Journal of Manufacturing Systems 64 (2022) 197–210

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Technical Paper

# Federated learning-based semantic segmentation for pixel-wise defect detection in additive manufacturing

Manan Mehta, Chenhui Shao [*,1]

*Department of Mechanical Science and Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, United States*

ARTICLE INFO

*Keywords:*
Quality
Defect detection
Federated learning
Semantic segmentation
Metal additive manufacturing
Data privacy

ABSTRACT

Semantic segmentation is a promising machine learning (ML) method for highly precise fine-scale defect detection and part qualification in additive manufacturing (AM). Most existing segmentation methods utilize convolutional neural network architectures that require large quantities of training data. However, obtaining sufficient data—both in quality and quantity—to train such models is expensive and time-consuming for individual AM practitioners, which severely limits the deployment of semantic segmentation in a data-scarce production environment. Similar data may be readily available with other AM practitioners that cannot be pooled together for conventional centralized learning (CL) due to its sensitive nature or conflicts of interest. This paper develops a federated learning (FL)-based method to simultaneously alleviate the constraints of data availability and data privacy. A U-Net architecture is created for semantic segmentation and is trained under the FL framework. The effectiveness of the developed FL-based semantic segmentation approach is demonstrated using case studies on layer-wise images from the laser powder bed fusion process. Results show that the proposed technique achieves a comparable defect detection performance with CL, which shares data among manufacturers/clients but does not preserve data privacy, and significantly outperforms individual learning, where each manufacturer trains a model using its own data. Additionally, the impact of data distribution across clients, incentives to participate in FL, and the learning dynamics of FL are discussed in detail. It is found that data diversity within and across clients improves FL performance, and FL does not involve a significantly higher training cost compared to CL. Lastly, transfer learning is shown to enhance FL generalizability, thus allowing more manufacturers with heterogeneous machines or technologies to benefit from participating in a data federation. Overall, this work puts forth FL as a promising paradigm for privacy-preserving collaborative ML in AM process control.

## 1. Introduction

In recent years, process monitoring and control have emerged as an important research area in additive manufacturing (AM) [1–3]. Contemporary research in high-precision defect detection has centered around computer vision (CV) techniques powered by machine learning (ML) [4]. In particular, semantic segmentation has proven to be a promising method for highly precise fine-scale defect localization using image data, enabling rapid cost-effective monitoring and part qualification. In semantic segmentation, each pixel in an image is labeled into one or more human-interpretable classes (e.g., powder, part, defect). Among ML-based CV tasks such as image classification (“there is a defect in the image") and localization (“there is a defect at approximately this location in the image"), semantic segmentation (“there is a defect at exactly these pixels in the image") is one of the more challenging problems.

Semantic segmentation using deep neural networks has been used in different contexts in AM for defect identification and quality control. A dynamic segmentation network is formulated in [5] for highly localized defect classification on layer-wise images across three printing technologies. A fast real-time segmentation network is developed for fused filament fabrication in [6] using the DeepLabv3 architecture. A three-dimensional convolutional neural network (CNN) is used for volumetric defect segmentation in X-ray computed tomography (XCT) images of AM samples [7]. Other applications include fabrication of medical constructs [8], measurement and monitoring of melt pool

* Corresponding author.
*E-mail addresses:* mananm2@illinois.edu (M. Mehta), chshao@illinois.edu (C. Shao).
[1] https://mechse.illinois.edu/people/profile/chshao.

https://doi.org/10.1016/j.jmsy.2022.06.010
Received 18 February 2022; Received in revised form 1 May 2022; Accepted 21 June 2022
Available online 1 July 2022

geometry [9], and surface quality improvement [10].

Most conventional semantic segmentation models are built around CNN architectures which are inherently data hungry and require large high-quality datasets for training [11]. However, data availability is a major bottleneck in AM and the amount of data obtainable for training—in quantity and quality—is often not sufficient [12–14]. AM machines can be tasked with arbitrary production volumes for creating distinct yet complex 3D designs. This makes collecting high-quality training data expensive, time-consuming, and challenging. Furthermore, labeling image data obtained from XCT or manufacturer-installed cameras is laborious and requires a human expert throughout the process. Large and open-source datasets containing millions of training examples such as ImageNet [15], Coco [16] and Pascal VOC [17] are readily available to ML practitioners; however, acquiring such data is cost-prohibitive in AM.

Another challenge is that of data privacy [12,18]. Individual manufacturers working on similar printers or printing technologies have their own supervised learning datasets; it is thus promising to integrate their data and learn a powerful collective model via centralized learning (CL). However, this requires data sharing among the participating manufacturers or uploading their data to a third-party server, which is not feasible due to conflicts of interest and the sensitive nature of the data. Even departments within the same organization can face resistance against data integration due to legal or administrative constraints [19]. Thus, useful data exists as isolated 'data islands' which cannot be leveraged for collaborative training.

To simultaneously alleviate the constraints of data availability and data privacy, this paper develops a novel federated learning (FL)-based semantic segmentation method for pixel-wise defect detection in AM. In FL, tens or potentially millions of participants collaboratively solve an ML problem under the coordination of a central entity [20–22]. The participants are called *clients*, the central entity is called a *central server*, and their collection is called the *data federation*. In the AM context, clients can be printers used for prototyping or production by different industries, printers of the same technology at different locations, or printers of different technologies. In training FL models, the sensitive raw data held by individual clients are not exchanged; only local client updates are securely transmitted to the server for aggregation. This allows clients participating in the data federation to build a powerful collaborative model while keeping their individual data private.

FL was originally introduced as *cross-device* FL with an emphasis on decentralized learning with millions of mobile devices [20]. However, recent studies have shown the usefulness of *cross-silo* FL, in which the clients consist of fewer organizations (e.g., manufacturers, hospitals, banks) with reliable computing resources [23]. Throughout this paper, FL refers to the cross-silo setting where the participants are AM practitioners interested in building highly accurate semantic segmentation models.

We demonstrate the effectiveness of FL-based semantic segmentation using case studies on layer-wise image data from laser powder bed fusion (L-PBF) builds. A U-Net [24] is constructed for the segmentation task and trained under the FL framework. It is shown that the proposed method (1) achieves comparable defect detection performance as CL while preserving data privacy, and (2) significantly outperforms individual learning (IL), where manufacturers train models on their own limited data. The manufacturers participating in FL are shown to achieve highly accurate defect detection performance with as few as one or two images available for training. In addition, important characteristics of FL including the effect of data diversity among manufacturers and incentives to participate in FL are studied. Lastly, a transfer learning perspective is discussed to enhance FL generalizability and account for data heterogeneity, which can enable practitioners using different AM technologies to benefit from FL. While data from the L-PBF process is used for method demonstration, it is expected that the developed FL-based U-Net is applicable to pixel-wise defect detection in other AM technologies.

It is important to emphasize that the limitations of data availability and data privacy discussed above are not specific to defect detection in AM. These limitations are ubiquitous in almost every stage of the AM lifecycle where ML algorithms have added significant value, including design for AM [25,26], process optimization [27,28], in-situ monitoring [29,30], and part quality control [31–33]. In fact, these limitations can be generalized to ML for smart manufacturing as a whole [34]. While this paper presents an FL-based semantic segmentation framework for AM, the case studies are designed to demonstrate the general facets of FL. We anticipate that the FL framework, which can be coupled with other types of ML algorithms for various tasks, has very broad application scenarios in manufacturing. Thus, results and conclusions from this work can be utilized by practitioners and researchers across the manufacturing community.

The remainder of this paper is organized as follows. Section 2 introduces the dataset and formulates the semantic segmentation problem in the FL context. Section 3 explains the federated training methodology and the network architecture used for semantic segmentation. Section 4 presents four case studies to demonstrate the effectiveness of FL-based segmentation followed by a discussion of the learning dynamics. Lastly, Section 6 concludes the paper and unfolds crucial areas for future research. The developed code is publicly available at https://github.com/mananm2/FLAM.

## 2. Problem formulation

### 2.1. Dataset description

The data used in this paper contains layer-wise grayscale images of 20 layers from a ConceptLaser M2 L-PBF machine printed using Stainless Steel 316L powder, collected and compiled at Oak Ridge National Laboratory [35]. The complete dataset consists of layer-wise images from three machines (technologies) viz. ConceptLaser M2 (L-PBF), Arcam Q10 (electron beam powder bed fusion), and ExOne Innovent (binder jet). Of these, only the ConceptLaser M2 data is chosen to demonstrate FL because it (1) contains diverse builds with a rich topology of manufactured parts and defects, and (2) has very few unlabeled pixels compared to the data from the other two machines. To demonstrate federated transfer learning, four cropped images from the ExOne Innovent machine are separately utilized in Case Study D.

Each of the 20 L-PBF layers contains two images—one immediately following powder fusion (post-fusion) and the other immediately following powder spreading (post-spreading)—captured using the manufacturer-installed 5 MP visible-light camera. Of the 20 L-PBF images provided, 19 images have a calibrated size of $2325 \times 2325$ pixels, and one image has a calibrated size of $2174 \times 2174$ pixels.

The dataset includes pixel-wise annotations (segmentation masks) for each image with integer entries for different classes. The classes are divided into three categories—powder (label 0), printed part (label 1), and various types of defects (labels 2–8). Each defect label is associated with a particular type of defect, for instance recoater hopping (label 2), recoater streaking (label 3), soot (label 8), and so on. Unlabeled pixels are annotated with label $-1$. Each annotation file can be visualized as a $2325 \times 2325$ (or $2174 \times 2174$) matrix with each entry signifying the class to which the corresponding image pixel belongs. The complete dataset is highly imbalanced, with $\sim 80$ % pixels being powder (label 0), $\sim 14$ % being part (label 1), and $\sim 6$ % being defects (labels 2–8).

