the *validation dataset* consists of a random 10 % *tile*-wise withholding of the training database reported in Table 2. The *testing dataset* consists of fully annotated layers of powder bed imaging data (also never seen by the DSCNN during training) with the number of labeled layers varying by machine. Note the following canonical deviations:

1 Minimal hyperparameter tuning has been performed using the *validation dataset* at this time, therefore some conclusions about generalizability can still be drawn from the *validation* performances.
2 Because it is difficult to capture the full variety present within the population via the relatively small *testing datasets*, the *validation* performance may be more representative of DSCNN performance for some anomaly types.
3 Due to the nature of the global *leg* of the DSCNN, some of the information contained within the *validation dataset* will have been seen during training.

A variety of methods for calculating prediction accuracy for segmentation tasks are available, including intersection over union [31], considering groups of same-label pixels to be “objects” and prioritizing their detection over their exact boundaries, and naive pixel-wise prediction accuracy. All these methods have benefits and limitations; however, the authors feel that simple, direct comparison of the pixel-wise predictions to the pixel-wise labels quantifies the DCNN performance in the most meaningful terms.

Table 7 reports the *training* and *validation* performance of the DSCNN across the multiple machine types. As in the authors’ prior work [18,19], the *validation* performance is often higher than the *testing* performance in part because the human-labeler is typically reticent to include ambiguous data in the training database (and therefore the *training* and *validation datasets*) while the construction of the *testing dataset* forces the inclusion of all data in a given layer. For machine types without substantial *testing datasets*, only the *validation* performances are reported. Table 7 also shows a direct comparison to the testing performance of the authors’ previous MsCNN [19] on the same testing dataset.

While final training of the DSCNN for each machine was performed with the data specified in Table 2, the transfer learning process, by design, implies that DSCNN *features* may be learned on data from multiple machines. Because of the limited *training* data available for the EOS M290, Renishaw AM250, and Arcam Q10 machines, their DSCNNs were transfer learned from the ConceptLaser M2 DSCNN and therefore had the opportunity to benefit from that machine’s qualitatively higher-quality *training* dataset. Note that the *testing* data for the Renishaw AM250 and Arcam Q10 machines are sourced from the same build available for the training database. Also, note that the MsCNN *testing* performance reported above differs from that reported in Scime et al. [19]. This discrepancy is due to differences in the ground truth labels:

1 In the previous work the MsCNN only needed to make accurate predictions *patch*-wise, a much less aggressive performance metric than pixel-wise predictions.
2 In the current *testing dataset*, the entirety of each layer of powder bed imaging data is labeled, including all potentially ambiguous pixels. This was done to further reduce the potential for biasing the performance results but depresses the reported accuracy.

Table 7
Class-wise *validation* (top) *testing* (bottom) performances for the six machine types. *Testing* results are only available for the EOS M290, ConceptLaser M2, Renishaw AM250, and Arcam Q10 machines and are based on 5, 5, 1, and 1 fully-labeled layers, respectively. True positive rates are reported first, followed by false positive rates in parentheses. All values are percentages with perfect performance for a given class defined as 100.0 (0.0).

	EOS M290		ConceptLaser M2		Renishaw AM250	Arcam Q10	ExOne M-Flex	ExOne Innovent
Anomaly Classification	MsCNN	DSCNN	standard	skeptical				
Powder	– 90.3 (2.5)	99.5 (1.6) 99.0 (1.4)	99.0 (1.2) 98.5 (1.1)	99.3 (0.4) 98.6 (1.2)	98.9 (1.4) 98.6 (1.1)	97.6 (2.4) 97.9 (1.1)	98.4 (2.7) -	99.8 (1.4) -
Part	- -	97.1 (9.2) 83.5 (17.0)	98.3 (3.8) 96.3 (6.6)	98.5 (2.6) 96.1 (6.2)	99.3 (1.0) 99.7 (2.6)	97.8 (2.2) 98.5 (2.7)	97.1 (8.9) -	99.8 (1.3) -
Recoater Hopping	– 92.9 (90.9)	82.9 (10.0) 70.3 (35.1)	–	–	–	–	–	–
Recoater Streaking	– 49.7 (74.0)	57.9 (8.8) 31.5 (41.2)	39.4 (38.6) 30.2 (80.7)	73.8 (20.4) 31.5 (79.0)	–	–	60.7 (33.1) -	N/A[a] -
Incomplete Spreading	– 74.1 (24.5)	92.4 (2.8) 95.7 (5.5)	71.6 (5.1) 92.4 (3.9)	99.4 (1.1) 92.6 (4.0)	–	–	97.4 (7.2) -	83.2 (0.1) -
Swelling	–	75.3 (29.5) 68.5 (57.0)	65.7 (26.1) 60.4 (31.2)	79.2 (15.0) 58.9 (31.5)	75.1 (9.7) 56.7 (52.6)	–	–	–
Spatter	–	–	–	–	47.1 (43.6) 49.0 (35.0)	–	–	–
Soot	–	- -	17.5 (74.9) 63.9 (39.6)	73.5 (31.0) 64.2 (38.5)	–	–	–	–
Debris	– 82.7 (71.5)	72.1 (14.5) 67.3 (28.1)	78.6 (20.6) 67.5 (29.1)	89.1 (12.2) 63.8 (27.4)	–	–	15.7[b] (18.1) -	–
Super-Elevation	– 92.0 (54.4)	86.4 (5.0) 81.4 (5.2)	72.1 (64.5) 93.0 (17.5)	98.4 (11.8) 92.3 (11.7)	–	–	–	–
Part Damage	– 90.5 (63.3)	– 53.9 (49.7)	–	–	–	–	–	–
Porosity	–	–	–	–	–	64.6 (32.9) 78.4 (43.3)	–	–

[a] Because few labeled *tiles* containing *recoater streaking* exist in the training database, none of them made it into the *validation dataset*.
[b] There is currently a large degree of visual ambiguity and human-labeler uncertainty regarding the *debris* class.

Directly comparing the performance of the MsCNN [19] to the DSCNN is challenging as the former produces *patch*-wise classifications and the latter produces *pixel*-wise classifications. For example, the true-positive rate for *recoater hopping* in the EOS M290 drops from 92.9% to 70.3% because if the MsCNN observes any *recoater hopping* in a given *patch*, all of the pixels within that *patch* are labeled as *recoater hopping*. Therefore, it is important to also observe that the corresponding false-positive rate improves from a staggering 90.9 % to only 35.1 %. The results are similar for the other small-scale anomaly classes. For the larger scale *incomplete spreading* class, the true-positive rate increases from 74.1% to 95.1% and the false-positive rate also improves from 24.5% to 5.5 %.

4.2. Effects of inter-machine transfer learning

While a complete study of the impact of inter-machine transfer learning on DSCNN performance must be left to future work, its effects have been studied here for one scenario. In this experiment, a DSCNN for the ConceptLaser M2 machine was trained for only 10,000 (31 epochs) mini-batches (approximately eight hours) both with and without transfer learning from the fully trained ExOne Innovent DSCNN (Table 7). The *training* (Table 2), *validation* (Table 2), and *testing* data used were the same as those used for the ConceptLaser M2 DSCNN reported in Table 7. These two AM machines were chosen for this experiment because the ConceptLaser M2 database is qualitatively the most diverse and the ExOne Innovent database is large and its DSCNN was trained without transfer learning.

A comparison between these two cases is presented in Table 8. After 10,000 mini-batches, the *testing* loss for the DSCNN initialized using transfer learning is only 16 % of the *testing* loss measured for the DSCNN initialized without transfer learning. This indicates that the training process has progressed comparatively further for the transfer learned DSCNN –an expected outcome, based on the behavior of transfer learning for other CNNs in other problem domains. Table 8 also reports the class-wise averages of the true positive and false positive rates for both the *validation* and *testing* datasets, and here, the effects of transfer learning become more opaque. Both the class-wise average *validation* and *testing* true positive rates are worse for the transfer learned DSCNN by 7.6 % and 5.4 %, respectively. In contrast, the class-wise average *validation* and *testing* false positive rates for the transfer learned DSCNN are better by 6.0 % and 5.5 %, respectively.

