Defect Detection Automation for Metal Additive Manufacturing With Deep Learning

Collin Broderick

March 4, 2022

A Major Qualifying Project submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of Bachelor of Science

Authors:

Collin Broderick

Date:

March 4, 2022

Report Submitted to:

Professor Rodica Neamtu, Computer Science, Project Advisor
Professor Diana Lados, Materials Science, Project Co-advisor

Worcester Polytechnic Institute

Authorship

This publication reflects the common work of Collin Broderick and Adrianna Staszewska. This submission details all work completed up until its publication, and will be followed by a separate report detailing the additional work Adrianna will complete.

Acknowledgements

We would like to thank our project advisor, Professor Rodica Neamtu, for providing us this project opportunity as well as countless suggestions, encouragements, and the overall coordination we needed to overcome the challenges associated with developing this model.

We would also like to thank our project co-advisor, Professor Diana Lados, for providing us invaluable insights into the nature of the materials science domain of this project, as well as acting as an essential bridge between the computer science and mechanical engineering students who made this project possible.

Additionally, we wish to acknowledge Akshatha Chandrashekar Dixith, Deepali Purushottam and Bonnie Whitney for their crucial knowledge regarding the intersection of computer vision and additive manufacturing.

Finally, we thank the Material Science team: Patick Bowles, Charles Carlo, Gabriella Cervone and Braeden Desmonts for your hard work on delivering and labeling the data for this project.

Abstract

Metal 3D Printing, or Additive Manufacturing (AM), is a layer-by-layer manufacturing process that involves adding the material to create a structure instead of removing it. In contrast to conventional manufacturing techniques, AM allows for much faster, low-cost prototyping, the construction of the parts with complex geometries, and an overall reduction in material waste. However, AM is a complex process governed by many parameters, which are often interdependent and impacted by the manufacturing environment. Optimization of these parameters is crucial to ensure the high quality of the manufactured parts. Manual quality inspection of these parts is not only a time-consuming process but also depends on the skill of the quality control inspector. This project aims to utilize Deep Learning techniques, specifically Convolutional Neural Networks, to identify, localize and classify defects from cross-sectional image scans of aluminum Al-10Si-Mg parts manufactured via laser powder bed fusion to automate the defect detection and quality assessment for those parts.

To achieve the localization and classification of defects on these image scans, we utilized Matterport Inc.'s implementation of Mask R-CNN, a robust Region-Based Convolutional Neural Network designed for the detection of objects present in high-dimensional images [1]. Image scans for use in the model were generated and annotated on a rolling basis. As the data became available, several steps of data cleaning were necessary to properly train Mask R-CNN with these image scans. These steps included generating pixel-perfect masks corresponding to each annotated defect, thresholding any defects considered to small to classify, separating classes into sub-classes to ensure a balance of data, and in some cases stitching entire samples and splitting along custom boundaries to ensure the geometries of extreme defects were preserved. After several iterations of the model, the final model was trained on 13 different sets with 1349 overall instances of defects, and achieved an overall classification accuracy of 69.23%. Based on these promising results, our paper outlines possible opportunities to improve the overall classification performance of the model, as well as the potential for the data to be utilized in further research. The project will be continued in D-term by one of the team members.

Table of Contents

Table of Figures

1 Introduction

Additive manufacturing (AM), commonly known as 3D printing, is the "process of joining materials to make objects from 3D model data, usually layer upon layer, as opposed to subtractive manufacturing and formative manufacturing methodologies" [2]. AM is an increasingly popular method of production as it enables rapid prototyping, lower prototyping costs, reduced waste, and the creation of part geometries not previously possible with conventional manufacturing techniques. In 2020, the estimated value of the 3D printing market was $12.6bn [3]. One of the areas where AM is getting more attention in recent years is the aerospace industry. Aerospace components are exposed to various conditions that can cause cracks and defects which can impact the quality of the material. With the high production cost of these components, they need to be repaired instead of being replaced. AM technologies provide new opportunities to repair these parts; components previously labeled as "unweldable", due to the chance of distortion due to the high heat input of welding, can now be repaired with AM [4]. AM technologies are also found in various applications in the healthcare field: personalized implants, saw guides, splints, and artificial tissue are just some examples of specialized parts made using this technology [5]. With a wide range of applications, AM market is expected to grow by 17 percent year-over-year, reaching $37.2bn in 2026 [3]. One particular subset of AM that has been gaining attention due to its wide applicability to a number of industries is laser powder bed fusion (LPBF). With this method of printing, a variety of metal alloys can be fabricated with complex geometries, opening many possibilities to enhance the capabilities of metal parts. However, LPBF is a complex process dependent on many parameters, both related to the properties of the material, the process, and the environment. During printing, defects in the material can appear due to improper process parameters, which include laser power, scan speed, and hatch spacing. These defects often cannot be eliminated with the post-processing of the part. Undetected presence of these defects can lead to change in the material properties. For this reason, there is a need for a better understanding of how the changes in the manufacturing process and the environment impact the properties of the manufactured parts. To do this, the quality of the parts manufactured under different conditions needs to be assessed. Manual quality inspection of these parts is not only a time-consuming process but also depends on the skill of the quality control inspector.

In lieu of these constraints, our team approached the issue of LPBF part inspection by creating a pipeline that will detect and classify defects as well as return the final porosity score. Pipeline is a series of chained operations, where the output of the previous operation serves as the input into the next one [6] . We identified two main components of our pipeline: defect region segmentation and defect classification. Each of these components can be developed and verified independently of the others but in the end, the modules can be combined into a unified pipeline. To achieve this, we utilized an Agile software development methodology.

2 Background

2.1 Additive Manufacturing

Additive Manufacturing, or 3D printing, is a process of constructing a 3D object layer-by-layer from a digital model [2]. Different AM technologies can be differentiated by the material feedstock, type of energy source, and the method of material feed. Some of those popular additive manufacturing technologies today are binder jetting, directed energy deposition, material extrusion, and powder bed fusion. This project focuses on metal laser powder bed fusion (LPBF), which utilizes powder bed fusion technology and metallic powder to fabricate metal parts.

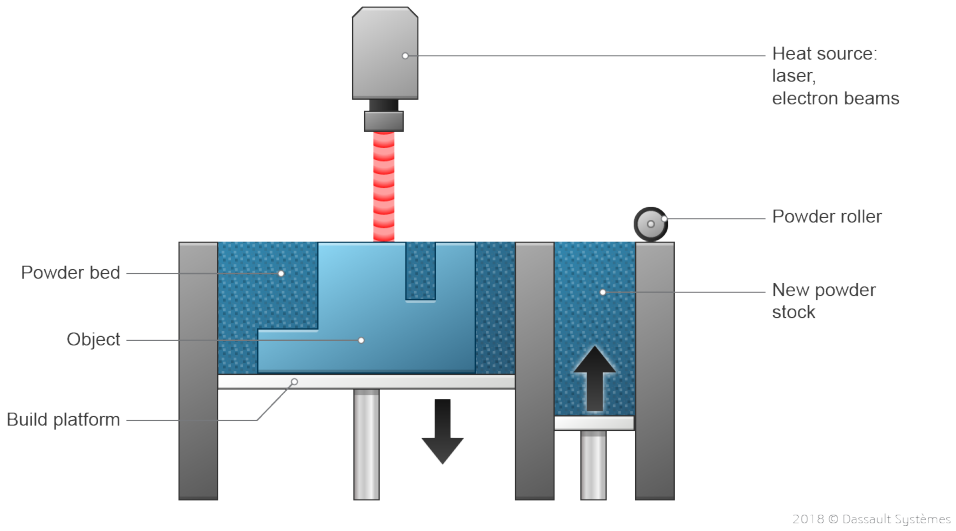


Figure 1: LPBF schematic [7].