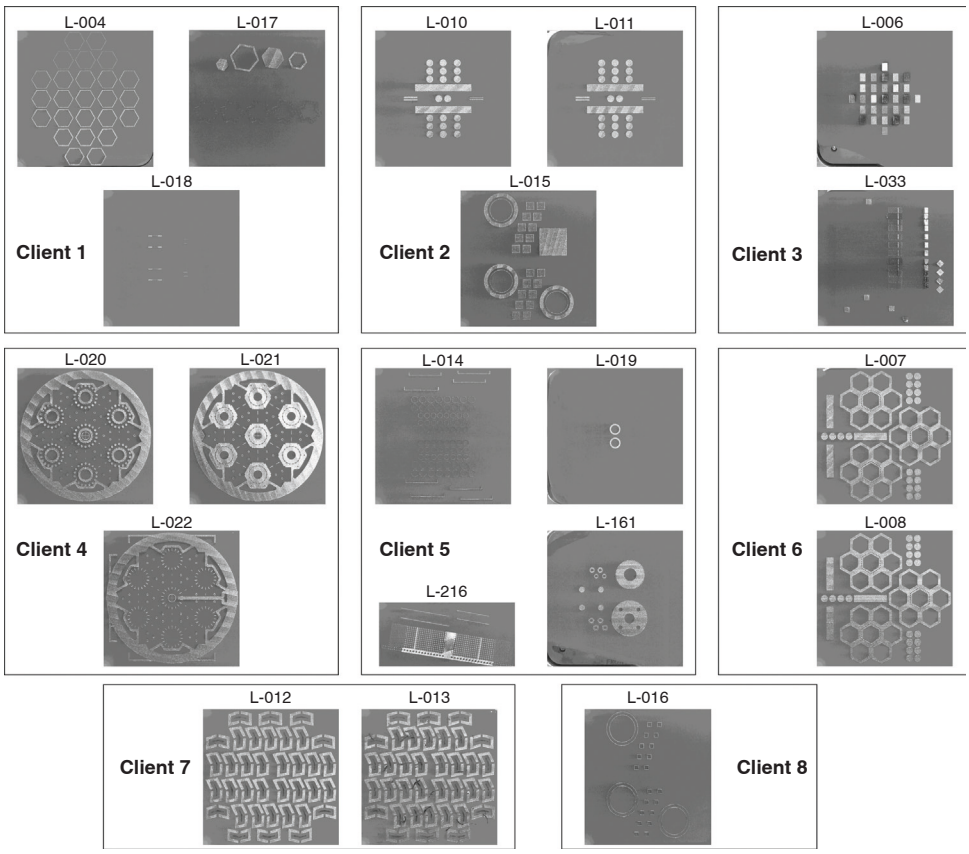


**Fig. 1.** Distribution of the 20 layer-wise L-PBF images into 8 clients for FL.

### 2.2. Semantic segmentation task and metric

Semantic segmentation aims to label each pixel in an image into one or more human-interpretable classes. This paper focuses on three pixel classes—powder (class 0), part (class 1), and defect (class 2). To this end, labels 2–8 in the annotations corresponding to different defect types are combined into a single 'defect' class. Thus, the output of the semantic segmentation task is a mask of the input image with each pixel labeled as powder, part, or defect.

Note that classifying each defect separately requires a highly complex, specific, and calibrated network structure. For instance, the same data was used in [5] to develop a Dynamic Segmentation CNN (DSCNN) with four parallel networks which transmit different types of information at different spatial scales. The performance of the DSCNN was enhanced using class-wise balancing weights, different loss functions, intelligent heuristics, and careful hyperparameter tuning. Implementing such techniques would shift the focus from FL to network optimization, making the results challenging to interpret solely from an FL perspective. Thus, the segmentation task in this paper is kept as generalizable and interpretable as possible.

Mean Intersection over Union (MeanIoU) is used as the metric for segmentation performance which quantifies the percent overlap between the true and predicted segmentation masks. For binary classification, Intersection over Union (IoU) or Jaccard index is defined as

$$\text{IoU} = \frac{\text{True Mask} \cap \text{Predicted Mask}}{\text{True Mask} \cup \text{Predicted Mask}}. \tag{1}$$

Stated simply, IoU measures the number of pixels common between the true and predicted masks divided by the total number of pixels present across both masks. IoU can be computed as

$$\text{IoU} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives} + \text{False Negatives}}. \tag{2}$$

For multi-class classification, MeanIoU is computed by averaging the IoU for each semantic class. For the case studies in this paper,

$$\text{MeanIoU} = \frac{\text{IoU}_{\text{powder}} + \text{IoU}_{\text{part}} + \text{IoU}_{\text{defect}}}{3}. \tag{3}$$

Canonically, IoU $> 0.5$ is considered a 'good' segmentation [36]. We use MeanIoU (and not classification accuracy) due to the heavy class-wise imbalance in the data. MeanIoU is a stricter metric for semantic segmentation compared to accuracy, where bad prediction performance in any class heavily penalizes the overall metric. A good MeanIoU in the case studies automatically translates to a high prediction accuracy.

### 2.3. Client-wise data distribution

The 20 layer-wise L-PBF images in the dataset are separated into eight clients for the purpose of demonstrating FL, as shown in Fig. 1. To help track the client-wise data, images are labeled as shortened versions of their file names in the dataset (e.g., image 0000004 in the dataset is labeled L-004). The data is separated based on the following precepts: .

1. Images from the same build or similar builds are pooled together into one client, e.g., L-020, L-021 and L-022 in Client 4.

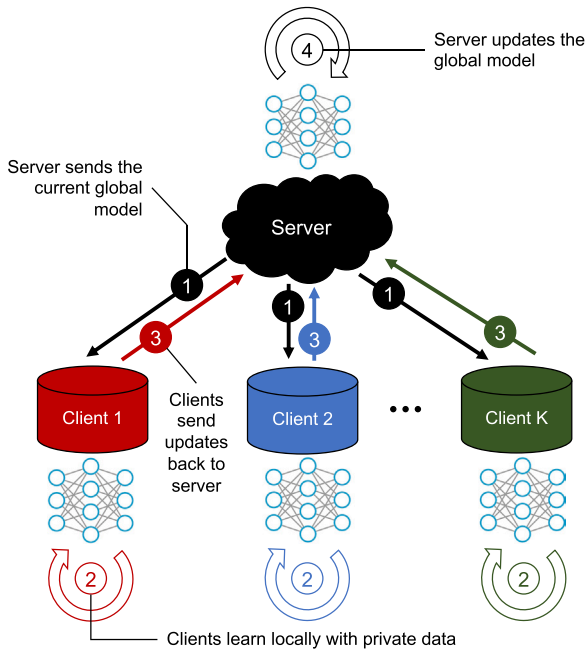


**Fig. 2.** A schematic for FL and its training procedure.

2. Images with similar underlying defects are pooled together into one client. For example, in Client 3, L-006 and L-033 both have 'soot' as the dominant defect type (over 80 % of the defect pixels in both layers are soot).
3. In the case when neither of the above is possible, images with parts having similar shapes (e.g., hexagons) are pooled together.

Forming clients based on the above precepts leads to a highly non-IID and unbalanced data distribution. Data within each client is highly similar; however, no client is representative of the complete dataset. This type of data distribution can be expected in practice, because individual manufacturers with a particular type of printer are likely to produce parts with similar characteristic features and underlying defects [37].

In the above data distribution, an individual client can only learn features very specific to their own data viz. a particular underlying defect or a peculiar part shape. Thus, a network trained on any one client is expected to be non-generalizable. Moreover, the amount of data from each client is dissimilar (e.g., Client 5 has four images while Client 8 has only one). This pathological non-IID and unbalanced data distribution helps in gauging the robustness of FL in a realistic and worst-case scenario.

## 3. Methodology

### 3.1. Federated learning

FL is an emerging ML paradigm where several participants collaboratively train a model under the orchestration of a central entity, while keeping the training data decentralized [20]. Conventionally, a global model is constructed using CL, where raw data from different clients is pooled together in a central database for model training. FL enables clients to obtain a globally optimized model without uploading data to a central database or sharing proprietary data with other clients, thus maintaining data privacy without compromising model performance.

The steps involved in training an FL model are shown in Fig. 2. First, the central server initializes a global model and shares it across all clients participating in the data federation. Each client downloads this model and trains it based on its local data for a predetermined number of cycles (local epochs). The parameter updates of the local models (weights of the neural network) are subsequently uploaded to the central server where they are aggregated and incorporated into the global model. This cycle (server round) is repeated until a convergence criterion of the global model is satisfied. Thus, information is transferred indirectly between clients through model updates while all learning happens locally, which effectively guarantees data privacy.

Two core challenges make the FL setting distinct and more complicated from traditional CL in the AM context:

1. Non-IID Data Distribution: A common assumption in ML is that the complete data is generated independently from the same underlying statistical process or distribution. This assumption may not hold in the FL setting, i.e., data generated locally at any manufacturer's site may not be representative of the population distribution.
2. Unbalanced Client Data: The amount of data generated at each manufacturer's site may be different; thus, local training may take place on varying amounts of client data.

As explained in Section 2.3, the above statistical heterogeneity is simulated in the L-PBF data to demonstrate the robustness of FL for different client distributions.

The Federated Averaging (FedAvg) algorithm is at the core of learning a global ML model from local client data. FedAvg can be applied to any finite-sum minimization of the form

$$\min_{w\in\mathbb{R}^d} f(w), \text{ where } f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w). \tag{4}$$

For neural networks, the non-convex minimization objective is the loss on a training example predicted with parameters $w$, i.e., $f_i(w) = \ell(x_i, y_i; w)$.