This discrepancy between the performance metrics and the apparent ambiguity regarding the efficacy of transfer learning are a direct result of the class-wise imbalance in the datasets combined with the spatially small size of many of the classes in this this problem domain (i.e. images of AM powder beds). Essentially, during early stages of training, the DSCNN has not yet learned to properly localize many of the anomaly classes and vastly overpredicts their spatial size on the powder bed. This behavior has been noted by the authors across machine types and under multiple training configurations. The inflation of many anomalies' spatial sizes increases the true positive rates for these anomaly classes, and while it does suppress the true positive rate for the "surrounding" classes such as *powder* and *part*, those class are so much more prevalent (Table 2) that the decrease is not commensurate. As a

Table 8
A performance comparison between the ConceptLaser M2 DSCNN trained with and without transfer learning from the ExOne Innovent DSCNN.

	Without Transfer Learning	With Transfer Learning
Training Loss (no units)	1.35	0.22
Validation True Positive Rate	77.3 %	69.7 %
Validation False Positive Rate	44.9 %	38.9 %
Testing True Positive Rate	84.0 %	78.6 %
Testing False Positive Rate	38.0 %	32.5 %

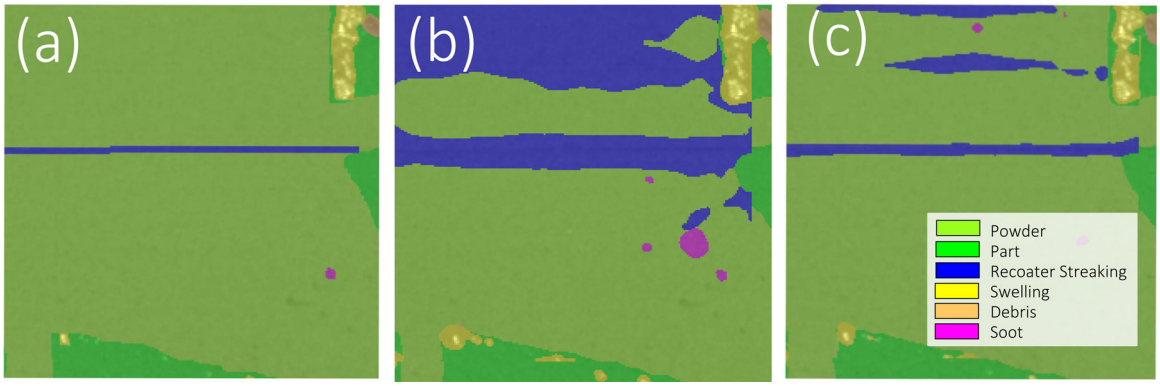

Fig. 5. Comparison of the (a) *testing* ground truths, (b) segmentations from the ConceptLaser DSCNN without transfer learning, and (c) segmentations from the ConceptLaser DSCNN with transfer learning. The gross spatial overprediction in the no transfer learning segmentation is particularly apparent for the *recoater streaking* and *soot* classes but is also visible for the *swelling* class. The additional instances of *recoater streaking* shown in (b) and (c) are discussed in Section 4.3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

result, the true positive rates for most classes can actually decrease with increasing training time (even when the model is not overfitting) while the average false positive rate will improve. Fig. 5 illustrates this unique scenario with one of the *testing* images. The challenges associated with measuring DSCNN performance under these conditions are barriers to efficient hyperparameter tuning and are discussed further in the following subsection.

4.3. Improving algorithm accuracy

As may be expected, significant ambiguity exists in the ground-truth labeling of the anomalies. Specifically, the precise spatial boundaries of a class may be poorly defined, even for a content expert. For example, the reported true-positive rate for *porosity* detection in the Arcam Q10 machine is 78.4 %, but if labels within two pixels of a ground truth class interface are ignored, that performance rises to 89.5 %. The class interfaces are calculated by applying a distance transform [58] to the binary mask of each class. Ambiguity also exists between visually similar anomaly classes such as *debris* vs. *soot* and *recoater streaking* vs. *powder*; furthermore, the human annotators are not immune from making mistakes during the ground truth labeling process. As introduced in Section 3.7, the skeptical loss function may mitigate these limitations. Table 7 reports the performance of the ConceptLaser M2 DSCNN when trained with both the standard and the skeptical loss functions. In both cases the DSCNN was trained until the *training* loss stopped decreasing. A trust factor (λ) of 0.9 was used for the skeptical loss function. Observe that the reported *validation* performances are either comparable or higher for all classes when the DSCNN is trained with the skeptical loss function. In particular, the true-positive rates for the *recoater streaking*, *incomplete spreading*, *swelling*, *soot*, and *super-elevation* classes increased by between 13.5 % and 56.0 % (percentage points). Relatedly, after careful examination, the false positive rate for *recoater streaking* using the skeptical loss function (79.0 %) is likely inflated as the DSCNN appears to correctly identify many instances of *recoater streaking* which were missed by the human labeler; Fig. 5c may show an example of this.

As might be expected, the performance improvements were greatest for the anomaly classes with the most subjective boundaries. However, the improvement shown for the *super-elevation* class is less expected as its boundaries are relatively unambiguous. Upon closer inspection, 65.6 % of the *validation* false-positives for the *super-elevation* class were misidentified as *incomplete spreading*. *Incomplete spreading* and *super-elevation* are commonly spatially co-located in this machine type and the standard loss function may have trouble learning a model to delineate the two classes at their interface. Interestingly, the *testing* performances were virtually identical across all the anomaly classes for the two loss functions. This may due to the larger variety contained within the *validation dataset* relative to the *testing dataset*. This behavior must be investigated in future work, along with the sensitivity of DSCNN performance to the trust factor.

The size and composition of the *training dataset* can significantly impact the performance of the algorithm; similarly, the quantification of this performance can be highly dependent upon the composition of the *testing dataset*. While the hundreds of millions of labeled pixels reported in Table 2 are sufficient for the DSCNN to learn the various anomaly classes, the robustness of any neural network-based model is dependent upon how well the variation present within the population is represented by the sample comprising the *training dataset*. That is, it is imperative that the *training dataset* contain the “rare” events which may occur during *testing* (use) of the algorithm. This behavior is the result of the inherent non-linearity of neural networks, which can allow small perturbations in the input data to propagate through the network and result in substantial changes to the model predictions. To address this fundamental challenge, the authors have made the DSCNN and associated annotation tools available to machine operators and AM content experts at the MDF. Users of the software tool are able to seamlessly view DSCNN analysis results for each new printer build, flag layers which were improperly segmented, and iteratively update the DSCNN. It is hoped that by broadening access to these tools, additional ground truth data can be labeled to improve both the algorithm performance as well as the quantification of that performance. Expanded *datasets* will also enable future studies on hyperparameter tuning, machine to machine transfer learning, and interpretation of the DSCNN behavior.

Finally, algorithm performance is also strongly dependent upon the quality of the input data. If an anomaly is not visually apparent within the powder bed images, it is unlikely that the DSCNN can detect it. For example, while the top-down lighting used in the ConceptLaser M2 machine provides excellent *powder* vs. *part* contrast, it generally does not produce the stark shadows typical of side-lighting configurations (e.g. the EOS M290 machine). As a result, certain anomalies such as *incomplete spreading* and *recoater streaking* contrast poorly with the anomaly-free *powder*. While beyond the scope of this manuscript, improvements to the lighting and imaging configurations of the herein machines are expected to improve algorithm accuracy for multiple anomaly classes.

4.4. Computational burden

All the calculations reported in Table 9 were performed on a desktop computer with an NVIDIA Quadro P4000 graphics card and an eight-core 3.70 GHz processor. The reported layer analysis times only include *tiling* of the input images, DSCNN inference, and reconstruction

Table 9
Computation burdens.

AM Machine	sec. per layer	sec. per 10^6 pixels
EOS M290 (MsCNN)	6.3[a]	4900
EOS M290 (DSCNN)	0.5	0.7
ConceptLaser M2	2.3	0.4
Renishaw AM250	1.2	0.5
Arcam Q10	1.5	0.6
ExOne M-Flex	1.2	0.3
ExOne Innovent	2.4	0.3

[a] The original mini-batch size of 256 was used for the MsCNN, preventing full utilization of the graphics card.

of the *tiled* results. These operations use only a single CPU core and the GPU.