Laser powder bed fusion involves depositing layers of powder material over a bed and solidifying that material via laser fusion. A LPBF printer will contain two chambers: a build platform and a powder chamber that contains new powder stock. First, the powder chamber is filled with powdered build material. A powder roller then deposits a thin layer of the material across the build platform. Next, the heat source melts the deposited powder on the top layer according to a 2D cross-section from part of the model being printed. When the layer has been scanned and fused, the build platform is lowered, and the powder chamber is raised. The powder roller then deposits the material onto the build platform again, and the process repeats until the entire part is fabricated.

A variety of defects can occur during the fusion process that compromise the quality of the final part. These defects are keyhole porosity, gas porosity, lack-of-fusion porosity, cracks, and balling. The smallest of the defects, keyhole porosity is created from fluid instabilities at high power and low scan speeds. Gas porosity is formed when gas diffuses into the melt pool during the solidification of melted powder, creating small elliptical pores in the part. Lack-of-fusion porosity is formed due to deficiency in energy input during solidification, creating large and irregularly shaped wedges. The biggest imperfections, cracks, form due to thermal residual stresses generated from large temperature gradients and rapid solidification during AM processing, compromising the mechanical integrity of the part. The last type of defect, balling, occurs when melted material shrinks into a sphere for one of several reasons: a rough sub-layer, high porosity, or splashing of adjacent material on the bed.

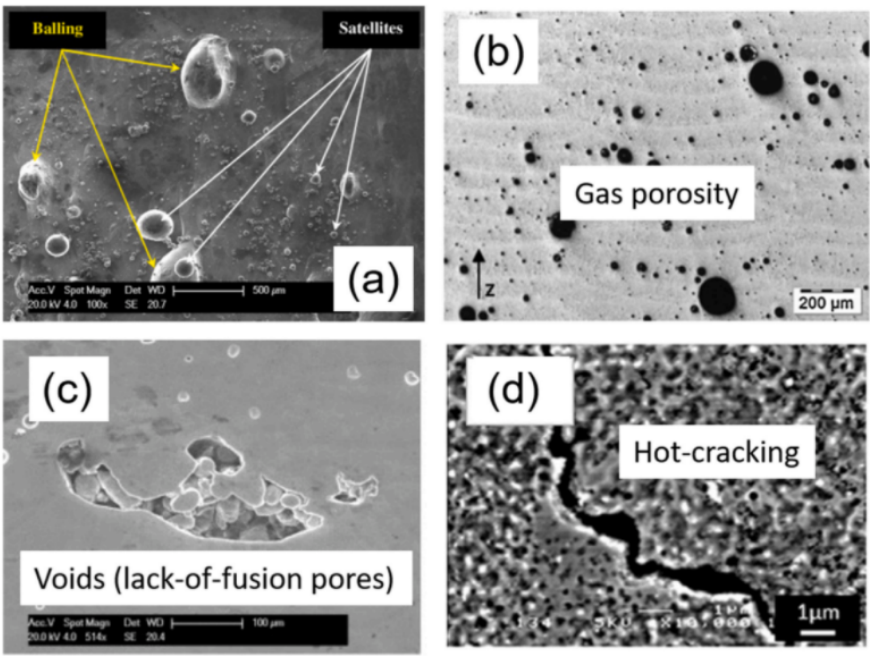


Figure 2: Depiction of the several types of defects in Al-Si alloys using LBPF: (a) balling, (b) gas pores, (c) lack-of-fusion, (d) hot cracking [8].

Post-processing can be used on LPBF parts to enhance its mechanical and microstructure properties [9]. Mechanical treatments such as laser polishing are effective in reducing the surface roughness of the final part. Chemical processes such as thermal-based treatments can improve mechanical hardness. Hot isostatic pressing is a type of thermal treatment that uses gas pressure to reduce the porosity of a part. However, post processing is ineffective in addressing many of the types of defects that are responsible for creating parts substandard to conventionally manufactured parts and adds additional time to the manufacturing process. Therefore, the aim should be to reduce porosity as much as possible at the front end by optimizing process parameters such as laser power, scan speed, and hatch spacing.

2.2 Deep Learning

The field of Additive Manufacturing is presented with challenges that can be approached with the help of Artificial Intelligence (AI), "the study of agents that receive percepts from the environment and perform actions" [10]. Machine Learning is a subfield of Artificial Intelligence that explores the ability of computers to learn patterns from the data without being explicitly programmed [11].

Machine Learning models are often categorized based on whether they are trained using labeled data or not. Fully supervised systems are fed data that includes the desired answers - labels. The algorithm makes a prediction on the given input, a set of features, and compares it to the desired output. On the other hand, during unsupervised training, the model is not given the expected output. Here instead of trying to make accurate predictions based on known answers, the model tries to discover patterns and structure in the data. One of the challenges of fully supervised learning is the amount of labeled data needed, as labeling is often a time-consuming process. Semi-supervised learning can be applied in this case which is usually a combination of supervised and unsupervised algorithms.

2.2.1 Neural Networks

By comparison to traditional Machine Learning where feature extraction is done by humans and then those features are provided to the model; deep learning models learn the features. Because

Deep Learning relies on representation learning, it requires significant amounts of data. Fundamental to the idea of Deep Learning is the concept of neural networks, which were designed to simulate the way the human brain works. Similar to how the brain is a network of neurons, neural networks are composed of nodes, which are organized into layers.

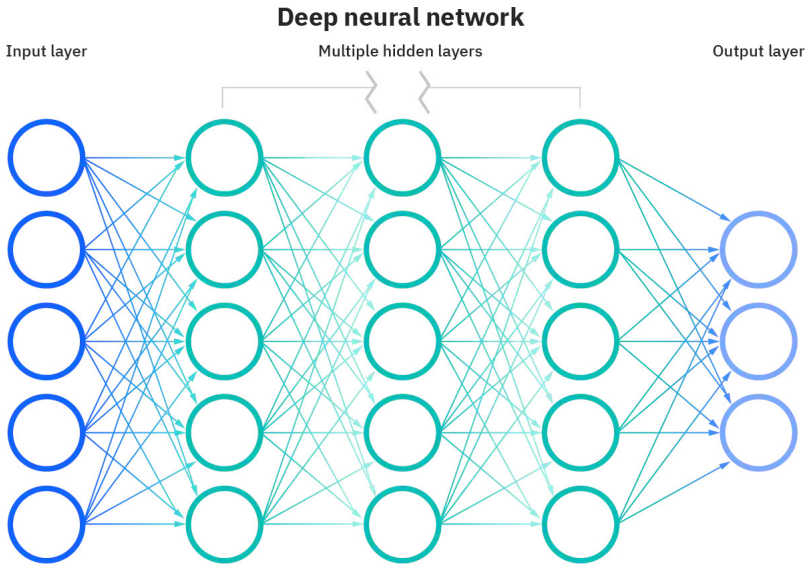


Figure 3: Deep neural network architecture [12].