In the FL setting, the data is partitioned over $K$ clients with $n_k$ data points in the $k$th client. Let $\mathscr{P}_k$ be the index set of data points in client $k$ and $n = \sum_{k=1}^{K} n_k$ be the total number of training points. The minimization in Eq. (4) can be rewritten for the FL setting as

$$\min_{w\in\mathbb{R}^d} \underbrace{\sum_{k=1}^{K} \frac{n_k}{n}\mathscr{L}_k(w)}_{\text{weighted average of K client losses}}, \text{ where } \underbrace{\mathscr{L}_k(w) = \frac{1}{n_k}\sum_{i\in\mathscr{P}_k} \ell(x_i, y_i; w)}_{\text{loss for the } k^{th} \text{ client}}. \tag{5}$$

**Algorithm 1** Federated Averaging (`FedAvg`)

**Require**: Distributed data $\{\mathcal{P}_k\}_{k=1}^{K}$ across $K$ clients, number of local epochs $E$, local mini-batch size $B$, local learning rate $\eta$, number of server rounds $S$

**Server side:**

initialize the global model with parameters $w_0$
**for** each server round $t = 1, 2, \ldots, S$ **do**
  send current global model $w_{t-1}$ to all clients
  **for** each client $k$ **in parallel do**
    $w_t^k \leftarrow \text{ClientUpdate}(k, w_{t-1})$
    receive updates $w_t^k$ from client $k$
  compute weighted average and update global model $w_t \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_t^k$

**Client side:** Run $\text{ClientUpdate}(k, w)$ on client $k$

initialize local model with $w$
**for** each local epoch $1, 2, \ldots, E$ **do**
  **for** each mini-batch b of size B in $\mathcal{P}_k$ **do**
    $w \leftarrow w - \eta \nabla_w \ell(b)$

**Algorithm 1**. Federated Averaging (FedAvg).

Algorithm 1 shows the pseudo-code for FedAvg used to solve the distributed minimization in Eq. (5). FedAvg performs a weighted average of the gradients obtained from individual clients to update the global model in each server round. In general, naïve averaging in the parameter space $w \in \mathbb{R}^d$ can produce an arbitrarily bad model [20,38], except when the models share the same random initialization. This holds implicitly in every server round in FedAvg.

Three additional hyperparameters—the local learning rate ($\eta$), the number of local epochs ($E$), and the local mini-batch size ($B$)—are required for FedAvg, which are tuned similar to other network hyperparameters to optimize the network performance. In cross-device FL where FedAvg is applied to millions of mobile devices, training on all devices in every server round is inefficient and expensive. Moreover, not all devices are always available for local training. Thus, another hyperparameter called client fraction ($C$) is commonly used, which is the proportion of total clients participating in each server round. However, reliability is not an issue in cross-silo FL where the clients are manufacturing organizations. Thus, an underlying assumption is that the participating manufacturers are always available to communicate with the server and for local model training ($C = 1.0$).

It should be noted that there are other algorithms for aggregating updates from clients and updating the global model. For example,

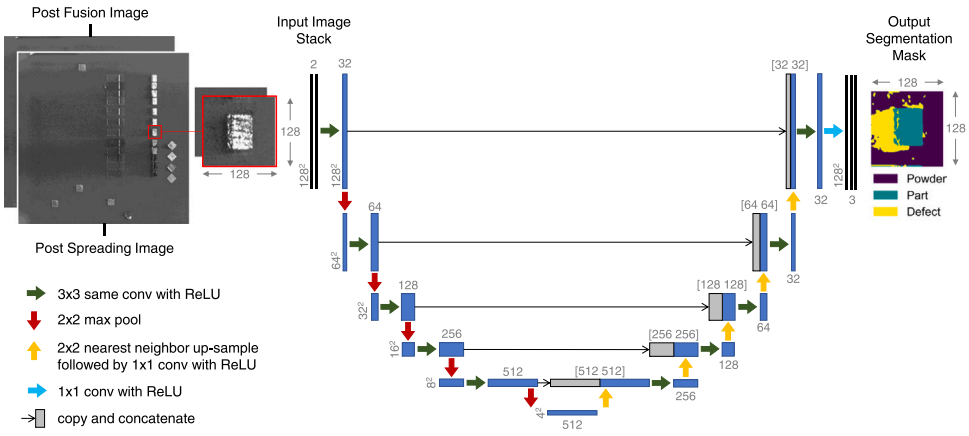


**Fig. 3.** Schematic for the U-Net architecture used for semantic segmentation. The inputs to the U-Net are tiles of size 128 × 128 pixels from the input images. The blue boxes in the U-Net correspond to a multi-channel feature map with the number of channels denoted at the top/bottom and the image size denoted to the left. Dimensions of the feature maps are representative and are not drawn to scale.

**Table 1**
Hyperparameters used for semantic segmentation using different training methodologies.

| Hyperparameter | CL and IL | FL |
|---|---|---|
| Learning Rate ($\eta$) | 8e-05 | 8e-05 |
| Mini-Batch Size ($B$) | 32 | 32 |
| Local Epochs ($E$) | – | 10 |
| Server Rounds ($S$) | – | 30 |
| Total Epochs ($N$) | 100 | 300 |

**Table 2**
MeanIoU ($IoU_{powder}/IoU_{part}/IoU_{defect}$) comparison for CL, FL, and IL on never-before-seen data.

| Test set | Layer ID | CL | FL | IL (average) |
|---|---|---|---|---|
| Client 6 | L-007 | 0.803 (0.964/ 0.971/0.473) | 0.775 (0.962/ 0.972/0.391) | 0.666 (0.937/ 0.913/0.147) |
| | L-008 | 0.845 (0.972/ 0.960/0.602) | 0.837 (0.972/ 0.953/0.587) | 0.698 (0.945/ 0.881/0.268) |
| Client 7 | L-012 | 0.859 (0.954/ 0.972/0.652) | 0.846 (0.948/ 0.966/0.623) | 0.727 (0.883/ 0.879/0.419) |
| | L-013 | 0.777 (0.896/ 0.882/0.552) | 0.777 (0.885/ 0.896/0.548) | 0.684 (0.837/ 0.820/0.395) |
| Client 8 | L-016 | 0.735 (0.978/ 0.761/0.465) | 0.782 (0.983/ 0.788/0.574) | 0.655 (0.956/ 0.672/0.337) |

FedProx adds a proximal term penalizing a client drifting away from the global model. Another alternative is federated stochastic gradient descent (FedSGD), in which the gradients are sent after every epoch. FedSGD has very low communication efficiency because there is no real gain in averaging gradients after each training iteration. FedAvg vastly outperforms FedSGD as shown in [20]. Additionally, it has been shown that FedAvg, despite being simplistic, outperforms all other federated algorithms [23]. As such, we select FedAvg over other algorithms in this research.

Unlike CL, setting aside a validation set is not possible in FL as the client data is not directly visible to the server or to other clients. This raises a natural concern for when to stop model training. In this work, the individual client losses are monitored at the end of each server round and the training is terminated when no client produces a significantly better model in a subsequent server round.

### 3.2. Network architecture

We develop a U-Net [24] architecture for pixel-wise defect detection, which is shown in Fig. 3. U-Net is an extension of fully convolutional networks [39] and is among the state-of-the-art networks currently used for semantic segmentation.

Each layer-wise image is first preprocessed by heuristically labeling or removing unlabeled pixels, followed by normalizing pixel intensities to the [0,1] range for stability. The details of these preprocessing steps are described in the Supplementary information (SI). The preprocessed images divided into tiles of 128 × 128 pixels for input to the U-Net. Such tiling of a large high-resolution image is typical in convolutional networks and helps to reduce the memory load of a single learning pass. A tile size of 128 is chosen to ensure that important contextual information is captured at the regional scale [5]. The post-fusion and post-spreading images are simultaneously tiled and stacked. Each stack is then passed as a two-channel input to the U-Net. This is shown with a representative tile stack in Fig. 3.

The first half of the U-Net is a contracting path that follows a typical CNN architecture, i.e., a repeated application of 3 × 3 same convolution followed by downsampling using 2 × 2 max pooling with stride 2. In each convolution operation, the number of filters is doubled and with each pooling operation, the size of the image is halved. This path of the U-Net extracts deep morphological features from the input image channels.

The second half of the U-Net is an expansive path that performs a series of upsampling operations followed by concatenation with the previously learned features. Typically in U-Nets, the upsampling operation is done using transpose convolution or up-convolution [24]. Contrarily in the above U-Net, a partially learned upsampling is performed using 2 × 2 nearest-neighbor upsampling followed by 1 × 1 convolution [5]. This operation requires fewer parameters than transpose convolution and is empirically found to perform better for the task at hand. The 3 × 3 convolutions in the expansive path map the preserved deeper features to their spatial locations obtained from concatenation.

The output of the series of upsampling operations is a 32-component feature vector at each pixel, which is mapped (collapsed) to the 3 output classes using a 1 × 1 convolution. Finally, a softmax activation is applied to this 3-element vector at each pixel, which rescales the class-wise responses (of arbitrary magnitude) to pseudo probabilities in (0, 1]. Mathematically, softmax transforms the 3-element vector $\mathbf{a} = [a_{\text{powder}}, a_{\text{part}}, a_{\text{defect}}] = [a_0, a_1, a_2]$ at a given pixel to a 3-element vector $\mathbf{p}$ where $p_j = \exp(a_j)/\left(\sum_{i=0}^{2}\exp(a_i)\right)$. A pixel-wise cross-entropy loss computes the 'degree of disagreement' between the network output $\mathbf{p}$ and the (one-hot encoded) true value $\mathbf{t}$ across all pixels. In the backpropagation step, this loss is minimized using the Adam optimizer [40].

To sum up, the contracting path of the U-Net learns *what* is present in the input images, and the expansive path learns *where* it is present. When put together, the output is a 128 × 128 image (i.e., an image at the same resolution as the input image) with each pixel classified as powder, part, or defect. The complete network contains 8,291,907 learnable parameters, which is small compared to many modern deep networks [41–44].

## 4. Case studies

In this section, four case studies are presented to demonstrate the effectiveness of the developed FL-based semantic segmentation method and understand its characteristics including (1) performance comparison with CL and IL, (2) the impact of data distribution across different clients, (3) incentives to participate in the data federation, and (4) the limitations of FL. Lastly, a detailed discussion of the training dynamics is presented.

The hyperparameter values used for centralized, federated, and individual learning are listed in Table 1.

In CL, data from all clients is pooled together for training. Thus, to train the CL model, the tiles from all training clients are batched together, shuffled, and passed to the U-Net. A random 10 % subset of training tiles is kept as a validation set, on which the performance of the network is monitored. The CL network requires 80–110 epochs for training (depending on the participating clients), after which the loss on the validation set does not decrease. For uniformity, all test set results are reported after $N = 100$ training epochs using a batch size $B = 32$.

In IL, the training methodology is the same as CL, except that a client only trains on its own data and does not utilize the data from other clients in any form.

In FL, each client trains the U-Net locally for $E = 10$ epochs with batch size $B = 32$, after which weighted model averaging is performed at the server (see Algorithm 1). This completes one server round. For the case studies on the given data, it is found that training the model for ~ 30 server rounds produces satisfactory model convergence, such that no client produces a significantly better model in subsequent iterations. For uniformity, all test set results are reported after $S = 30$ server rounds.