The computational speed increase between the MsCNN presented by the authors in [19] is immediately apparent, with the layer analysis time improving by a factor of twelve and the time per unique model prediction improving by up to four orders of magnitude. These dramatic improvements can be attributed to several factors including the shallow and parallelized DSCNN architecture, direct control over all calculations and memory handling, and the use of the localization network to return the pixel-wise predictions.

Also, observe that the time per unique classification is not independent of the input image size (Table 1). Indeed, doubling the number of input pixels generally results in less than a doubling in layer analysis time. While this behavior is dependent upon the choice of native *tile* size, this scalability was another important motivation for the use of a localization network. At the current camera resolutions, the DSCNN returns classifications at rates suitable for real-time monitoring of all the tested powder bed AM machines. However, additional improvements may be required as camera resolutions exceed 50 M P.

4.5. Qualitative evaluation (EOS M290)

Fig. 6b shows the DSCNN predictions overlaid on top of a post-spreading image from the EOS M290 machine, the same layer as shown in Fig. 13 in [19]. Fig. 6a shows the MsCNN predictions for the same layer. Several improvements in the DSCNN predictions over the MsCNN predictions are immediately apparent:

1 Vastly improved localization of *incomplete spreading*, *debris*, *recoater streaking*, and *recoater hopping*.
2 Less confusion between *incomplete spreading* and the *powder* and *debris* classes.
3 Detection of some *debris* occurrences (left-hand side of the powder bed) which are only visible in the corresponding post-fusion image.
4 Detection of additional anomaly classes including *part* and *swelling*.
5 More semantically consistent *super-elevation* detections.

5. Results: powder bed machines

5.1. ConceptLaser M2

Fig. 7 shows several examples of anomaly classification results for the ConceptLaser M2. Subfigures (a) and (b) report segmentation results for the same layer, overlaid on top of the post-fusion and post-spreading images, respectively. Observe that the *part* detections are, as expected, closely tied to information in the post-fusion images. *Super-elevation* is correctly predicted in locations where the fused part geometry is still visible in the post-spreading image. Because *incomplete spreading* is annotated using the post-spreading images in the training database, the predicted boundary more closely matches that in (b) rather than (a). Fig. 7c shows predictions of *swelling* and *debris*; in this case the *debris* is due to improper bonding between the part and the support structures. Because the *debris* moves after powder spreading, it is detected in different locations in the post-fusion and post spreading images. Finally, Fig. 7d shows *soot* detections surrounding and covering several test blocks.

Fig. 8a shows several as-printed *misprints* which are visible as triangular "ledges" with a thickness of a single layer. Fig. 8b shows the three-dimensional reconstruction of the DSCNN classifications for this build. Disagreements between the intended part geometry and the detected part geometry are highlighted in dark orange while *part* detections are shown in green. Qualitative agreement between the DSCNN *misprint* classifications and the as-printed part is excellent.

Fig. 9 shows a heat map of cumulative *swelling* predictions throughout the height of a ConceptLaser M2 build. Brighter regions on the heat map indicate locations with consistent *swelling* detections. The brightest regions correspond to the edges and corners of the printed parts, suggesting that these local geometries are the most prone to

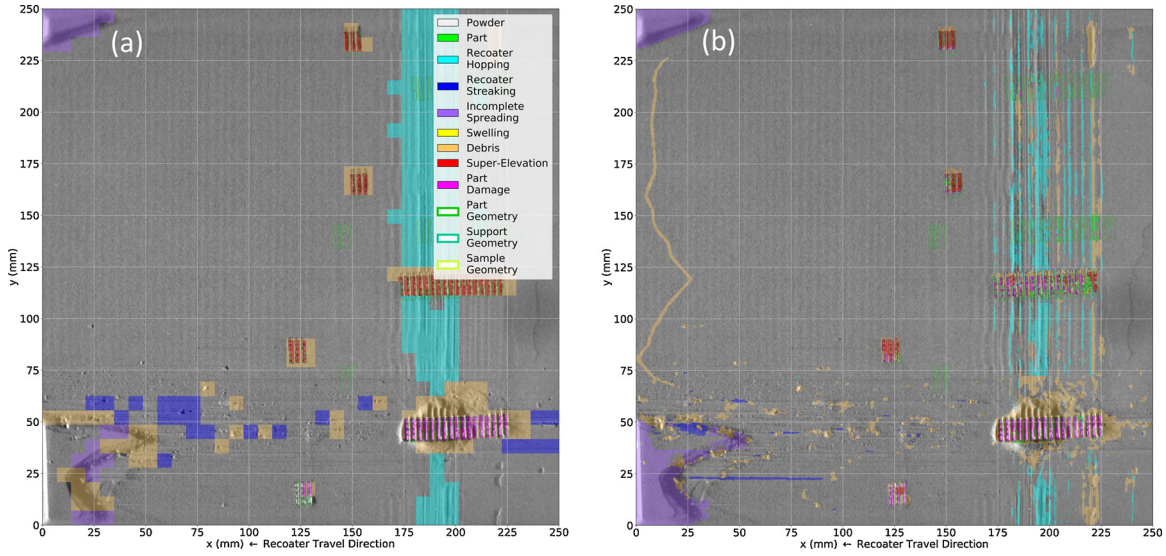

Fig. 6. Comparison of (a) MsCNN and (b) DSCNN [19] predictions for a layer of EOS M290 powder bed imaging data overlaid on top the post-spreading image. Part geometries are recovered from the CAD data. Several of the "part geometries" are filled in with *powder* classifications instead of *part* classifications because those parts were deactivated by the machine operator earlier in the build but remained in the CAD file. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