Each node is a linear regression unit, with an input function, an activation function, and an output. The input function computes the weighted sum of input vector $\mathbf{x}$. Each edge in the network is associated with a weight, and for each layer in the network, there exists a corresponding weight vector θ. The activation function, ϕ, which is typically a nonlinear function, decides whether a node should be activated or not. Therefore, the output of each node can be written as:

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \phi(\boldsymbol{\theta}^T \mathbf{x} + \mathbf{b}) \tag{1}$$

Therefore, activations $a^(j)$ for the entire layer j can be computed in the following way:

$$\mathbf{z}^{(j)} = \Theta^{(j)} \mathbf{a}^{(j-1)} + \mathbf{b}^{(j)} \tag{2}$$

$$\mathbf{a}^{(j)} = \phi(\mathbf{z}^{(j)}) \tag{3}$$

Before training, all the weights are randomly initialized. During training, they are adjusted with backpropagation. After the single pass through a neural network, a prediction $\hat{y}$ is made, some loss function $L(y, \hat{y})$ is used to calculate how close the prediction was to the expected answer. This measure can be used to update all the parameters (weights and bias terms) of the network, for each layer j, where α is the learning rate:

$$\Theta^{(j)} = \Theta^{(j)} - \alpha \frac{\partial L}{\partial \Theta^{(j)}} \tag{4}$$

$$\mathbf{b}^{(j)} = \mathbf{b}^{(j)} - \alpha \frac{\partial L}{\partial \mathbf{b}^{(j)}} \tag{5}$$

Since we want to minimize the loss, partial derivatives are used to find the rate of steepest loss L in respect to θ and $\mathbf{b}$. This process is done for every training example.

2.2.2 Computer Vision

Computer Vision is a field in Computer Science that deals with image and video data. Computer vision techniques are used in a variety of applications, such as object recognition for automated checkout lanes, fingerprint recognition for automatic access authentication, construction of 3D models from aerial photographs and rapid parts inspection for quality assurance [13].

Computer vision is a challenging problem that requires substantial fine-tuning and computational strength to achieve a reasonable efficacy. While identifying and understanding an image is a relatively intuitive task for a person who has had a lifetime of learning and experience with the natural world, computer vision models are trained in a relatively short period of time with limited data. Image data is high dimensional and noisy, and the same image class can be shown from multiple perspectives. An effective computer vision model must be able to identify features in an image based on a large set of examples from different contexts, while recognizing and filtering extraneous data.

Convolutional Neural Networks Convolutional Neural Networks (CNNs) are one of the main techniques used in the field of computer vision. CNN architecture mimics the way human eye deals with image inputs, each neuron in the brain reacts only to stimuli located in a limited regions – no neuron sees the whole image at once [14]. That is why, as opposed to Multi-Layer Perceptron (MLP), where all layers are fully connected, in CNN most layers are sparse. This allows the CNN to process local (pixels located close to one another) rather than global information that then is passed in more generalized form to the next layer [15]. For this reason, CNNs are composed of several building blocks: convolution layers, pooling layers, and fully connected layers not used in standard neural network architecture.

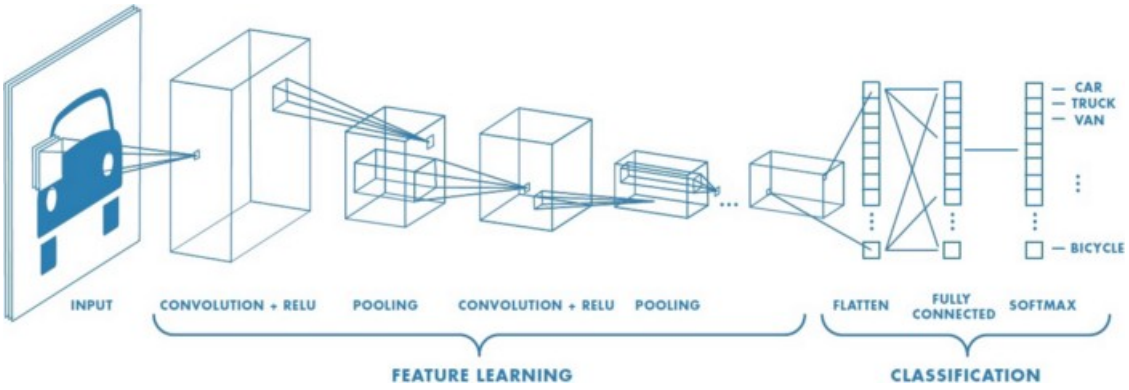


Figure 4: Neural network with many convolutional layers [16].

Convolution layers are, as the name of these types of networks suggests, the most important and identifying aspect of CNNs. The convolutional layer is responsible for convolving input images to extract the most useful features for the purpose of detection. This is achieved by applying what is known as a filter, or a matrix of weights that modifies the image on a per-pixel basis. Before being passed into further convolutional layers, this output is passed through a pooling layer, where the image is down sampled to achieve a reduction in dimensionality for optimization purposes. After being convolved and down sampled over the specified number of convolutional layers, the image data passes through a fully connected layer. Here the image is flattened to a vector and classified based on its probability of belonging to one of several predetermined classes.

Convolutional Neural Network for defect detection There are a lot of examples in literature of CNN use for defect detection in AM. Cui et al. [17] used Convolutional Neural Networks to classify types of porosity in optical images of parts build with laser metal deposition. Even their data was limited, as AM data is quite costly to get, they applied data augmentation techniques like rotation, flipping and blur to increase their sample size. Each sample was only a 224x224 section of an original image and only one or no defects. Images were classified into four classes: crack, gas porosity, lack of

fusion and good quality. They applied L2 regularization and dropout techniques to avoid overfitting. 5 shows the architecture of their CNN.

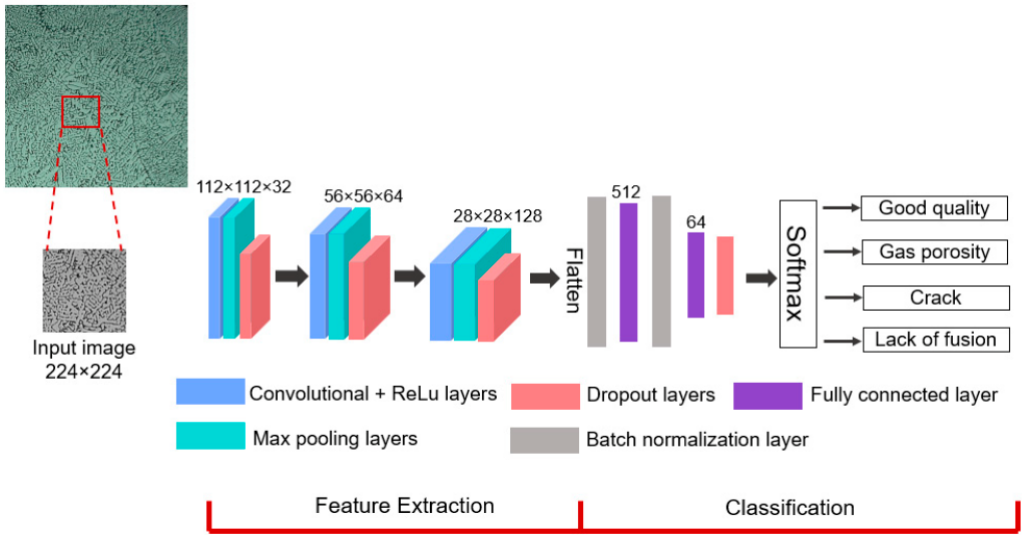


Figure 5: CNN architecture for defect classification [17].