### 4.1. Case study A: performance comparison between CL, FL, and IL

In this case study, the performance of FL is evaluated on completely new layers/builds, i.e., never-before-seen data, and is compared to two

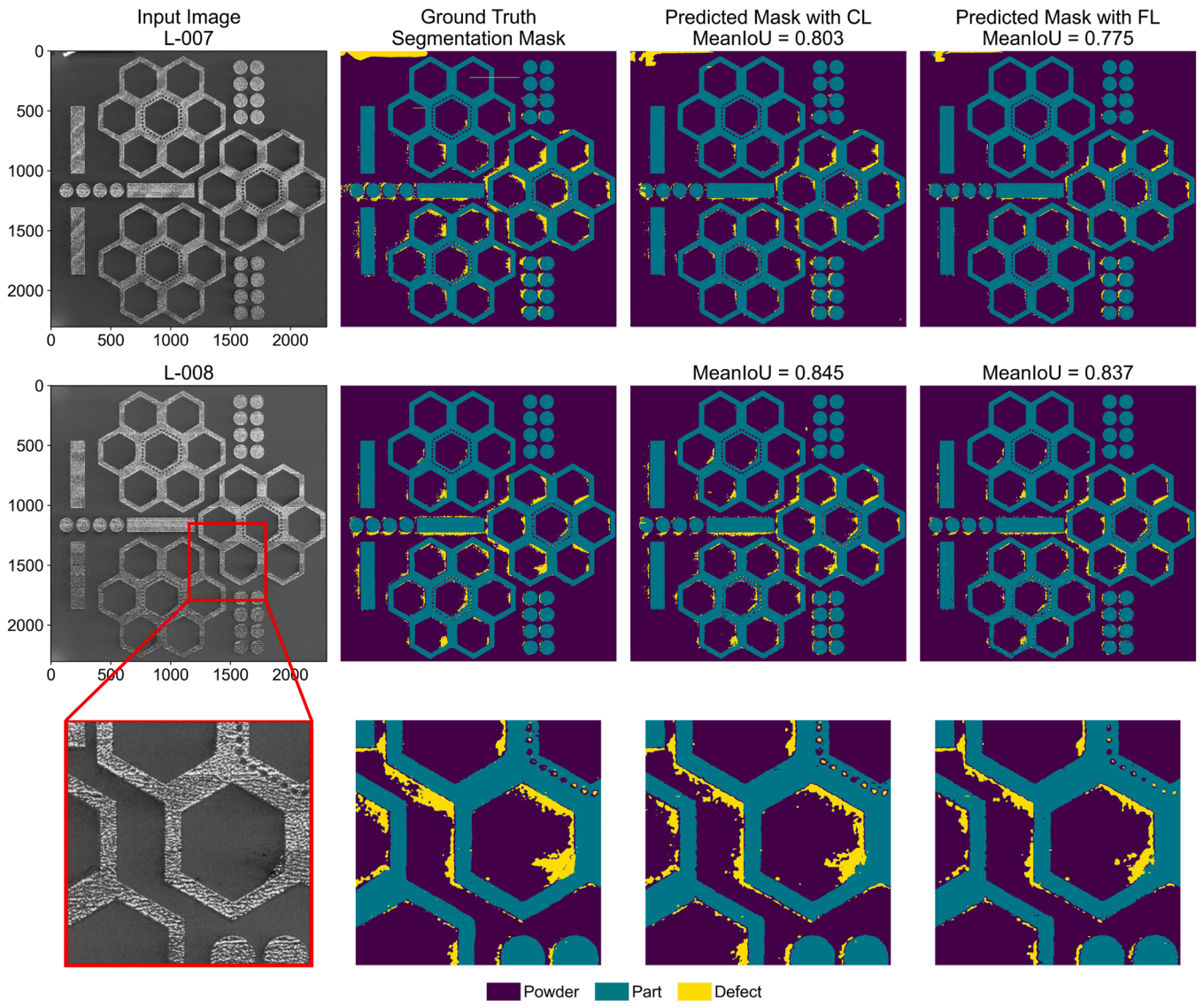


**Fig. 4.** A visual comparison of the input image, the ground truth segmentation mask and the predicted masks with CL and FL for Client 6. A region of size 5 tiles × 5 tiles (640 pixels × 640 pixels) is zoomed-in for a detailed view.

baselines—when images from all clients are pooled together (CL) and when clients train their models individually (IL).

Note that other traditional methods in defect detection are highly specific to the data being used (e.g., type of printer, number of images) and the task being performed (e.g., melt pool recognition, detection of a particular type of defect, binary classification of part quality). Moreover, investigation on pixel-wise defect detection in AM is still in its nascent stage and classifying each pixel is a significantly harder problem than other defect detection tasks. This research probes the merits of FL for pixel-wise defect detection to deal with data scarcity and data privacy. Thus, there are no other conventional methods we can directly compare with FL and only CL and IL with the same U-Net architecture are used as baselines for comparison.

To compare FL and CL, both models are trained using seven clients and their performance is tested on the layers in the remaining client, which acts as a holdout test set. To further ensure generalizability, three different holdout sets including Clients 6–8 are used. When Client 8 is used as a test set, Clients 1–7 all participate in training; the same procedure applies when Clients 6 and 7 are used as test sets. Note that when "Client 6 acts as a holdout test set", it loses the notion of being a client in the federation. Instead, the images in Client 6 (L-007 and L-008) are simply used to test and compare the performance of different algorithms. The MeanIoU for the segmentation along with IoUs for each class are reported in Table 2.

For IL, each client is allowed to individually learn the U-Net and the performance on the three test sets is noted. In Table 2, the average performance of all individually learned clients is reported. The complete client-wise IL results are provided in the SI.

From Table 2, it is clear that the MeanIoUs for FL are closely matched to the values for CL, whereas the average MeanIoU for IL is significantly worse than FL. More importantly, the defect detection performance ($IoU_{defect}$) for CL and FL are comparable, whereas the $IoU_{defect}$ for IL is significantly worse.

The above results draw three important conclusions. First, individual clients do not have sufficient data to learn accurate models solely using their own data. This can be seen in the significantly worse defect detection performance for IL compared to CL or FL. It is thus beneficial for the clients to build a collaborative model. Second, although each client has somewhat specific attributes in their private data, the FL model inherits those attributes without directly observing any client's data. Third, there is practically no reduction in the defect detection performance with FL compared to CL, i.e., the participating

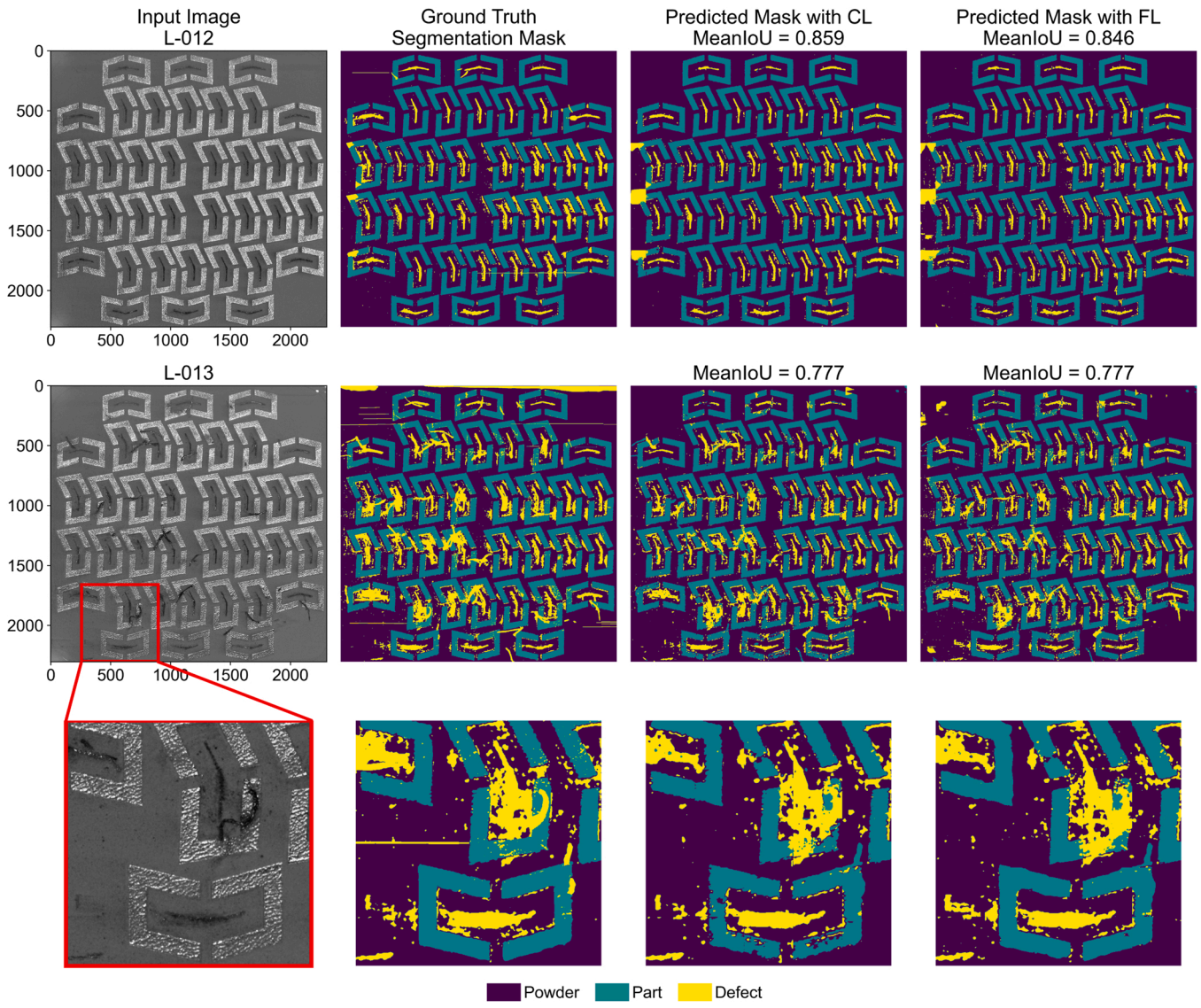


**Fig. 5.** A visual comparison of the input image, the ground truth segmentation mask and the predicted masks with CL and FL for Client 7. A region of size 5 tiles × 5 tiles (640 pixels × 640 pixels) is zoomed-in for a detailed view.

manufacturers achieve the same model performance without compromising privacy.