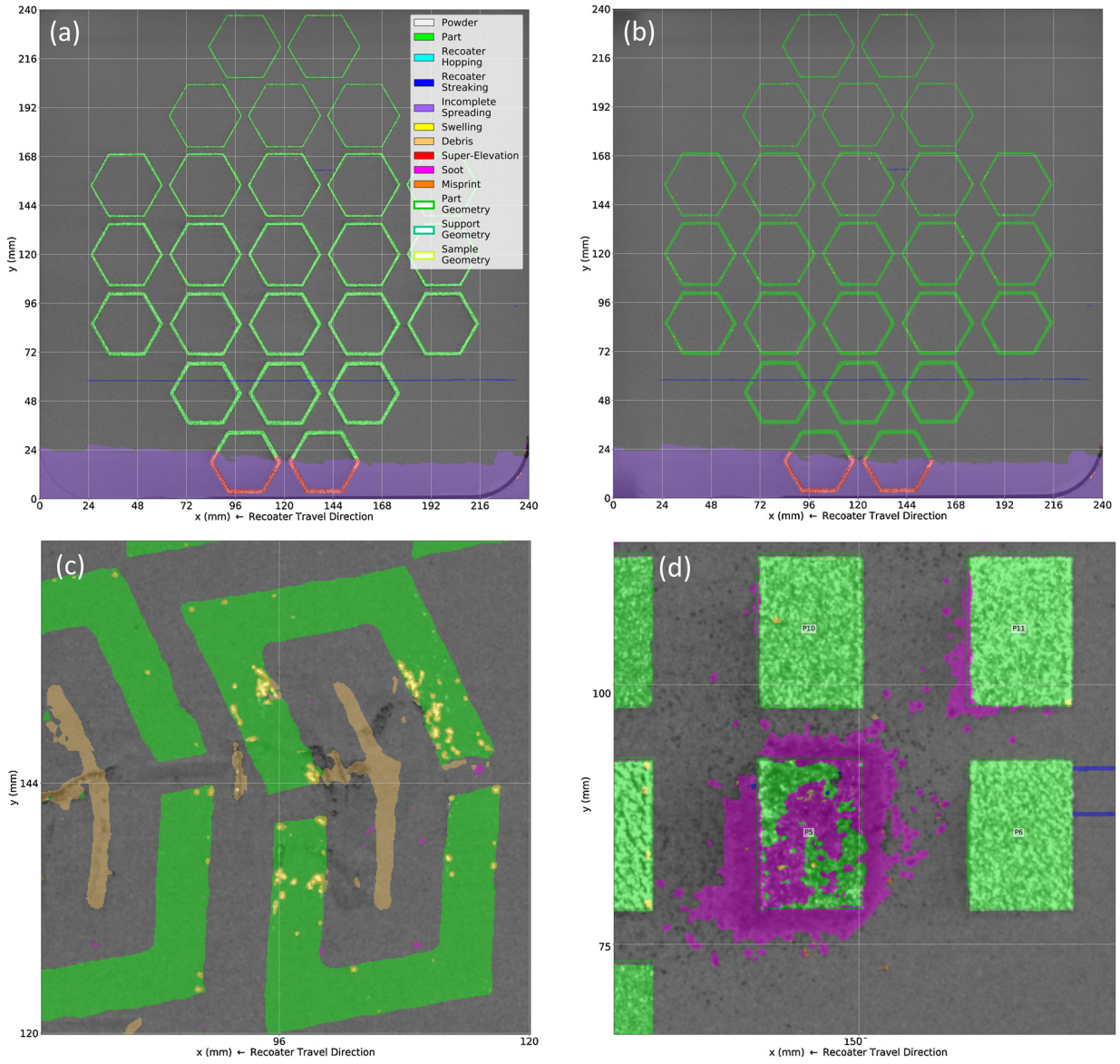

Fig. 7. DSCNN predictions are overlaid on ConceptLaser M2 post-fusion images in (a) and (d) while post-spreading images are shown in (b) and (c). Note that (c) and (d) are zoomed in from the full FoV to highlight the regions of interest. Part outlines are recovered from the CAD data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

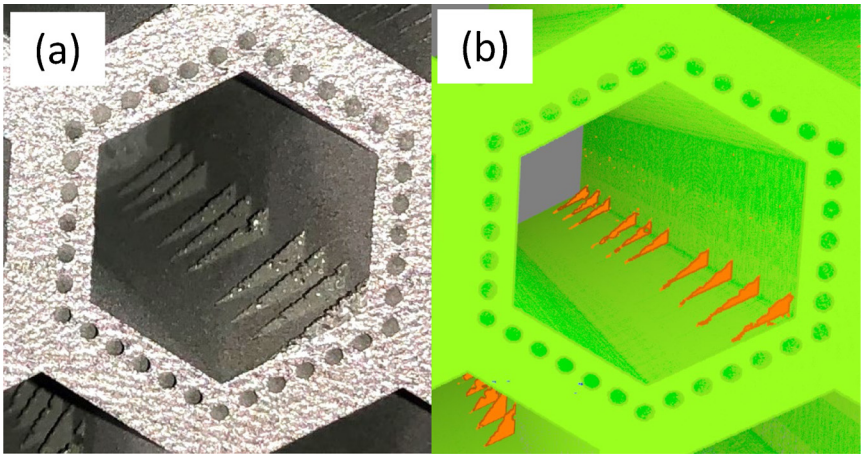

Fig. 8. Comparison of *misprint* anomalies as (a) observed ex-situ and (b) detected by the DSCNN. (For interpretation of the color in this figure, the reader is referred to the web version of this article).

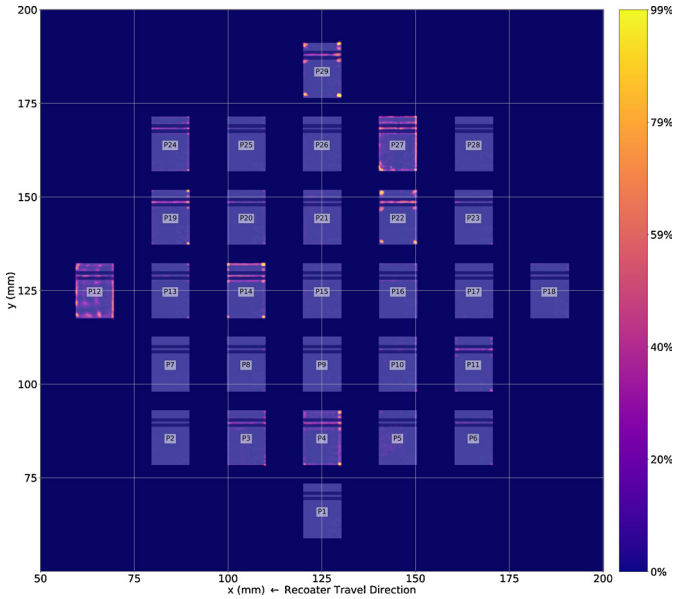

Fig. 9. A heat map of cumulative *swelling* predictions throughout the height of a ConceptLaser M2 build. The color bar indicates the percentage of layers (out of the full build height) at which a given pixel (in the *x*-*y* plane) is predicted as *swelling*. Silhouettes of the intended CAD geometry are also visible along with experiment-specific annotations. (For interpretation of the color in this figure, the reader is referred to the web version of this article).

either thermal warping or swelling. This predilection is supported by the literature [15] and has been previously observed by the authors [59]. Such behavior is often attributed to an increase in energy density as the melt tracks shorten and the possible heat conduction paths are truncated near the part edges and corners [15]. Variation in the degree of *swelling* between blocks is the result of varying melting process parameters.

5.2. Domain transfer

A key requirement of the algorithm is the transfer of learned knowledge between powder bed AM machines. Section 3.9 addresses the mechanics underlying the transfer learning process. As an example, transferring the lower-level DSCNN *features* learned on the ConceptLaser M2's training database to the Renishaw AM250 machine proceeded as follows:

1 A content expert (e.g. a machine operator) determines the types of anomalies that the Renishaw AM250 DSCNN should detect.
2 Camera calibration for the Renishaw AM250 proceeds as described in Section 2.2, requiring approximately five minutes.
3 Several layers of representative imaging data are manually annotated as described in Section 3.2, requiring approximately three hours.
4 Transfer learning from the ConceptLaser M2's DSCNN proceeds as described in Section 3.9, requiring approximately two hours.
5 An initial Renishaw AM250 build of several hundred layers is analyzed with the transfer-learned DSCNN, requiring approximately fifteen minutes.

In total, the entire process of analyzing a build from a "brand new" AM powder machine takes the user less than a day, start-to-finish. This time estimate assumes that sufficient data have already been collected and of course more labeled data can be expected to improve the model predictions.

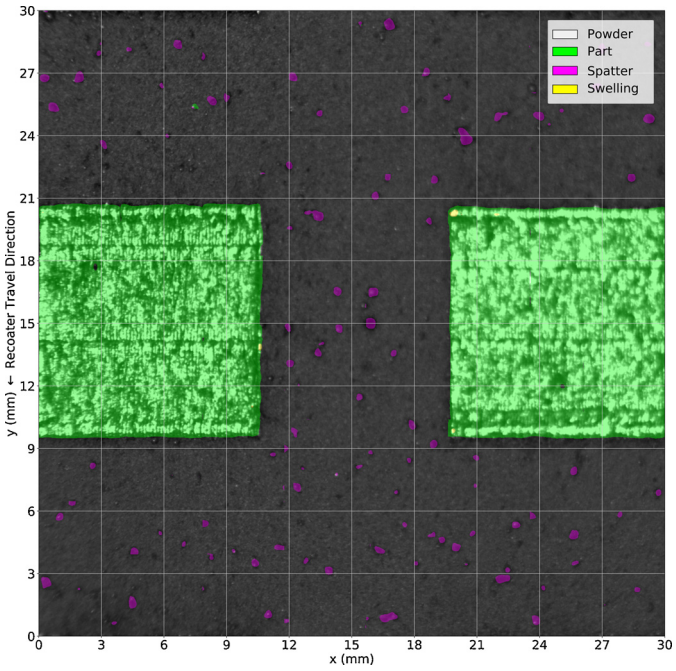

Fig. 10. DSCNN predictions are overlaid on a Renishaw AM250 post-fusion image. The reported dimensions are approximate. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

5.3. Renishaw AM250

Fig. 10 shows an example of anomaly classification and segmentation results for the Renishaw AM250 overlaid on a post-fusion image. Observe that the higher effective resolution of the camera as well as the top-down lighting allows for robust *part* segmentation. Again, the *swelling* predictions occur at pixels for which the part is still visible in the post-spreading image.