However, this approach has some shortcomings. The authors do not propose a technique for defect localization, and the algorithm is fed already segmented instances, not the entire scan. Instances were manually selected which means that data was highly preprocessed and might not mirror real-world data as "some of the image blocks were not selected as they consisted of unusable regions, such as the mounting epoxy materials" [17]. Second, since images are already segmented from the original data, the location of a particular pore is not taken into the account when classifying. This might not be an issue for defect classification, but it creates challenges for porosity prediction task.

Tao et al. [18] proposed a cascaded autoencoder (CASAE) architecture for defect segmentation and localization. The autoencoder network first transforms the original image into a prediction mask. This mask is then processed by the threshold module to refine the result of the prediction mask and obtain an accurate defect contour. Then candidate defect regions are extracted, and defects are cropped into separate images (minimum enclosing rectangle for each defect). Each of these regions is then classified by the compact CNN. CASAE was able to achieve 89.60% intersection-over-union and the classification module achieved 86.82% accuracy.

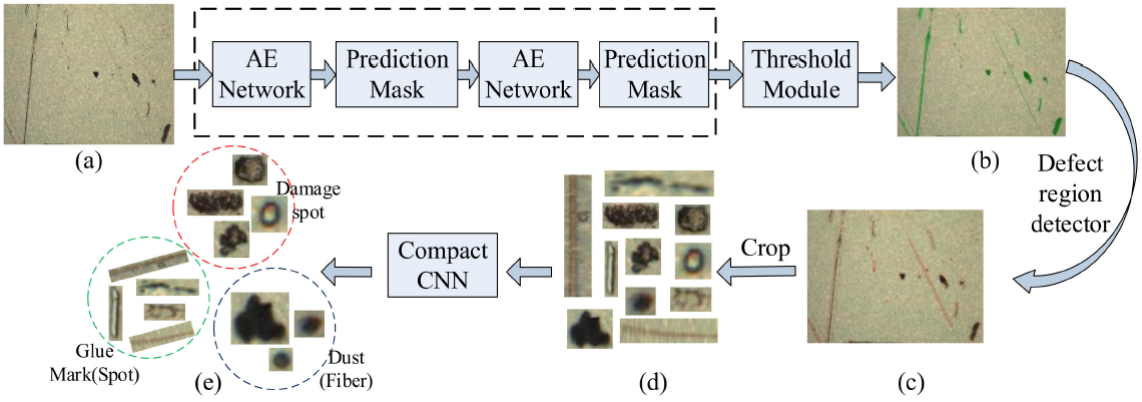


Figure 6: The pipeline of the proposed metallic surface defect inspection architecture [18].

Here the neural network localizes the defects and segments them but again the information about each individual pore is separated from the entire image with not reference point to its original

location. Since defects are segmented into bounding boxes, they do not represent defects' edges and do not describe their shape. For this project, a more robust algorithm is needed that can localize the defects with greater precision.

R-CNN and Image Segmentation R-CNN describes a subset of CNN optimized for image segmentation. R-CNNs introduces the idea of regional proposals to CNNs, where a set of discrete bounding boxes are identified before classification as opposed to dynamically generating an arbitrary number of regions (Girshick et al., 2014). The regions are identified with a selective search algorithm, which generates segmentations based on similarities: color similarity, texture similarity, size similarity, and fill similarity. Figure 9. gives a simple schematic of an R-CNN. Fast R-CNN was developed to address some of the issues R-CNN had, notably its multi-staged approach and high computational cost. Fast R-CNN differs from R-CNN by pooling features corresponding to each region proposal and sharing computations between overlapping regions [19]. Additional optimizations were introduced by Shaoqing et al. in 2017 when they proposed Faster R-CNN. This model introduced the Regional Proposal Network (RPN), which aimed to share convolutional layers with object detection networks. RPNs provide end-to-end predictions of regions and classification scores.

Most recently, Mask R-CNN was developed as an extension of Faster R-CNN. Mask R-CNN introduces a third output branch to the network: an object mask [20]. This additional mask gives a much finer feature extraction than previously possible, creating classifications that occur on a per-pixel basis. 7 shows the output of Mask-RCNN and its addition of masks to region proposals.

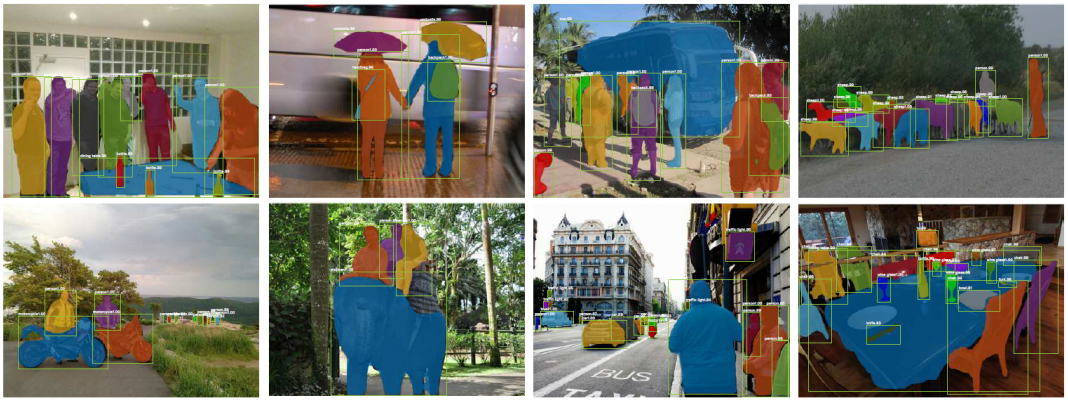


Figure 7: Mask R-CNN results [20].

Han et al. [21] were able to train a binary classification model based on R-CNN for identifying defects on a dataset of 773 labeled AM part scans, 1200 x 1600 each from the AM Defect Recognition Benchmark Dataset. The defects were classified as porosity, porosity array, or crack. Their model contained a pipeline of networks that served to accomplish the following tasks: read part images and adapt input image size, identify candidate areas for defect features, pass the candidate area and a convolution feature map through a RoIAlign layer, used in CNNs to obtain an area feature map, and finally pass this feature map through two parallel maps to obtain target box coordinates and binary masks for the defect instances. Due to the relatively small size of the dataset available, a technique known as transfer learning was used to augment the training process, where weights learned from the COCO (Common Objects in Context) image recognition data set was transferred to the defect recognition network. The network achieved a mean average precision of 89.4 across all types of defects instances in all images, outperforming ResNet, Faster R-CNN, and YOLOv3 on the dataset used. Figure 8. gives a labeled sample from the dataset used, and Figure 9 gives the described network architecture.

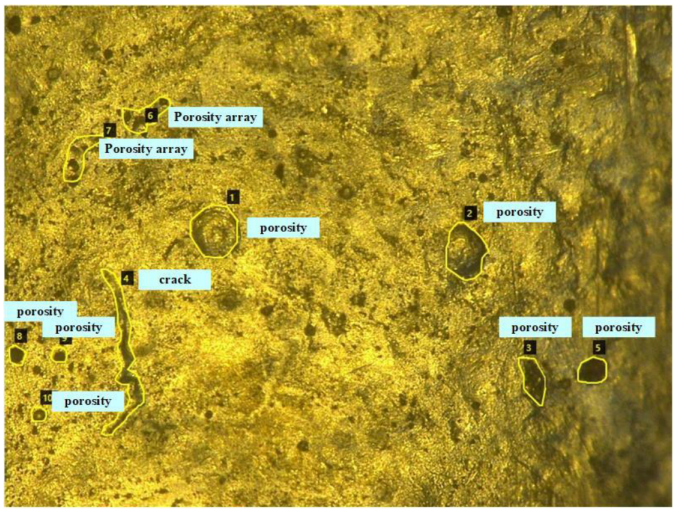


Figure 8: Defect detection dataset labeled sample [21].