For a visual comparison between the defect detection performance of CL and FL, the original test set images, ground truth segmentation masks, and the predicted masks for Clients 6–8 are provided in Figs. 4, 5, and 6, respectively. The figures show near-perfect classification of powder and part pixels for both CL and FL. While individual defect pixels are significantly harder to classify, the regions where defects are present are correctly and identically predicted for both CL and FL. A 5 tiles × 5 tiles zoomed-in view of the larger image is provided in each figure for a detailed comparison.

### 4.2. Case study B: impact of data diversity on FL performance

The scope of the client distribution in Fig. 1 is restricted by the limited training data available. In a real-world AM data federation, it is possible that participating clients have a lot more local data available. In such cases, there is a good chance that characteristics from new builds would have already appeared in training. For example, if some parts in a test build are hexagons, similar-but-not-identical hexagons have already appeared in training through some client's data. In this case study, the effect of observing similar-but-not-identical data during training is examined.

To simulate a scenario with greater data diversity, L-008 from Client 6 and L-013 from Client 7 are swapped. This creates two new clients as shown in Fig. 7, of which L-007 and L-013 participate in training, and L-008 and L-012 are used as a holdout test set. The remaining clients (1–5, 8) are unchanged. Now that images similar to the test set are observed in a client during training, the aim is to compare this FL model to the one in Case Study A. The results are shown in Table 3.

The MeanIoU for both L-008 and L-012 improves when similar images are present during training. More importantly, the improvement in MeanIoU in both layers is solely due to an improvement in defect prediction. As seen from Table 3, $IoU_{defect}$ improves from 0.587 to 0.632 for L-008 and from 0.623 to 0.652 for L-012, while $IoU_{powder}$ and $IoU_{part}$ remain consistent. This is because the powder and part classes are easier to learn and predict compared to the defect class due to which their IoUs remain unaffected on observing similar layers. Defects, however, might be correlated to the exact part geometry and the type of build. For example, the network may learn that the inner vertical edges of hexagons in L-007 almost always have defects in the direction of recoater travel. On observing a similar geometry in L-008 during prediction,

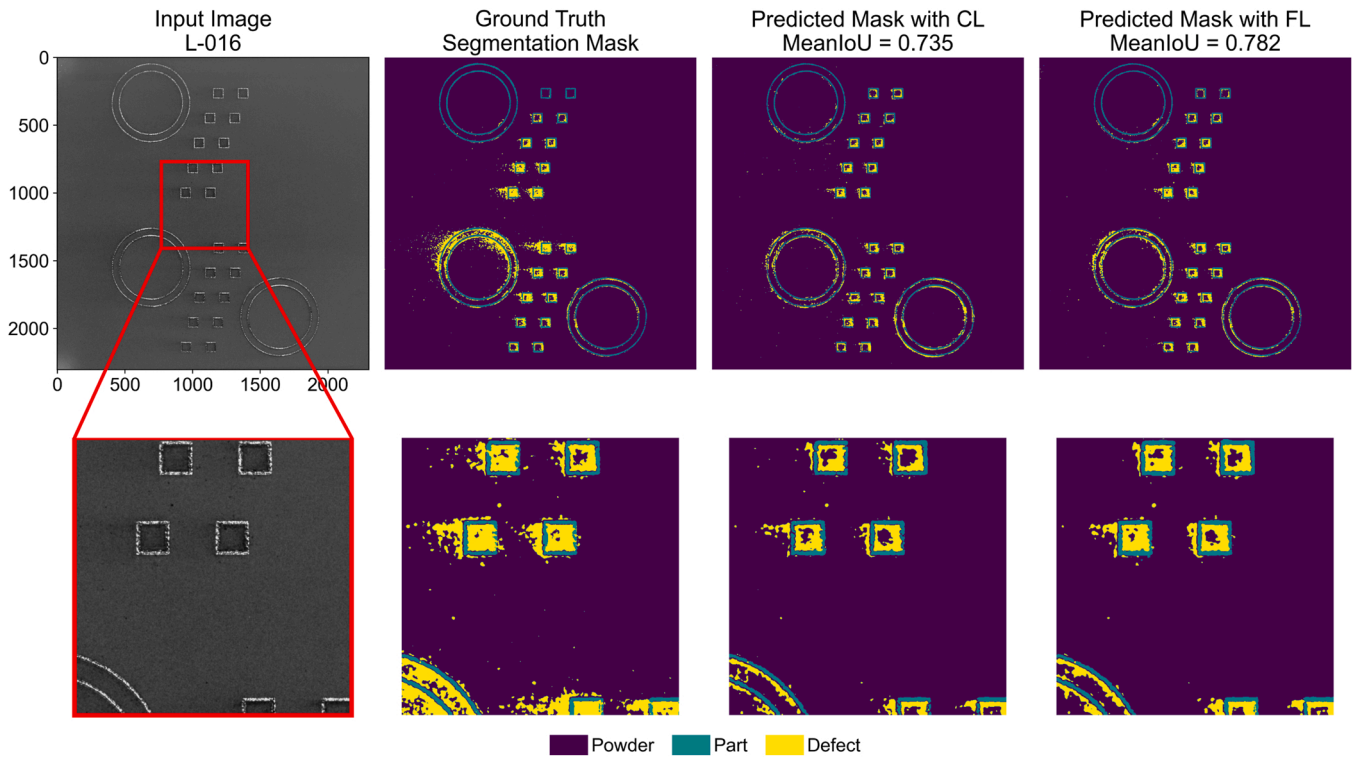


**Fig. 6.** A visual comparison of the input image, the ground truth segmentation mask and the predicted masks with CL and FL for Client 8. A region of size 5 tiles × 5 tiles (640 pixels × 640 pixels) is zoomed-in for a detailed view.

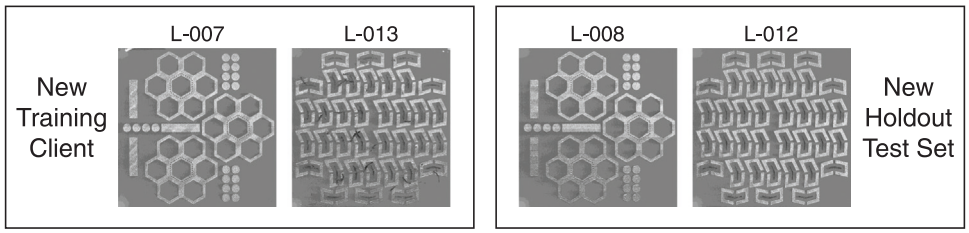


**Fig. 7.** Redistribution of the original Clients 6 and 7 for Case Study B. The new client (L-007 and L-013) participates in FL training whereas L-008 and L-012 are used as the holdout test set.

pixels which were previously labeled powder are now correctly classified as defects. This is appealing from a practitioner's perspective as manufacturers are always more interested in defect detection.

The above results imply that increasing data diversity in the client distributions enhances the quality of the learned FL model, particularly for the defect class which is harder to learn and predict. This intuition is true for CL [45], and for ML in general. In the FL context, data diversity can be increased within each client, or by adding new clients to the data federation. The latter is explored in Case Study C.

### 4.3. Case study C: incentive to participate in FL

For FL to be practically applicable, manufacturers participating in the federation must have sufficient incentive. A data federation benefits from client diversity, and with FL, individual clients do not have to worry about sharing proprietary data to improve their own models. In this case study, the effect of incentives is examined from the perspectives of both an existing data federation and an added client. To this end, an FL model is learned using Clients 1–4, and its performance is evaluated on Client 8 as the holdout test set. Next, Clients 5–7 are progressively added to the federation and the performance change is tracked.

**Table 3**
MeanIoU ($IoU_{powder}$/$IoU_{part}$/$IoU_{defect}$) comparison for federated learning when similar layer images are absent vs. present during training.

| Layer ID | Similar image absent | Similar image present |
|---|---|---|
| L-008 | 0.837 (0.972/0.953/0.587) | **0.852** (0.972/0.952/**0.632**) |
| L-012 | 0.846 (0.948/0.966/0.623) | **0.857** (0.951/0.968/**0.652**) |

#### 4.3.1. Incentive for the existing federation to accept new clients

The trends of the holdout test set loss and MeanIoU are plotted in Fig. 8. The loss on the test set decreases from 0.076 to 0.045, and the prediction performance increases from a MeanIoU = 0.726 to MeanIoU = 0.807 as new clients are added. Thus, the existing data federation benefits from adding new clients. This is because the added clients augment the data diversity of the federation and lead to an overall positive information transfer.

Table 4 shows the IoUs for each class after progressively adding new clients. As discussed in Case Study A, the powder and part classes are relatively easier to predict than the defect class. Thus, the increase in MeanIoU on adding new clients is almost entirely due to an improvement in defect prediction. While the existing federation had a defect detection IoU of 0.413, adding new clients improves it to 0.686. As explained earlier, this improvement in defect detection performance is highly desirable in practice. It is worth noting that the results in Fig. 8

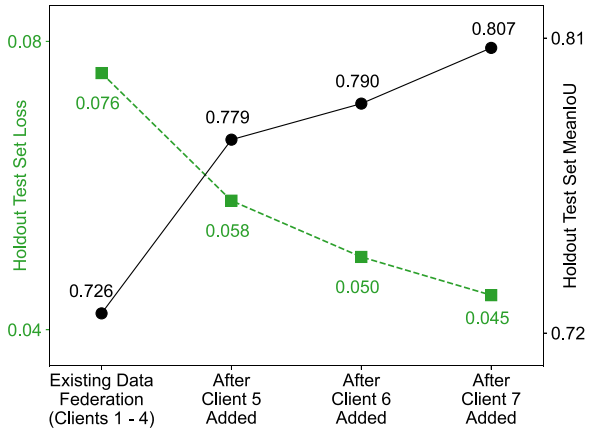


**Fig. 8.** Loss and MeanIoU for the holdout test set (Client 8) when Clients 5–7 are progressively added to the federation.