5.4. ExOne M-Flex

Fig. 11 shows an example of anomaly classification and segmentation results for the ExOne M-Flex overlaid on a post-spreading image. The *part* predictions are based primarily on information contained in the corresponding post-binder deposition image. Detection of *incomplete spreading* is robust, while some instances of *debris* are not correctly predicted – possibly because *debris* exists in only a few layers of training data at this time and it is difficult for the human to maintain consistency when labeling *debris*.

Fig. 12 reports the percentage of pixels (within a region of interest) belonging to each anomaly class as a function of layer number for a build on the ExOne M-Flex. Observe the sudden change in the detections of the *part* class at layers 346, 357, and 371. This change, which can also be detected in real-time by a simple control chart rule, indicates that a *jet misfire* has occurred.

5.5. ExOne innovent

Fig. 13 shows an example of anomaly classification and segmentation results for the ExOne Innovent overlaid on a post-binder deposition image. Observe the detections of *recoater streaking* across the top of the powder bed. This is most likely due to a contaminate or clumps of powder particles stuck to the recoating roller.

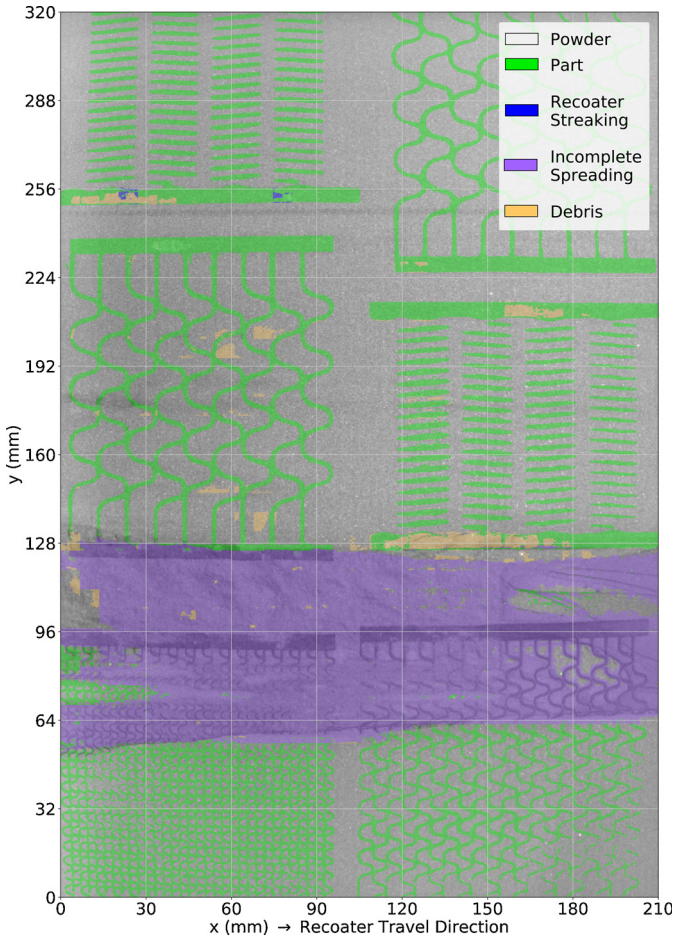

Fig. 11. DSCNN predictions are overlaid on an ExOne M-Flex post-spreading image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

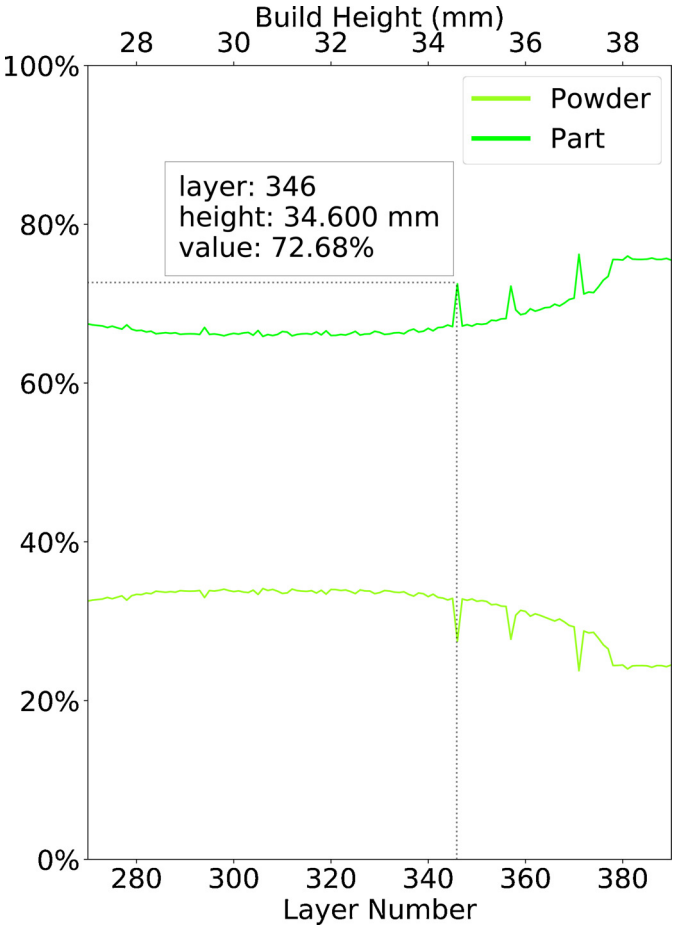

Fig. 12. A plot of the occurrence of each ExOne M-Flex anomaly class as a function of the build height. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

5.6. Arcam Q10

Fig. 14 shows an example of anomaly classification and segmentation results for the Arcam Q10 overlaid on a post-fusion image. Observe the scattering of *porosity* detections throughout the presented layer, particularly toward the center of the build area. Also observe some potential *tiling* artifacts (apparent "holes" in the *part* classifications) toward the center of the image; additional training data at *tile* interfaces might remove these artifacts. Additionally, as can be seen in Fig. 15, the porosity detections steadily decrease through the first 300 layers of the build while the part geometry remains constant. Based on unpublished ex-situ measurements of this build, this predicted porosity trend is considered accurate.

6. Discussion and conclusions

Overall, the DSCNN performed admirably, addressing each of the four goals:

1 With layer-wise segmentation times of 0.5 s – 2.4 s on a desktop computer, results are returned rapidly enough for real-time analysis of in-situ data from all the explored machine types.
2 Anomaly classifications are provided pixel-wise, at the native resolution of each imaging system. This is a dramatic step forward in localization accuracy compared to the *patch*-wise classifications of the authors' previous works.
3 The DSCNN was successfully demonstrated on data from nine different powder bed AM machines (six of which were presented) spanning three distinct technologies including L-PBF, binder jetting, and EB-PBF. Furthermore, the imaging systems explored included visible-light, MWIR, and NIR cameras with sensor sizes ranging from 1 M P – 20 M P and effective resolutions ranging from 20 μm to 290 μm. Significantly, the DSCNN allowed learned knowledge to be seamlessly transferred between different powder bed machines, thereby reducing the data collection and manual data labeling burden associated with each individual machine.
4 Sensor fusion of two different imaging modalities was demonstrated for the ExOne M-Flex system. The sensor fusion capabilities of the DSCNN will be more thoroughly explored in a separate publication.

In addition to achieving the four primary goals, the DSCNN also realized several improvements over the authors' previous MsCNN. For the EOS M290 dataset, the false-positive rates dropped for all the anomaly classes, and by more than 25 % (percentage points) for four of the seven classes. The true-positive rates improved for larger-scale anomalies but dropped for several of the smaller-scale anomalies due to the inherent difficulty of pixel-wise prediction relative to *patch*-wise prediction. Furthermore, layer-wise classification times dropped by a factor of twelve while the improved localization capability and the inclusion of the post-fusion image enabled the addition of the *part* and *swelling* classes.