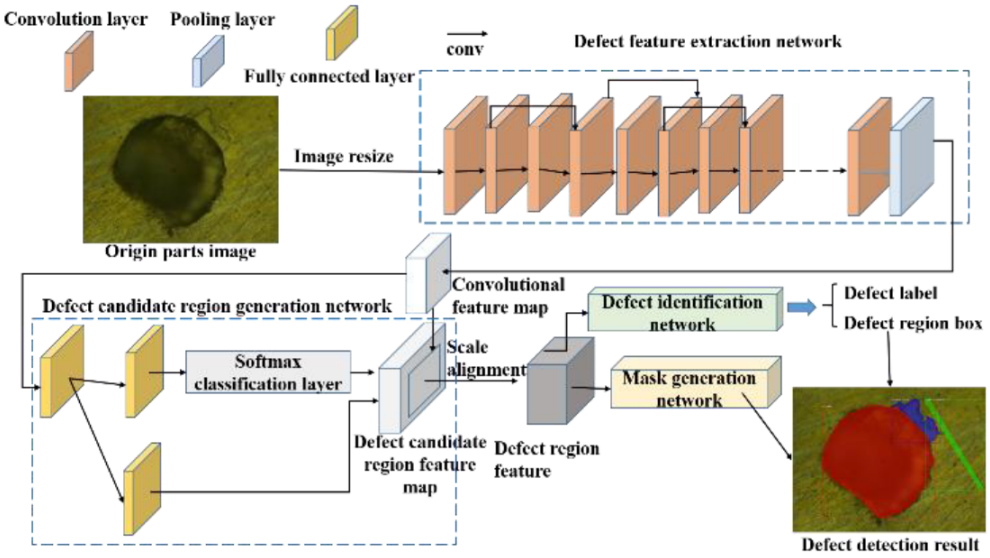


Figure 9: Defect Detection Network Architecture [21].

The superior performance of this multi-layered approach to more conventional segmentation approaches such as Faster RCNN and YOLO suggest that compartmentalizing the tasks of region detection, mask generation, and classification to separate networks could prove a promising approach to optimizing classification accuracy.

Porosity Prediction with Convolutional Neural Networks Defect detection is not the only area in Additive Manufacturing where CNNs were found to be successful. Ho et al. [15] proposed the use of Residual-Recurrent Convolutional Neural Networks (Res-RCNN) for "effectively performing the end-to-end porosity prediction in real-time using thermal images of melt pool". Similarly, to He et al. [20] transfer learning was used to decrease the amount of data needed. Authors of this paper decided that pure CNN might not be the best approach to the problem they were facing as "the context information from the higher layers of a deep feed-forward CNN model may fail to modulate the activities of the lower layer neurons that are responsible for detecting smaller objects such as pores in melt pools"[15]. RCNNs solve this problem by introducing a new type of layer to the standard CNN architecture – Recurrent Convolutional Layer (RCL) which has not only feed-forward connections but also recursive connections within the same layer. These additional connections allow for the model to

better capture both local and global context as they increase the receptive field of each individual pixel, compared to pure CNN which mostly focuses on the local context.

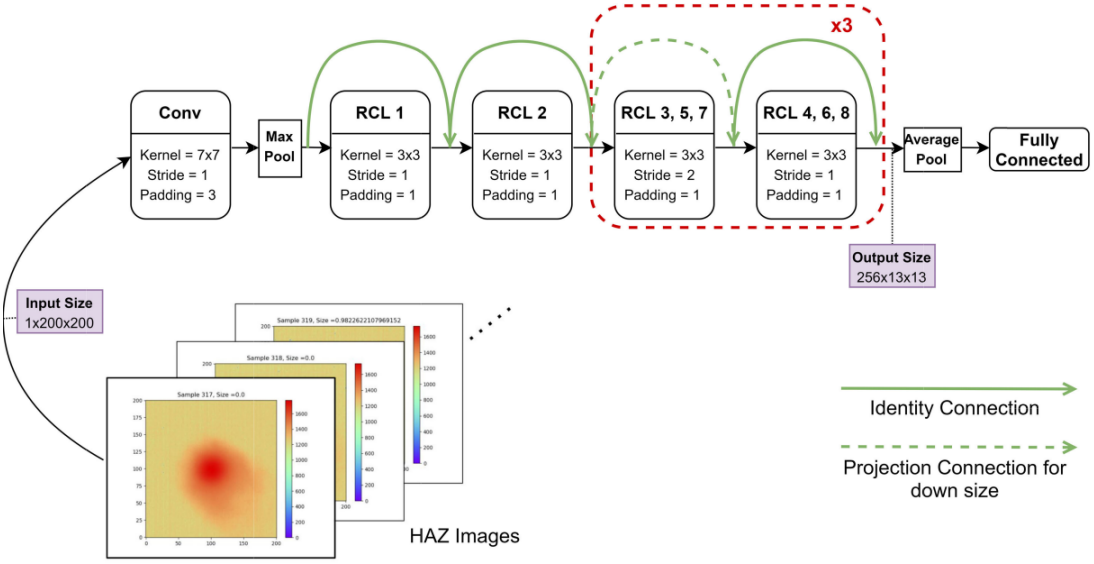


Figure 10: Res-RCNN architecture for porosity prediction [15].

This model was evaluated with various metrics like accuracy, sensitivity, precision and F1-score. It outperformed pure CNN model with 20% higher sensitivity and 10.5% higher F1-score, while having 43% fewer trainable parameters. Even though this model performed well, it has few shortcomings like the binary output (pore vs. normal), instead of outputting a porosity score, and the inability to distinguish between pore sizes and types. Authors suggest that "with the ability to predict porosity types and understand differences in operating parameters, DLAMs [Deep Learning based porosity prediction for Additive Manufacturing] would be ideal for a generalized feedback system" [15].

Zhang et al. [22] developed a CNN that predicted porosity attributes in real-time during laser additive manufacturing. This model was constructed closely after a simplified version of the well-known CNN AlexNet. To train the model, melt pool images corresponding to a position along laser scan direction was given a class label obtained from porosity inspection. A total of 2842 samples were generated and split into training and validation sets. The classification labels indicated the presence of porosity in a given region: true = pore, false = no pore. A pore size threshold was obtained using 10-fold cross-validation, and 10 μm was selected as the optimal value. Various convolutional layers were experimented with via cross validation, where the best classification accuracy was achieved with 7 layers at 90.39%.

In addition to detection of the presence of individual pores during deposition, the model also generated values for volume porosity, or the ratio of porosity volume to deposition volume. This calculation was achieved by dividing the sum of pore size at each longitudinal position on the track by the depth of the track. Figure 11. gives the results of the CNN's predictions of local volume porosity as a function of longitudinal distance.