**Table 4**
Individual and Mean IoUs after progressively adding new clients.

| | $IoU_{powder}$ | $IoU_{part}$ | $IoU_{defect}$ | MeanIoU |
|---|---|---|---|---|
| Existing federation (clients 1–4) | 0.979 | 0.786 | **0.413** | 0.726 |
| After client 5 added | 0.983 | 0.754 | **0.599** | 0.779 |
| After client 6 added | 0.984 | 0.744 | **0.642** | 0.790 |
| After client 7 added | 0.985 | 0.750 | **0.686** | 0.807 |

and Table 4 are contingent upon the type of holdout set used. Thus, the existing data federation should use a representative validation set/build to gauge the impact of adding a new client.

#### 4.3.2. *Incentive for a new client to enter an existing federation*

The primary incentive for a client to enter a federation is the remarkable performance improvement compared to IL (see Table 2). Besides a more accurate model, individual clients can also have significantly smaller training time on entering an existing federation. This is demonstrated using a representative client (Client 6) in Fig. 9, where the training loss is plotted for FL and IL. When Client 6 enters the federation, a transfer learning [46] effect is observed where its local network is initialized with the weights of the well-trained existing federation rather than randomly. Thus, the loss begins with a low value and reduces very rapidly. This network attains a loss of 0.04 in only 10 local epochs compared to ~ 90 epochs when learned individually. Training deep networks is time and resource expensive and training large networks from scratch may take several hours or even days. For such large networks, faster training can be an incentive for clients to enter an existing federation.

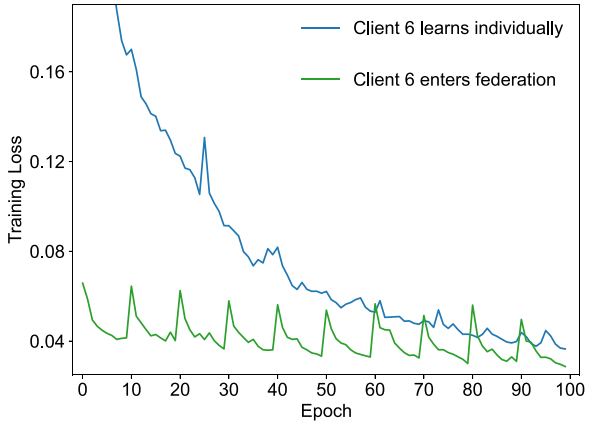


**Fig. 9.** Training loss comparison when Client 6 learns individually vs. enters the existing data federation.

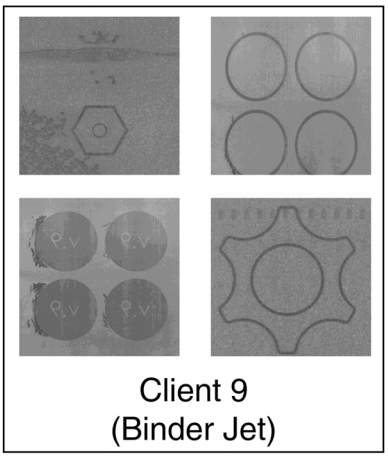


**Fig. 10.** New client added to the data federation consisting of four 1920 × 1920 images from the ExOne Innovent binder jetting machine.

**Table 5**
MeanIoU ($IoU_{powder}$/$IoU_{part}$/$IoU_{defect}$) comparison for FL model trained with and without the images from the binder jetting Client (Client 9).

| Test set | Layer ID | Client 9 not included in federation | Client 9 included in federation |
|---|---|---|---|
| Client 8 | L-016 | 0.782 (0.983/0.788/0.574) | 0.716 (0.963/0.711/0.474) |

### 4.4. *Case study D: negative transfer and improving generalizability*

In Case Studies A–C, the clients consist of layer-wise images from the same printing technology (L-PBF) and positive information transfer is demonstrated. However, is more data always good? It is important to analyze the extent to which FL can handle dissimilar clients. In this case study, FL performance is evaluated when a client from a different printing technology (binder jetting) participates in the data federation.

Four layer-wise images from an ExOne Innovent binder jetting machine (Fig. 10) are used to form a new client–Client 9. Each image in Client 9 is a 1920 pixels × 1920 pixels cropped version of a full image from the dataset. The images are preprocessed and tiled in the same manner as the L-PBF images. Client 9 contributes a total of 900 tiles to the federation, which is comparable to the existing clients. The FL model is now trained with Clients 1 through 7 and Client 9; the performance is evaluated on Client 8. The MeanIoUs with and without the binder jetting client are compared in Table 5.

The test set MeanIoU decreases when the binder jetting images are included in training, which indicates that the data federation becomes worse when Client 9 is added. This is an example of negative information transfer due to the inclusion of a dissimilar client, i.e., Client 9.

This negative transfer can be attributed to the fundamentally distinct mechanisms generating the parts and defects in L-PBF and binder jetting. The quality, type, and field of view of the cameras capturing these images are also different for both machines. This level of dissimilarity is not handled well by FL, i.e., more data is not necessarily beneficial. A closer look at pixel intensities reveals the numerical reason underlying the negative transfer. The pixel intensity values corresponding to the part labels in binder jetting are approximately equal to the values for the soot defect in L-PBF. Let this pixel intensity be '$p$'. Since Client 9 has only ~ 13 % weight in the federation, the global model (mapping $p \rightarrow$ defect) prevails over its local model (mapping $p \rightarrow$ part) during federated training. Fig. 11 shows the prediction on an image from Client 9 after the FL model is trained. The entire part in the image is classified as a defect leading to a poor MeanIoU for Client 9 on

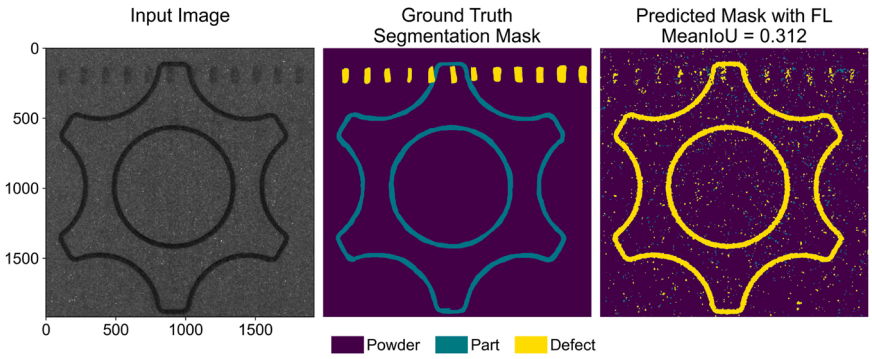


**Fig. 11.** An example of negative transfer with FL for a binder jetting image. The pixel intensity of the part in the input image is similar to an L-PBF defect, due to which the complete part is characterized as a defect.

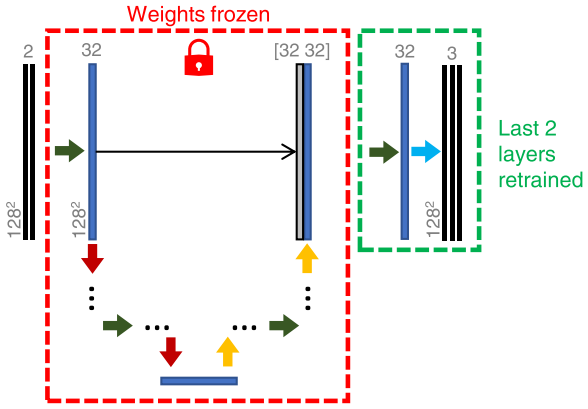


**Fig. 12.** A schematic for the federated transfer learning strategy. The weights from the pre-trained data federation are frozen and the last 2 layers are retrained for the binder jetting images. The complete U-Net is shown in Fig. 3.

its own training set.

Note that in Fig. 11, the part geometry is correctly identified, i.e., the geometric feature extraction performed by the global network works fine. Only the label attribution is incorrect (part labeled as defect) because the local Client 9 model is overpowered by the global model in FedAvg. Instead of including Client 9 in the data federation, an additional experiment is performed where weights for Client 9 are initialized with the weights of the global network. Moreover, except the weights of the last two convolution layers, all other weights are frozen as shown in Fig. 12 (a typical transfer learning setting). Only the last two layers (18,563 parameters out of ~ 8.3 million) are re-trained using Client 9 data, and the comparison is shown in Fig. 13.

The performance on the Client 9 image is significantly better for the transfer learned network than when it participates in the federation. This is because the final two layers are not subject to federated averaging, rather, are specifically fine-tuned for Client 9's own data. These two layers use the geometric features extracted from the federated model (e.g., "there is an edge here"), but learn the part label specifically for Client 9 (e.g., "pixel intensities to the left of the edge denote a part in binder jetting").

This result has important implications in manufacturing in general. Using varying levels of transfer learning, generalizability of the global FL model (or personalization on the local client models) can be improved [47,48]. This can enable vastly different clients to privately benefit from each other's data in some capacity without worrying about negative transfer. Within an existing federation, the network can be constructed such that only specific layers participate in FedAvg and the rest remain 'free' to learn client-specific features [49]. More generally, weights for a completely different federation (performing the same task but on different data) can be initialized using a well-trained federation and then fine-tuned. This can reduce training time and ensure positive transfer of information, thereby enhancing FL generalizability for collaborative deep learning.

### 4.5. Learning behavior

In this section, the learning dynamics of federated training using FedAvg are discussed. Revisiting Case Study A, we consider the scenario of learning Clients 1 through 7 and testing on Client 8. The training losses are plotted in Fig. 14 and are discussed below.