Across AM machine types and anomaly classes, the *validation* true-positive and false-positive rates ranged from 99.8 % – 15.7 % and 0.4%–43.6%, respectively. While the *testing* true-positive and false-positive rates ranged from 99.7 % – 31.5 % and 1.1%–57.0%, respectively. In general, the anomalies most challenging for the DSCNN to detect fell

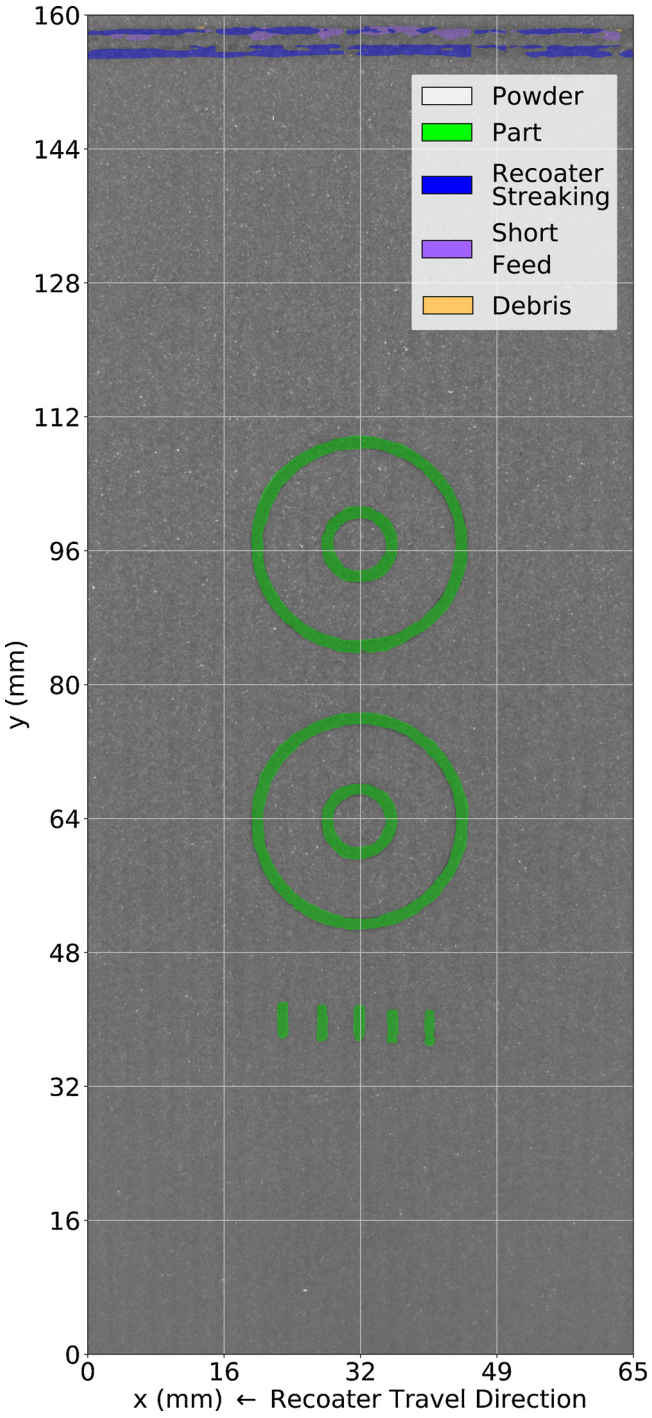

Fig. 13. DSCNN predictions are overlaid on an ExOne Innovent post-binder deposition image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

into one or more of the following categories:

1 Relatively spatially small, such as *recoater streaking* and *swelling*.
2 Poor contrast with the surrounding pixels due to the imaging configuration.
3 Nebulous or ambiguous boundaries between anomaly classes affecting the labeling of both *training* and *testing* data. This is a particular challenge for the *soot* and *spatter* classes in the ConceptLaser M2 and Renishaw AM250 datasets and *debris* in the ExOne M-Flex dataset.

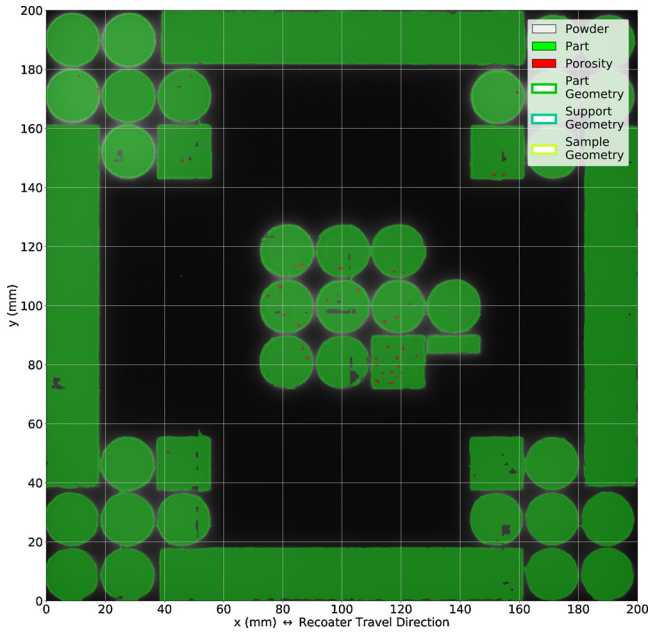

Fig. 14. DSCNN predictions are overlaid on an Arcam Q10 post-fusion image. The dark regions without a color overlay are unfused powder. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

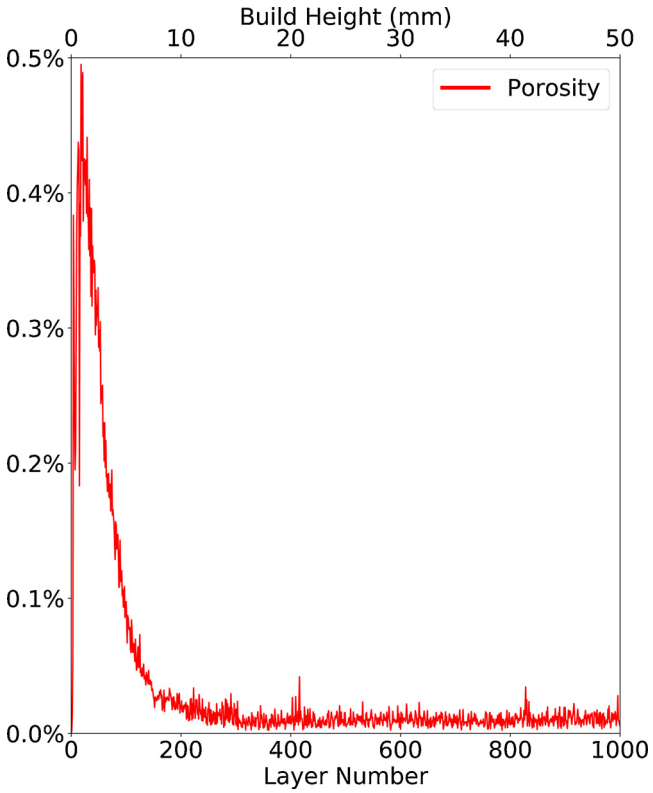

Fig. 15. A plot of the occurrence of the Arcam Q10 *porosity* anomaly class as a function of the build height.

Additional hyperparameter tuning may improve algorithm performance. However, meaningful improvement will require the development of more robust performance metrics. Indeed, one challenge of quantifying algorithm performance was demonstrated through the discussion of interface ambiguity for the *porosity* anomaly class. Another challenge is capturing the true variation found across the

population via the *testing dataset*. Full quantification of the benefits of machine to machine transfer learning will similarly require more extensive *testing datasets* (as transfer learning is expected to primarily improve the generalizability of the learned models) and more highly engineered performance metrics. Currently, the ConceptLaser M2 *dataset* is the most robust as it is constructed from the largest number of unique layers and builds and has the most extensive set of anomaly classes. Robust evaluation of the efficacy of transfer learning will require a second *dataset* of a similar caliber (which can then be subsampled) as well as a corresponding high-quality *testing dataset*. This study is left for future work and is expected to use data from the AddUp FormUp 350 L-PBF machine at the MDF. Skeptical learning was also briefly investigated and shown to dramatically improve the *validation* performance for several anomaly classes. Its effect on the *testing* performances appeared to be minimal – a behavior warranting further investigation.