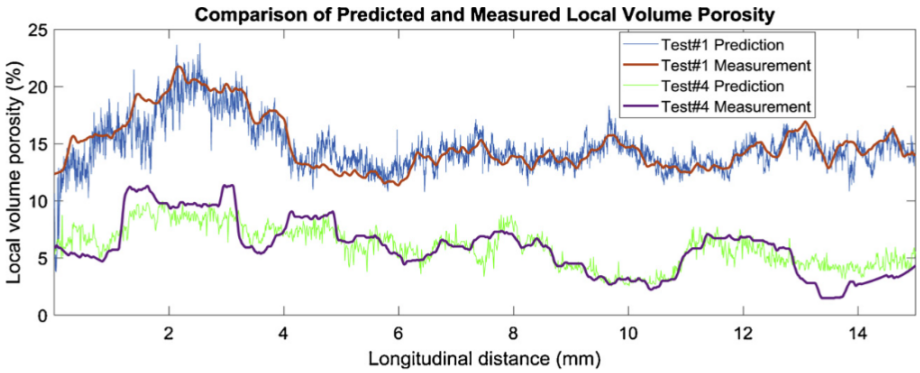


Figure 11: CNN predictions for local volume porosity [22].

Literature has shown that various types of CNNs are effective in identifying different defects that commonly occur in AM parts. Even though data collection can be a costly process in AM so collecting enough data for deep learning models can be difficult, transfer learning can be used to set-up an already trained model. Then such model can be fine-tuned with collected data to make accurate predictions. However, there remains work to be done to develop a robust pipeline that can handle defect segmentation, defect classification, and porosity percentage prediction.

3 Methodology

3.1 Agile Development Methodology

We decided to follow the *Agile Software Engineering Methodology* based on iterative and incremental software development. This methodology follows twelve principles, outlined in the "Manifesto for Agile Software Development", which include delivering working software frequently and paying attention to good design and technical excellence [23]. To follow Agile methodology, we decided to operate in the form of *sprints*, each with a clear implementation goal. At the end of each sprint, the code produced is evaluated and ready to use. However, machine learning development is unique from other types of software development, as training and tuning the model on new data do not always lead to a performance improvement. Therefore, our sprints also implement machine learning workflow (Figure 12.), which includes model evaluation even after deployment [24].

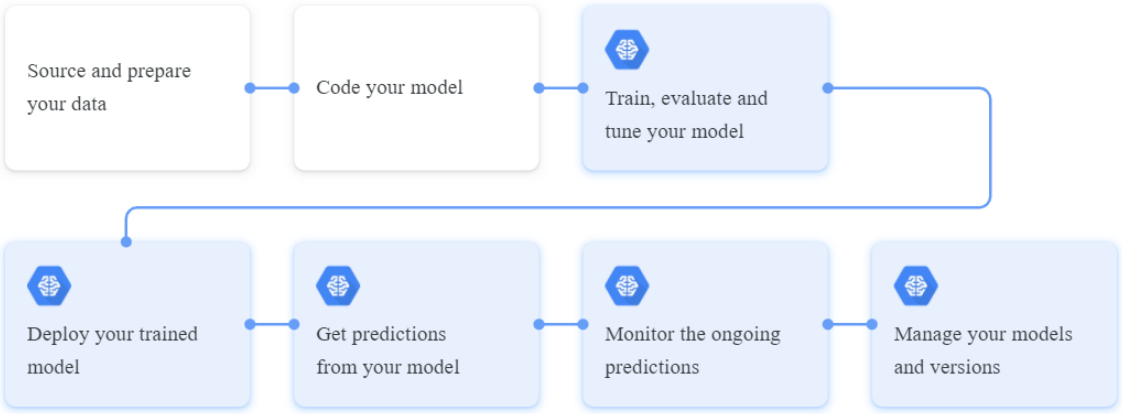


Figure 12: Machine learning workflow [24].

The Gantt chart in Figure 13. shows the rough timeline of the project, which was divided into five sprints, each with distinct stages of implementation, evaluation, and deployment.

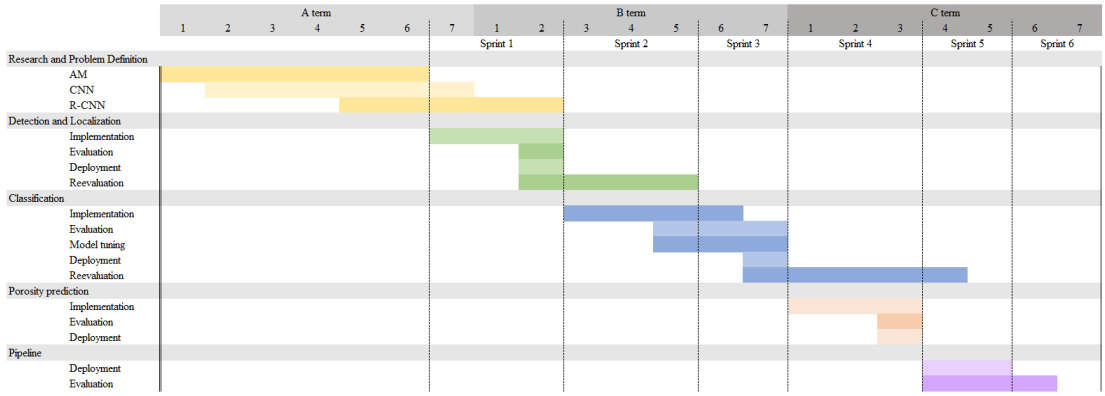


Figure 13: Project timeline.

This project can be divided into three separate components, each with its own machine learning workflow timeline, that can be combined into a pipeline:

1. Data preprocessing
2. Defect detection and localization

3. Defect classification

3.2 Technologies used

Our programming language of choice for this project is Python, which is a general-purpose programming language. This allows us to write code to handle all the tasks necessary for the project, from data cleaning through running machine learning models to creating a user interface in a single programming language. With a broad spectrum of ML libraries, Python seemed like an obvious choice for this project.

Python libraries used:

1. **TensorFlow** is a Python-based, open-source machine learning platform that provides an ecosystem of tools for easy development and deployment of ML applications [25].
2. **Keras** is a deep learning API written in Python, running on top of the machine learning platform TensorFlow [26].
3. **Tensorboard** is TensorFlow's visualization kit that allows for visualising the performance of the model [25].
4. **Matplotlib** is a library used to create data visualizations [27].
5. **Pandas** is a library that allows for data manipulation and analysis [28].
6. **Shapely** is a BSD-licensed Python package for manipulation and analysis of planar geometric objects [29].

Other tools:

1. **Slurm** is a workload manager for Linux used to allocate resources and manage jobs in HPC (high performance computing) applications [30].
2. **JupyterLab** is a web-based interactive development environment. For the ease of use, we decided to write our code with Jupyter notebooks [31].

3.3 Dataset

3.3.1 Dataset aquisition

We obtained our data set from the team working on the materials science side of this project. There were 36 samples collected in total, each created with different processing parameters, like laser power, scan speed, layer thickness and hatch spacing (Figure 14). For each of the 36 defected samples containing significant amounts of porosity, about 100 bright field images taken at 200x magnification on an optical microscope were be taken. Before the images were taken, samples needed to be cut, grinded and polished. Each individual image constituted a subsection of the larger sample, and measured 1280 x 1024 pixels.