In Fig. 14(a), the training loss in CL and the average training loss for all 7 clients in FL are plotted against the number of training epochs. In CL, the network converges (no further decrease in training loss) after

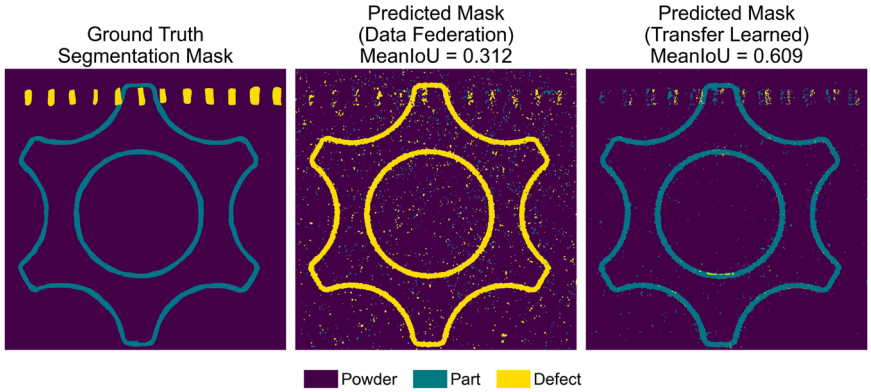


**Fig. 13.** Comparison between the predictions on a binder jetting image when Client 9 participates in the data federation vs. when transfer learning is utilized. The part is correctly labeled by the transfer learned network leading to a significant increase in MeanIoU.

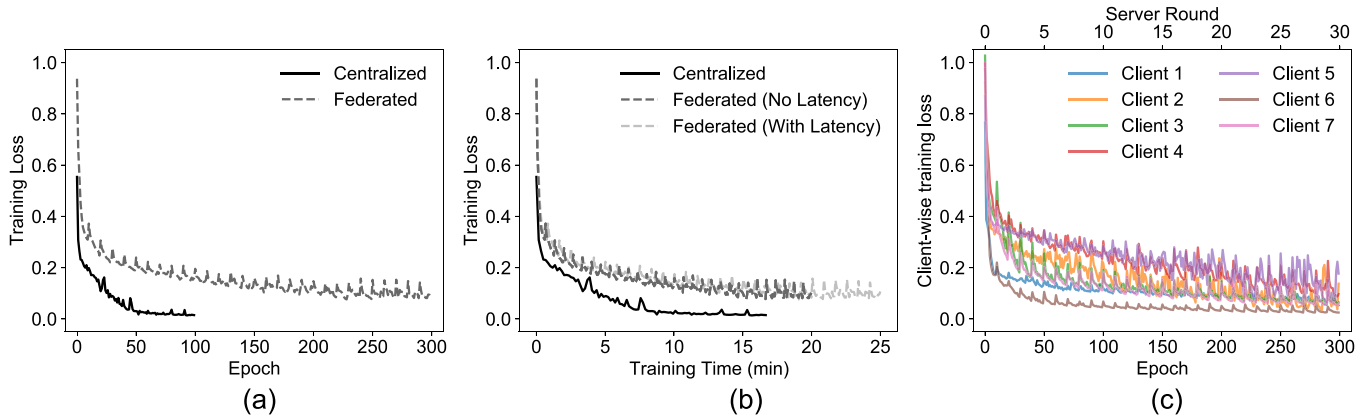


**Fig. 14.** Training loss for CL and average training loss for FL with (a) training epochs and (b) training time. Adding a 10 s latency after each server round in FL increases the training time by 25 %. (c) Training loss for individual clients.

$\sim$ 100 epochs. Contrarily, the global network takes more client epochs to converge, i.e., the network learns slower. A spike in the average training loss is observed after each server epoch in FL. This is indicative of a generalization step after every $E$ local epochs, where the local models 'absorb' information from other non-similar clients.

The difference in training epochs in Fig. 14(a) does not directly translate into training time, as shown in Fig. 14(b). For CL, the network iterates through $\sim$ 5800 tiles in each epoch which takes 10 s/epoch. Thus, the total training time for CL is 10 s/epoch $\times$ 100 epochs = 16.67 min. Contrarily for FL, local learning on each client happens in parallel (see Algorithm 1); thus, the training time for each server round is upper bounded by the training time for the largest client (w.r.t. the number of training examples). For instance, Client 5 is the largest client in the federation with 1066 training tiles. The local model for Client 5 takes 4 s/epoch for training. All other clients (with lesser number of training examples) take $\leq$ 4 s/epoch; thus, the total training time for FL is 4 s/epoch $\times$ 10 local epochs $\times$ 30 server rounds = 20 min. FL takes $\sim$ 1.2 times longer for training compared to CL, which can be considered a time cost for manufacturers for maintaining data privacy.

Note that the above time computation assumes no system-level latency in FL. The local model upload to the server, the global model download from the server, and the model averaging at the server, are assumed to be instantaneous. When a 10 s latency is added after each server round in FL, the training time increases by 25 % as seen in Fig. 14 (b). This can further be exacerbated by unequal computing resources available to each manufacturer. As such, in a real-world scenario, the training time for FL may not be straightforward to compute and must be evaluated for the task at hand.

The losses for each client are plotted separately in Fig. 14(c). It is clear that some clients (e.g., Client 6) are easier to train compared to others (e.g., Clients 4, 5). Further, Client 6 has no incentive to continue local computation after 10 server rounds, whereas Client 4 has incentive to continue local computation beyond 30 server rounds. This naturally leads to the idea of different hyperparameter settings for different clients, i.e., specifying $E_k$, $B_k$, and $\eta_k$ individually for client $k$. Recent work in adaptive federated optimization [50,51] has addressed this; however, it is beyond the scope of this paper and can be investigated separately as future work.

## 5. Discussion and future directions

The case studies in Section 4 demonstrate important facets of FL and its applicability in an AM production environment. To deploy FL at scale across multiple enterprises, some crucial challenges must be addressed. This paper unfolds several important research directions, some of which are discussed in this section.

### 5.1. Classifying multiple defect classes

We use a single defect class in this work which pools together individual defects like recoater streaking, soot, and misprint. In reality, not all defects significantly affect product quality and reliability; thus, manufacturers may be more interested in classifying some defects compared to others. The relevant defect class may also be different for each manufacturer in the federation. Thus, the weights assigned to the defect classes in the client empirical risks might differ. This is a more challenging problem which can be tackled using a more sophisticated network architecture and agreement between participating manufacturers on the FL loss and hyperparameters.

### 5.2. Enhancing privacy guarantees

FedAvg provides a rudimentary notion of privacy due to data minimization. When a data federation is formed across multiple organizations, formal privacy guarantees are necessary. Clients in FL are susceptible to adversarial attacks, e.g., data poisoning, model evasion, model update poisoning from a participating client. Cross-silo systems also need protection against data inference attacks, where a malicious client or the central server tries to infer information about the participants' data. Methods to protect client data by enhancing privacy like differential privacy, homomorphic encryption, secure multiparty computation, and blockchain technology have been discussed in FL literature. However, experiments on real-world AM data are required to evaluate their applicability in practice.

### 5.3. Personalization of client models

In Section 4.3, it is seen that adding heterogeneous clients to the federation dampens the global model performance. In a real-world scenario, the heterogeneity may not be as pronounced as different printing technologies, but more subtle like unit-to-unit variation between AM machines. The same type of machines can have different print quality even while printing similar parts [32,33,37]. Moreover, the same AM machine can be set up differently by different clients which results in dissimilar part qualities and defect types. To this end, personalized FL (PFL) is required to ensure only positive transfer of information across clients, which cannot be achieved using vanilla FL. In Section 4.4, PFL is achieved through transfer learning which enhances the generalizability of the global model (or personalization of the local models). More advanced techniques for PFL include meta-learning, multi-task learning, model regularization, and client clustering. Physics-informed hybrid models reported in [52,53] may also be extended to enable PFL.

### 5.4. Incentive design

In this work, all clients participating in the federation receive the same final model irrespective of their individual contributions. Thus, clients may worry that their data unfairly benefits other clients who do not contribute the same quality or amount of data (also called the free-rider problem [23,54]). To remedy this, designing sustainable incentive mechanisms for honest client participation is of foremost importance.

## 6. Conclusion

Limited data availability coupled with the sensitive and heterogeneous nature of data has limited the use of ML, especially deep learning, in AM production. In this work, an FL-based method is developed for pixel-wise defect detection in metal AM and is shown to alleviate these challenges simultaneously. The presented case studies demonstrate that when participating clients use the same printing technology, the defect detection performance of FL is comparable to CL and significantly better than IL. The manufacturers participating in FL compromise neither model accuracy nor data privacy. Further, increasing data diversity within individual clients or by adding new clients can improve FL performance. However, adding significantly different clients (e.g., clients from a different printing technology) may hurt an existing federation. To remedy this, transfer learning is proposed to make FL more generalizable. Lastly, to deploy FL in an AM production environment, some crucial challenges are discussed along the lines of privacy, personalization, and incentives.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The authors would like to thank the authors of the layer-wise imaging dataset at the Oak Ridge National Laboratory for compiling and releasing the pixel-wise annotated data. Support for DOI 10.13139/ORNLNCCS/1779073 dataset is provided by the U.S. Department of Energy, project Peregrine under Contract DE-AC05-00OR22725. Project Peregrine used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

## Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.jmsy.2022.06.010.