The case studies provided in Section 5 suggest significant potential for post-build investigations beyond that already discussed by the authors in [18,19]. In particular, the inclusion of post-fusion/binder deposition imagery allows for detection of part-geometry related defects such as *misprints* and *jet misfires* as well as anomalies typically obscured by the powder layer such as *soot* and *spatter*. Additionally, the much finer discriminating power of the DSCNN may enable correlation of powder bed anomalies with processing conditions. For example, observe the variation in *swelling* detections across the 29 sample blocks in Fig. 9. This high detection fidelity is also apparent in Fig. 15, where the *porosity* detections show robust discrimination on a vertical axis with a maximum value of only 0.3 % of the entire powder bed.

Over the last six months, several hundred builds at ORNL have been analyzed using the DSCNN. This volume of data has significant implications for the application of data analytics and advanced visualization techniques to the DSCNN inference results. While the datasets used herein cannot be publicly released verbatim, example datasets, along with a compiled software tool and the associated instruction manual, will be publicly released in 2020. Future publications will explore drawing conclusions from layer-wise image data both by itself and in combination with other registered data modalities. Future work will also explore sensor fusion with different imaging wavelengths, lighting conditions, and certain spatially mapped sensor modalities. Finally, while the primary purpose of the DSCNN is the eponymous semantic segmentation of powder bed imaging data, the authors remain cognizant of the desire to utilize real-time process monitoring to achieve real-time process control for powder bed AM. Another publication will describe early successes in this arena for one AM powder bed technology.

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05 − 00OR22725 with the US Department of Energy (DOE). Research was sponsored by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Advanced Manufacturing Office and by the Office of Nuclear Energy. Funding was also provided by the Transformational Challenge Reactor (TCR) program. It would be inappropriate for anyone currently affiliated with ORNL's Manufacturing Demonstration Facility to review this work. The lead author also recently graduated from Carnegie Mellon University and therefor it would also be inappropriate for anyone affiliated with either CMU's Mechanical Engineering or Materials Science and Engineering Departments in the last five years to review this work. There are no other known conflicts of interest to disclose.

CRediT authorship contribution statement

Luke Scime: Conceptualization, Methodology, Software, Validation, Writing - original draft, Writing - review & editing, Project administration. **Derek Siddel:** Data curation. **Seth Baird:** Resources. **Vincent Paquit:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition.

Acknowledgements

This research was partly sponsored by the Transformational Challenge Reactor (TCR) program and supported by the US Department of Energy, Office of Nuclear Energy. The authors would like to thank Dr. Derek Rose and William Halsey of ORNL, and Dr. Christopher Kantzos of CMU for many fruitful discussions regarding machine learning. Collection of data from the ConceptLaser M2 machine was performed by Chase Joslin, Keith Carver, and Frederick List III of ORNL. Collection of data from the ExOne Innovents was performed by Dylan Richardson and Desarae Goldsby of ORNL. The EOS M290 data were collected by the authors with the support of the NextManufacturing Center at Carnegie Mellon University.

References

[1] E. Atzeni, A. Salmi, Economics of additive manufacturing for end-usable metal parts, Int. J. Adv. Manuf. Technol. 62 (2012) 1147–1155, https://doi.org/10.1007/s00170-011-3878-1.
[2] D.S. Thomas, S.W. Gilbert, Costs and Cost Effectiveness of Additive Manufacturing a Literature Review and Discussion, (2014), https://doi.org/10.6028/NIST.SP.1176.
[3] R.E. Laureijs, J. Bonnín Roca, S. Prabha Narra, C. Montgomery, J.L. Beuth, E.R.H. Fuchs, Metal additive manufacturing: cost competitive beyond low volumes, J. Manuf. Sci. Eng. 139 (2016) 1–9, https://doi.org/10.1115/1.4035420.
[4] D. Bourell, M. Leu, D. Rosen, Roadmap for additive manufacturing: identifying the future of freeform processing, Solid Free. Fabr. Proc. Austin, TX, 2009 (accessed May 23, 2017), http://wohlersassociates.com/roadmap2009A.pdf.
[5] B. Fisher, J. Mireles, S. Ridwan, R. Wicker, J. Beuth, Consequences of part temperature variability in Electron beam melting of Ti-6Al-4V, J. Mater. (2017), https://doi.org/10.1007/s11837-017-2597-y.
[6] R. Cunningham, S.P. Narra, C. Montgomery, J. Beuth, A.D. Rollett, Synchrotron-based X-Ray microtomography characterization of the effect of processing variables on porosity formation in laser power-bed additive manufacturing of Ti-6Al-4V, J. Mater. 69 (2016), https://doi.org/10.1007/s11837-016-2234-1.
[7] 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 Free. Fabr. Austin, TX, 2015, pp. 295–307, , https://doi.org/10.1017/CBO9781107415324.004.
[8] I. Gibson, D.W. Rosen, B. Stucker, Additive Manufacturing Technologies: Rapid Prototyping to Direct Digital Manufacturing, Springer, 2010 http://download.springer.com/static/pdf/764/bok%253A978-1-4419-1120-9.pdf?originUrl=http%3A%2F%2Flink.springer.com%2Fbook%2F10.1007%2F978-1-4419-1120-9&token2=exp=1496333413~acl=%2Fstatic%2Fpdf%2F764%2Fbok%25253A978-1-4419-1120-9.pdf%3ForiginUrl%3Dhttp%25 (accessed June 1, 2017).
[9] ExOne, Technology Overview, (n.d.). http://www.exone.com/Resources/Technology-Overview (accessed January 4, 2018).
[10] T. Craeghs, S. Clijsters, E. Yasa, J.-P. Kruth, Online quality control of selective laser melting, Solid Free. Fabr. Proc. (2011) 212–226 http://utwired.engr.utexas.edu/lff/symposium/proceedingsarchive/pubs/Manuscripts/2011/2011-17-Craeghs.pdf.
[11] ConceptLaser, Quality Management, (2018) (accessed February 8, 2018), https://www.concept-laser.de/en/products/quality-management.html.
[12] S.A. Khairallah, A.T. Anderson, A. Rubenchik, W.E. King, Laser powder-bed fusion additive manufacturing: physics of complex melt flow and formation mechanisms of pores, spatter, and denudation zones, Acta Mater. 108 (2016) 36–45, https://doi.org/10.1016/j.actamat.2016.02.014.
[13] J. Zur Jacobsmuhlen, S. Kleszczynski, G. Witt, D. Merhof, Detection of elevated regions in surface images from laser beam melting processes, Conf. IEEE Ind. Electron. Soc. (2016) 1270–1275, https://doi.org/10.1109/IECON.2015.7392275.
[14] A.J. Dunbar, E.R. Denlinger, J. Heigel, P. Michaleris, P. Guerrier, R. Martukanitz, T.W. Simpson, Development of experimental method for in situ distortion and temperature measurements during the laser powder bed fusion additive manufacturing process, Addit. Manuf. 12 (2016) 25–30, https://doi.org/10.1016/j.addma.2016.04.007.
[15] W.J. Sames, Additive Manufacturing of Inconel 718 Using Electron Beam Melting: Processing, Post-processing, & Mechanical Properties, Texas A&M University, 2015 file:///C:/Users/amostafa/Downloads/SAMES-DISSERTATION-2015 (1).pdf..
[16] M. Tang, P.C. Pistorius, J.L. Beuth, Prediction of lack-of-fusion porosity for powder bed fusion, Addit. Manuf. 14 (2017) 39–48, https://doi.org/10.1016/j.addma.2016.12.001.
[17] N. Boone, C. Zhu, C. Smith, I. Todd, J.R. Willmott, Thermal near infrared monitoring system for electron beam melting with emissivity tracking, Addit. Manuf. 22 (2018) 601–605, https://doi.org/10.1016/j.addma.2018.06.004.
[18] L. Scime, J. Beuth, Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm, Addit. Manuf. 19 (2018) 114–126, https://doi.org/10.1016/j.addma.2017.11.009.
[19] L. Scime, J. Beuth, A multi-scale convolutional neural network for autonomous anomaly detection and classification in a laser powder bed fusion additive manufacturing process, Addit. Manuf. 24 (2018) 273–286, https://doi.org/10.1016/j.addma.2018.09.034.
[20] M. Abdelrahman, E.W. Reutzel, A.R. Nassar, T.L. Starr, Flaw detection in powder bed fusion using optical imaging, Addit. Manuf. 15 (2017) 1–11, https://doi.org/