Sample Label	Laser Power (W)	Scan Speed (mm/s)	Layer Thickness (mm)	Hatch Spacing (mm)
G0	100	500	0.03	0.065
G3	100	1400	0.03	0.025
G7	150	800	0.03	0.098
G8	150	1100	0.03	0.088
G9	150	1400	0.03	0.078
H0	150	1700	0.03	0.071
H3	200	800	0.03	0.127
H4	200	1100	0.03	0.115
H5	200	1400	0.03	0.105
H6	200	1700	0.03	0.097
H7	200	2000	0.03	0.089
H8	250	500	0.03	0.212
J0	250	1100	0.03	0.135
J1	250	1400	0.03	0.124
J3	200	2000	0.03	0.107
J4	300	500	0.03	0.275
J5	300	800	0.03	0.206
J7	300	1400	0.03	0.141
J8	300	1700	0.03	0.131
J9	300	2000	0.03	0.122
K0	370	500	0.03	0.338
K1	370	800	0.03	0.261
K4	370	1700	0.03	0.149
K5	370	2000	0.03	0.139
Q0	100	500	0.03	0.049
Q3	250	1100	0.03	0.103
Q4	370	500	0.03	0.259
Q5	370	2000	0.03	0.107
Q6	100	500	0.03	0.057
Q8	100	2000	0.03	0.006
Q9	250	1100	0.03	0.119
R0	370	500	0.03	0.298
R2	100	500	0.03	0.072
R5	250	1100	0.03	0.151
R6	370	500	0.03	0.378
R7	370	2000	0.03	0.156

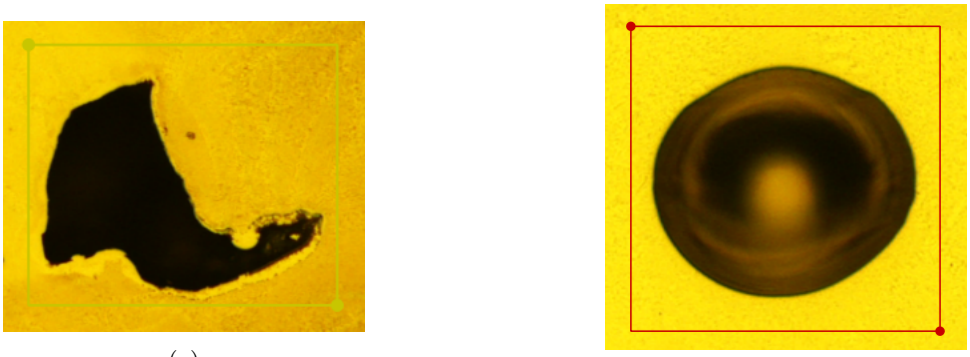
Figure 14: Processing parameters for each sample

Sample Label	Perceived porosity level	Most prominent defect type
G0	High	Lack of fusion
G3	Extreme	Lack of fusion
G7	Medium	Lack of fusion
G8	Medium	Lack of fusion
G9	High	Lack of fusion
H0	Medium	Lack of fusion
H3	Low	Lack of fusion
H4	Medium	Lack of fusion
H5	Medium	Lack of fusion
H6	Low	Lack of fusion
H7	Medium	Lack of fusion
H8	Low	Keyhole
J0	Low	Lack of fusion
J1	Low	Lack of fusion
J3	Low	Lack of fusion
J4	Medium	Lack of fusion,Keyhole
J5	Low	Keyhole
J7	Low	None
J8	Low	None
J9	Low	None
K0	Medium	Keyhole
K1	Low	Keyhole
K4	Low	None
K5	Low	Keyhole
Q0	High	Lack of fusion
Q3	Low	Lack of fusion
Q4	Medium	Keyhole
Q5	Low	None
Q6	High	Lack of fusion
Q8	Extreme	Lack of fusion
Q9	Low	None
R0	Medium	Keyhole
R2	High	Lack of fusion
R5	Low	None
R6	Medium	Keyhole
R7	Low	None

Figure 15: Summary of collected samples

3.3.2 Dataset annotation

LabelMe [32] software was used for data annotation. Each of the defects is enclosed in a bounding-box and labeled as a member of one of the three classes: lack of fusion porosity, keyhole porosity and other. These annotations were stored in a JSON format. Figure 21 gives examples of labeled defect belonging to each of the two classes, and Figure 17 gives a sample fully labeled in LabelMe.



(a) (b)

Figure 16: Labeled instances of (a) lack of fusion porosity and (b) keyhole porosity.

It can be noticed that keyhole porosity instances tend to have a regular, round shape while lack of fusion porosity instances have a much less regular and more rigged geometry.

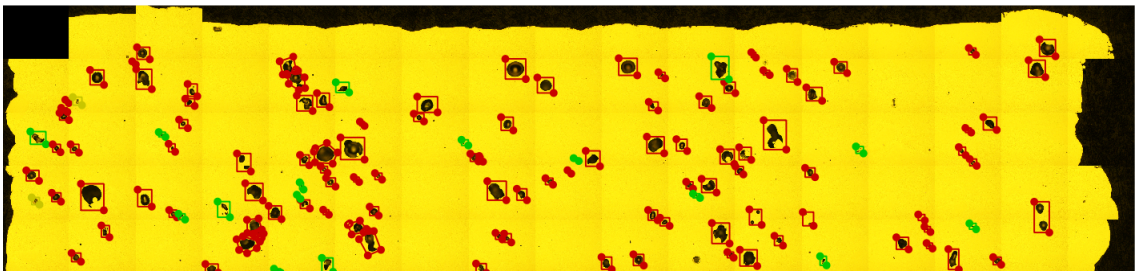


Figure 17: Fully labeled sample in LabelMe. Red are instances of keyhole porosity, and green annotations are instances of lack of fusion porosity.

Each image has a separate respective JSON file containing its annotations. The format of the annotation is shown in Figure 18. All instances (defects) are stored in "shapes" field as an array. Each instance has a assigned label, a shape type of the annotation (either a "rectangle" or "polygon"), and a set of points that define the shape. LabelMe automatically generates "group id" and "flags" fields and they are not used.

```
"shapes": [
  {
    "label": "keyhole porosity",
    "points": [
      [
        732.1739130434783,
        670.9130434782609
      ],
      [
        818.0434782608695,
        756.7826086956521
      ]
    ],
    "group_id": null,
    "shape_type": "rectangle",
    "flags": {}
  },
  {
    "label": "lack of fusion porosity",
    "points": [
      [
        683.2608695652174,
        140.47826086956522
      ],
      [
        721.3043478260869,
        173.08695652173913
      ]
    ],
    "group_id": null,
    "shape_type": "rectangle",
    "flags": {}
  }
],
```


Figure 18: Example JSON file containing label coordinates.

3.4 Computing Cluster

Due to the high computational costs of training a neural network on a large set of high-dimensional image data, we utilized WPI's Turing HPC (High Performance Computing) clusters to train our models. The Turing cluster comes loaded with a large number of libraries for machine learning and data manipulation applications upon login in via SSH. Job submissions are made via Slurm, Turing's workload manager, where a number of parameters including number of nodes, CPU cores, GPU cores, and a time limit are provided, in addition to a program to be run on the hardware.

The Turing cluster is composed of 56 computing nodes, with a variety of hardware configurations. Training was performed on the compute-0-27 node, utilizing the NVIDIA P100 Accelerator due to its high computational power.

Nodes	#of CPUs	Memory	Constraints (-C)	GPUs
compute-0-[01-04]	20	128 GB	K40, E5-2680	2
compute-0-[05-24]	20	128 GB	K20, E5-2680	2
compute-0-[25-26]	40	256 GB	K80, E5-2698	4
compute-0-27	16	512 GB	P100, E5-2667	4
compute-0-28	16	512 GB	V100, E5-2667	4
compute-0-[29-38]	36	256 GB	E5-2695	none
compute-0-[39-46]	48	256 GB	8168	none
compute-0-[47-54]	40	192 GB	6148	none
compute-0-55	40	192 GB	6148,K20	2
compute-0-56	40	192 GB	6148,P100	2

3.5 Performance measures

Since each of the three components of the pipeline is a different type of machine learning task, each of them has a separate set of performance metrics.