## References

[1] Ye Z, Liu C, Tian W, Kan C. In-situ point cloud fusion for layer-wise monitoring of additive manufacturing. J Manuf Syst 2021;61:210–22. https://doi.org/10.1016/J.JMSY.2021.09.002.
[2] Jafari-Marandi R, Khanzadeh M, Tian W, Smith B, Bian L. From in-situ monitoring toward high-throughput process control: cost-driven decision-making framework for laser-based additive manufacturing. J Manuf Syst 2019;51:29–41. https://doi.org/10.1016/J.JMSY.2019.02.005.
[3] Seifi SH, Tian W, Doude H, Tschopp MA, Bian L. Layer-wise modeling and anomaly detection for laser-based additive manufacturing. J Manuf Sci Eng Trans 2019;14(8). https://doi.org/10.1115/1.4043898/726779.
[4] Wang C, Tan XP, Tor SB, Lim CS. Machine learning in additive manufacturing: state-of-the-art and perspectives. Addit Manuf 2020;36:101538. https://doi.org/10.1016/J.ADDMA.2020.101538.
[5] Scime L, Siddel D, Baird S, Paquit V. Layer-wise anomaly detection and classification for powder bed additive manufacturing processes: a machine-agnostic algorithm for real-time pixel-wise semantic segmentation. Addit Manuf 2020;36:101453. https://doi.org/10.1016/J.ADDMA.2020.101453.
[6] Jin Z, Zhang Z, Ott J, Gu GX. Precise localization and semantic segmentation detection of printing conditions in fused filament fabrication technologies using machine learning. Addit Manuf 2021;37:101696. https://doi.org/10.1016/J.ADDMA.2020.101696.
[7] Wen V, Wong H, Ferguson M, Law KH, Lee Y-TT, Witherell P. Automatic volumetric segmentation of additive manufacturing defects with 3D U-net ; 2021. arXiv:2101.08993.
[8] Minnema J, van Eijnatten M, Kouw W, Diblen F, Mendrik A, Wolff J. CT image segmentation of bone for medical additive manufacturing using a convolutional neural network. Comput Biol Med 2018;103:130–9. https://doi.org/10.1016/J.COMPBIOMED.2018.10.012.
[9] Wang Y, Lu J, Zhao Z, Deng W, Han J, Bai L, et al. Active disturbance rejection control of layer width in wire arc additive manufacturing based on deep learning, vol. 67; 2021, p. 364–75. ⟨https://doi.org/10.1016/j.jmapro.2021.05.005⟩.
[10] Li R, Peng Q. Deep learning-based optimal segmentation of 3d printed product for surface quality improvement and support structure reduction. J Manuf Syst 2021;60:252–64. https://doi.org/10.1016/j.jmsy.2021.06.007.
[11] Gao Y, Li X, Wang XV, Wang L, Gao L. A review on recent advances in vision-based defect recognition towards industrial intelligence. J Manuf Syst 2021. https://doi.org/10.1016/J.JMSY.2021.05.008.
[12] Guo S, Agarwal M, Cooper C, Tian Q, Gao RX, Guo WG, et al. Machine learning for metal additive manufacturing: towards a physics-informed data-driven paradigm. J Manuf Syst 2022;62:145–63. https://doi.org/10.1016/J.JMSY.2021.11.003.
[13] Qi X, Chen G, Li Y, Cheng X, Li C. Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspectives. Eng 2019;5(4):721–9. https://doi.org/10.1016/J.ENG.2019.04.012.
[14] Meng L, McWilliams B, Jarosinski W, Park HY, Jung YG, Lee J, et al. Machine learning in additive manufacturing: a review. JOM 2020;72(6):2363–77. https://doi.org/10.1007/S11837-020-04155-Y/FIGURES/11.
[15] Deng J, Dong W, Socher R, Li L-J, Li Kai, Fei-Fei Li, ImageNet: a large-scale hierarchical image database; 2010, p. 248–55. ⟨https://doi.org/10.1109/CVPR.2009.5206848⟩.
[16] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: common objects in context. Lect Notes Comput Sci (Incl Subser Lect Notes Artif Intell Lect Notes Bioinforma) 8693 LNCS 2014:740–55. https://doi.org/10.1007/978-3-319-10602-1_48.
[17] Everingham M, Van Gool L, Williams CK I, Winn J, Zisserman A, Everingham M, et al. The Pascal visual object classes (VOC) challenge. Int J Comput Vis 2009;88(2):303–38. https://doi.org/10.1007/S11263-009-0275-4 [2009 882].
[18] Kontar R, Shi N, Yue X, Chung S, Byon E, Chowdhury M, et al. The internet of federated things (IoFT). IEEE Access 2021;9:156071–113. https://doi.org/10.1109/ACCESS.2021.3127448.
[19] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning. ACM Trans Intell Syst Technol 2019;10(2). https://doi.org/10.1145/3298981.
[20] McMahan B, Moore E, Ramage D, Hampson S, Arcas BAy. Communication-efficient learning of deep networks from decentralized data; 2017.
[21] Li T, Sahu AK, Talwalkar A, Smith V. Federated learning: challenges, methods, and future directions. IEEE Signal Process Mag 2019;37(3):50–60. https://doi.org/10.1109/MSP.2020.2975749.
[22] Li L, Fan Y, Tse M, Lin KY. A review of applications in federated learning. Comput Ind Eng 2020;149:106854. https://doi.org/10.1016/J.CIE.2020.106854.
[23] Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, et al. Advances and open problems in federated learning. Found Trends Mach Learn 2019;14(1–2):1–210. https://doi.org/10.1561/2200000083.
[24] Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. Lect Notes Comput Sci (Incl Subser Lect Notes Artif Intell Lect Notes Bioinforma) 2015;9351:234–41.
[25] Hertlein N, Buskohl PR, Gillman A, Vemaganti K, Anand S. Generative adversarial network for early-stage design flexibility in topology optimization for additive manufacturing. J Manuf Syst 2021;59:675–85. https://doi.org/10.1016/J.JMSY.2021.04.007.
[26] Zhang Y, Zhao YF. Hybrid sparse convolutional neural networks for predicting manufacturability of visual defects of laser powder bed fusion processes. J Manuf Syst 2021. https://doi.org/10.1016/J.JMSY.2021.07.002.
[27] Mojahed Yazdi R, Imani F, Yang H. A hybrid deep learning model of process-build interactions in additive manufacturing. J Manuf Syst 2020;57:460–8. https://doi.org/10.1016/J.JMSY.2020.11.001.
[28] Westphal E, Seitz H. Machine learning for the intelligent analysis of 3D printing conditions using environmental sensor data to support quality assurance. Addit Manuf 2022;50:102535. https://doi.org/10.1016/J.ADDMA.2021.102535.
[29] Khanzadeh M, Chowdhury S, Marufuzzaman M, Tschopp MA, Bian L. Porosity prediction: supervised-learning of thermal history for direct laser deposition. J Manuf Syst 2018;47:69–82. https://doi.org/10.1016/J.JMSY.2018.04.001.
[30] Snow Z, Diehl B, Reutzel EW, Nassar A. Toward in-situ flaw detection in laser powder bed fusion additive manufacturing through layerwise imagery and machine learning. J Manuf Syst 2021;59:12–26. https://doi.org/10.1016/J.JMSY.2021.01.008.
[31] Ferreira RDSB, Sabbaghi A, Huang Q. Automated geometric shape deviation modeling for additive manufacturing systems via bayesian neural networks. IEEE Trans Autom Sci Eng 2020;17(2):584–98. https://doi.org/10.1109/TASE.2019.2936821.
[32] McGregor DJ, Rylowicz S, Brenzel A, Baker D, Wood C, Pick D, et al. Analyzing part accuracy and sources of variability for additively manufactured lattice parts made on multiple printers. Addit Manuf 2021;40:101924. https://doi.org/10.1016/J.ADDMA.2021.101924.

[33] McGregor DJ, Bimrose MV, Shao C, Tawfick S, King WP. Using machine learning to predict dimensions and qualify diverse part designs across multiple additive machines and materials. Addit Manuf 2022:102848. https://doi.org/10.1016/j.addma.2022.102848.
[34] Wang J, Ma Y, Zhang L, Gao RX, Wu D. Deep learning for smart manufacturing: methods and applications. J Manuf Syst 2018;48:144–56. https://doi.org/10.1016/J.JMSY.2018.01.003.
[35] Scime L, Paquit V, Joslin C, Richardson D, Goldsby D, Lowe L. Layer-wise imaging dataset from powder bed additive manufacturing processes for machine learning applications (Peregrine v2021-03); 2021. ⟨https://doi.org/10.13139/ORNLNCCS/1779073⟩.
[36] Rezatofighi H, Tsoi N, Gwak J, Sadeghian A, Reid I, Savarese S. Generalized intersection over union: a metric and a loss for bounding box regression. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2019:658–66. https://doi.org/10.1109/CVPR.2019.00075.
[37] Yang Y, McGregor DJ, Tawfick S, King WP, Shao C. Hierarchical data models improve the accuracy of feature level predictions for additively manufactured parts. Addit Manuf 2022;51:102621. https://doi.org/10.1016/J.ADDMA.2022.102621.
[38] Goodfellow IJ, Vinyals O, Saxe AM. Qualitatively characterizing neural network optimization problems. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.; 2014.
[39] Shelhamer E, Long J, Darrell T. Fully convolutional networks for semantic segmentation. IEEE Trans Pattern Anal Mach Intell 2014;39(4):640–51. https://doi.org/10.1109/TPAMI.2016.2572683.
[40] Kingma DP, Ba JL. Adam: a method for stochastic optimization. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.; 2014.
[41] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25.
[42] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: Proc. IEEE comput. soc. conf. comput. vis. pattern recognit. 07-12-june-2015; 2014, p. 1–9. ⟨https://doi.org/10.1109/CVPR.2015.7298594⟩.
[43] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-December; 2015, p. 770–8. ⟨https://doi.org/10.1109/CVPR.2016.90⟩.
[44] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.; 2014.
[45] Gong Z, Zhong P, Hu W. Diversity in machine learning. IEEE Access 2019;7:64323–50. https://doi.org/10.1109/ACCESS.2019.2917620.
[46] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng 2010;22(10):1345–59. https://doi.org/10.1109/TKDE.2009.191.
[47] Dinh CT, Tran NH, Nguyen TD. Personalized federated learning with Moreau envelopes. Adv Neural Inf Process Syst 2020.
[48] Li T, Hu S, Beirami A, Smith V. Ditto: fair and robust federated learning through personalization. In: International conference on machine learning. PMLR; 2021, p. 6357–68.
[49] Arivazhagan MG, Aggarwal V, Singh AK, Choudhary S. Federated Learning with Personalization Layers; 2019. arXiv:1912.00818.
[50] Wang S, Tuor T, Salonidis T, Leung KK, Makaya C, He T, et al. Adaptive federated learning in resource constrained edge computing systems. IEEE J Sel Areas Commun 2019;37(6):1205–21. https://doi.org/10.1109/JSAC.2019.2904348.
[51] Reddi SJ, Charles Z, Zaheer M, Garrett Z, Rush K, Konečný J, et al. Adaptive federated optimization; 2020. arXiv:2003.00295.
[52] Shao C, Ren J, Wang H, Jin J, Hu SJ. Improving machined surface shape prediction by integrating multi-task learning with cutting force variation modeling. J Manuf Sci Eng Trans 2017;139(1). https://doi.org/10.1115/1.4034592/472250.
[53] Yang Y, Shao C. Hybrid multi-task learning-based response surface modeling in manufacturing. J Manuf Syst 2021;59:607–16. https://doi.org/10.1016/J.JMSY.2021.04.012.
[54] Fraboni Y, Vidal R, Lorenzi M. Free-rider attacks on model aggregation in federated learning; 2020. arXiv:2006.11901.