10.1016/j.addma.2017.02.001.
[21] B. Zhang, J. Ziegert, F. Farahi, A. Davies, In situ surface topography of laser powder bed fusion using fringe projection, Addit. Manuf. 12 (2016) 100–107, https://doi.org/10.1016/j.addma.2016.08.001.
[22] Z. Li, X. Liu, S. Wen, P. He, K. Zhong, Q. Wei, Y. Shi, S. Liu, In Situ 3D Monitoring of Geometric Signatures in the Powder-bed-fusion Additive Manufacturing Process Via Vision Sensing Methods, Sensors (Switzerland), 2018, p. 18, https://doi.org/10.3390/s18041180.
[23] L.T. Phuc, M. Seita, A high-resolution and large field-of-view scanner for in-line characterization of powder bed defects during additive manufacturing, Mater. Des. 164 (2018) 107562, , https://doi.org/10.1016/j.matdes.2018.107562.
[24] M. Aminzadeh, A Machine Vision System for In-Situ Quality Inspection in Metal Powder-Bed Additive Manufacturing, Georgia Institute of Technology, 2016.
[25] F. Caltanissetta, M. Grasso, S. Petrò, B.M. Colosimo, Characterization of in-situ measurements based on layerwise imaging in laser powder bed fusion, Addit. Manuf. 24 (2018) 183–199, https://doi.org/10.1016/j.addma.2018.09.017.
[26] A. Bin Anwar, Q.C. Pham, Study of the spatter distribution on the powder bed during selective laser melting, Addit. Manuf. 22 (2018) 86–97, https://doi.org/10.1016/j.addma.2018.04.036.
[27] M. Aminzadeh, T.R. Kurfess, Online quality inspection using Bayesian classification in powder-bed additive manufacturing from high-resolution visual camera images, J. Intell. Manuf. (2018) 1–19, https://doi.org/10.1007/s10845-018-1412-0.
[28] C. Gobert, E.W. Reutzel, J. Petrich, A.R. Nassar, S. Phoha, Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging, Addit. Manuf. 21 (2018) 517–528, https://doi.org/10.1016/j.addma.2018.04.005.
[29] F. Imani, A. Gaikwad, M. Montazeri, P. Rao, H. Yang, E. Reutzel, Layerwise in-process quality monitoring in laser powder bed fusion, ASME 2018 13th Int. Manuf. Sci. Eng. Conf. (2018) 1–10, https://doi.org/10.1115/MSEC2018-6477.
[30 M. Kirka, D. Rose, W. Halsey, A. Ziabari, V. Paquit, R. Dehoff, P. Brackman, Analysis of Data Streams for Qualification and Certification of Inconel 738 Airfoils Processed Through Electron Beam Melting, ASTM Int. (n. d) (2019).
[31] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit, 07-12-June, 2015, pp. 3431–3440, , https://doi.org/10.1109/CVPR.2015.7298965.
[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1–9, https://doi.org/10.1016/j.protcy.2014.09.007.
[33] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR, 2014, pp. 398–406 http://arxiv.org/abs/1409.1556.
[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, IEEE, 2014, pp. 186–191, https://doi.org/10.1089/pop.2014.0089.
[35] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, (2015), pp. 1–8, https://doi.org/10.1007/978-3-319-24574-4_28.
[36] H.G.R. Gouk, A.M. Blake, Fast sliding window classification with convolutional neural networks, Proc. 29th Int. Conf. Image Vis. Comput, New Zeal. - IVCNZ' 14, 2014, pp. 114–118, , https://doi.org/10.1145/2683405.2683429.
[37] W. Shen, M. Zhou, F. Yang, C.aiyun Yang, Jie Tian, Multi-scale convolutional neural networks for lung nodule classification, Int. Conf. Inf. Process. Med. Imaging (2015) 588–599, https://doi.org/10.1007/978-3-319-19992-4.
[38] G. Bertasius, J. Shi, L. Torresani, DeepEdge: a multi-scale bifurcated deep network for top-down contour detection, IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (2015) 4380–4389, https://doi.org/10.1109/CVPR.2015.7299067.
[39] A. Raj, D. Maturana, S. Scherer, Multi-Scale Convolutional Architecture for Semantic Segmentation, Pittsburgh, PA (2015).
[40] ASTM, Standard Terminology for Additive Manufacturing, (2015) (accessed May 3, 2017), https://compass.astm.org/download/ISOASTM52900.23551.pdf.
[41] ISO International, ISO-12233, 2014 Photography – Electronic Still Picture Imaging – Resolution and Spatial Frequency Responses, (2014) https://www.iso.org/standard/59419.html.
[42] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444, https://doi.org/10.1038/nature14539.
[43] S.W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, 1st ed., California Technical Publishing, 1997, http://www.dspguide.com/.
[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252, https://doi.org/10.1007/s11263-015-0816-y.
[45] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? Neural Inf. Process. Syst. Conf. (2014) 1–9 http://arxiv.org/abs/1411.1792.
[46] A. Karpathy, CS231n Convolutional Neural Networks for Visual Recognition, Standford Comput. Sci. (2017) (accessed February 5, 2018), http://cs231n.github.io/convolutional-networks/#overview.
[47] P. Joshi, What Is Local Response Normalization in Convolutional Neural Networks, Perpetual Enigm, (2016) https://prateekvjoshi.com/2016/04/05/what-is-local-response-normalization-in-convolutional-neural-networks/.
[48] A. Deshpande, A Beginner's Guide to Understanding Convolutional Neural Networks, (2016) https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/ (accessed February 7, 2018).
[49] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, 32nd Int. Conf. Mach. Learn. ICML 2015 (2015) 448–456.
[50] M.K. Smith, Overfitting, Univ. Texas Austin. (n.d.). http://www.ma.utexas.edu/users/mks/statmistakes/ovefitting.html (accessed December 30, 2016).
[51] A. Ng, J. Ngiam, C. Yu Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, S. Tandon, Optomization: Stochastic Gradient Descent, (n.d.). http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/ (accessed February 7, 2018).
[52] TensorFlow, Softmax Cross Entropy with Logits, (2018) (accessed January 29, 2019), https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits_v2.
[53] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, A. Rabinovich, Training Deep Neural Networks on Noisy Labels With Bootstrapping, ICLR, 2014, pp. 1–11, https://doi.org/10.2200/S00196ED1V01Y200906AIM006.
[54] D.P. Kingma, J. Ba Adam, A Method for Stochastic Optimization 1631 ICLR, 2014, pp. 58–62 http://arxiv.org/abs/1412.6980.
[55] R. Geirhos, C. Michaelis, P. Rubisch, ImageNet - Trained CNNS Are Biased Towards Texture ; Increasing Shape Bias Improves Accuracy and Robustness, ICLR, 2019, pp. 1–22.
[56] TensorFlow, Exponential Moving Average, (2018) (accessed January 29, 2019), https://www.tensorflow.org/api_docs/python/tf/train/ExponentialMovingAverage.
[57] J. Brownlee, What is the Difference Between Test and Validation Datasets, Mach. Learn. Mastery, 2017 (accessed February 8, 2018), https://machinelearningmastery.com/difference-test-validation-datasets/.
[58] R. Fisher, S. Perkins, A. Walker, E. Wolfart, Distance Transform, Univ. Edinburgh Sch. Informatics, 2003 (accessed May 10, 2019), https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm.
[59] L. Scime, Methods for the Expansion of Additive Manufacturing Process Space and the Development of In-Situ Process Monitoring Methodologies, Carnegie Mellon University, 2018.