3.5.1 Performance Evaluation of pore localization task

Pore localization can be evaluated by how well the predicted mask and bounding box areas match the placement of the ground-truth ones.

Intersection-over-Union (IoU) It is the most popular metric used for object detection and segmentation [33]. It compares the members of two sets to measure which members are similar and which are different. In the context of image segmentation, this measures how good the overlap between A and B is, where A is the ground-truth bounding-box and B is the prediction bounding-box.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6}$$

3.5.2 Performance Evaluation of pore classification task

Pore classification can be evaluated by comparing the ground-truth label to the one generated by the model. No one metric can capture all of the aspects of the classification model. For this reason, we are using multiple metrics that complement each other.

Confusion matrix It is a summary of all of the predictions made by the classification model that compares the predictions to the expected output [14]. These results can be visualized on the grid. When a positive instance is correctly classified as positive it is considered a true positive (TP) and when an instance is correctly classified as a negative instance it is a true negative (TN). False negative (FN) are positive instances that are incorrectly classified as negative class and false positive (FP) are negative instances that were incorrectly labeled as positive. This extends to multi-class classification.

Precision The ratio of correctly classified instances to all instances classified as positive.

$$P = \frac{TP}{TP + FP} \tag{7}$$

Recall The ratio of positive instances that are correctly classified.

$$R = \frac{TP}{TP + FN} \tag{8}$$

Mean Average Precision (mAP) Commonly used metric for evaluating the performance of object detectors. Gives a generalized score of classification performance between 0 and 1 [34]. It is calculated from the mean of average precision over all classes, which is equal to the area under the precision-recall curve shown in Figure 19. below.

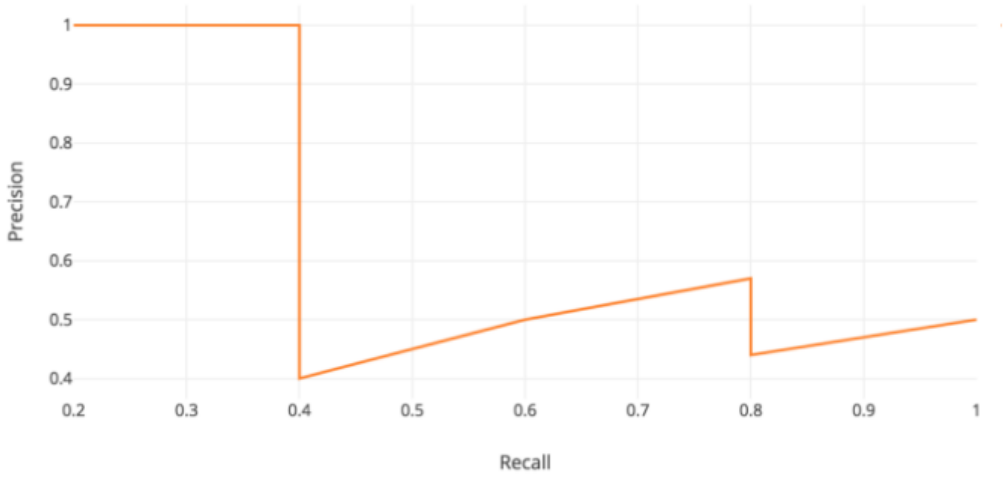


Figure 19: Precision-recall curve [35].

F1-measure It is the harmonic mean of the precision and recall [14].

$$R = \frac{2 \times P \times R}{P + R} \tag{9}$$

3.5.3 Mask R-CNN Model Evaluation

Since Mask R-CNN localizes and categorizes defects simultaneously, a multi-task loss is used to defined its performance during training [20]:

$$L = L_{cls} + L_{box} + L_{mask} \tag{10}$$

The classification loss L_{cls} is the negative average of the log of corrected predicted probabilities for each instance and the bounding-box loss L_{box} is the regression loss between the four verices of the predicted and ground-truth bounding boxes. Both L_{cls} and L_{box} are the same as the losses for Faster R-CNN model [19]. The mask loss L_{mask} is the average cross entropy loss between the ground-truth mask and the predicted mask. The smaller the total loss, the better the model is fitted to the data.

Overall, all these metrics give us a good understanding of the overall performance of the algorithm, as well as, of the performance of each of the components.

4 Implementation

The implementation of a program designed to train on a custom data set, generate classifications, and predict porosity on new samples required a multi-faceted approach to implementation. We started by pre-processing, cleaning the data and setting up the scripts to augment it.

4.1 Dataset

4.1.1 Data cleaning

Data cleaning was performed to prepare the annotated images into a single data set for the model to utilize. Since pores less than 20 microns in size do not contribute significantly to the fatigue life or mechanical properties of the alloy, a threshold of 800 pixels, the equivalent of 1744 square microns, was applied to to remove any annotations with a total area less than the threshold. Additionally, images that contained no annotated instances of porosity were omitted from training, as Mask R-CNN requires minimum one annotation per image.

Due to the dataset being labeled by multiple people, we have also found inconsistencies in the naming scheme of classes. We ensured to correct the class names so the final set of labels only included 'other', 'keyhole porosity' and 'lack of fusion porosity'.

Some samples contained very large instances of lack of fusion porosity than spanned multiple individual images. To ensure that the geometry of every defect was persevered for the model, these sets were split along custom boundaries. These regions were defined via bounding boxes and polygons that were added to existing annotated sets. These regions were then used to split images along these boundaries, and create a new image and JSON file. Figure 20 shows an example of these regions on sample Q0. The code used to convert these regions into separate images and transfer the labels can be found in the region_extraction method of utils/dataset.py, found in the Github repository [36].

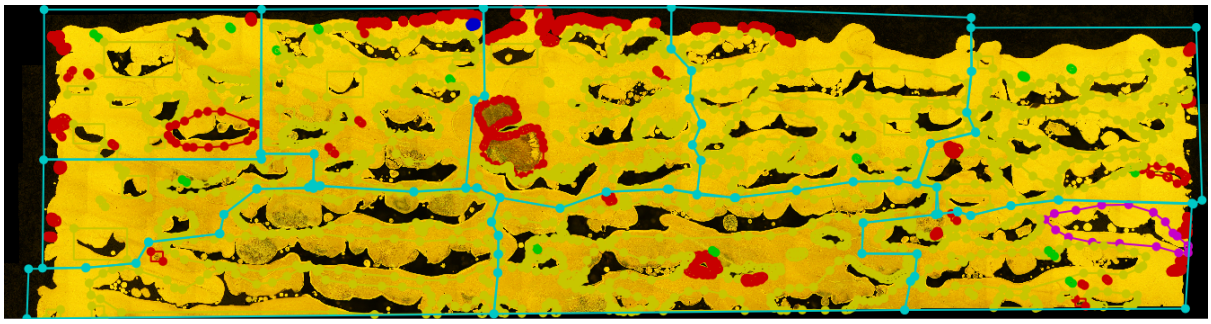


Figure 20: Sample Q0 split into regions, shown in blue

After splitting some of the images into regions some of the images ended up quite large, the the largest being XxX pixels. To ensure that the model does not exceed available memory we decided to downsize all images by 50%. The scripts that we used to do so can be found in the utils folder of out Github repository [36].

After data clearing our final data set consisted of 798 images. All labeled instance from the training data set were pre-processed to generate a binary mask. These masks are used during training of the Mask R-CNN model to generate convolutional filters and learn regions, and to generate masks in conjunction with bounding boxes during prediction. Figure 21 gives a labeled defect instance on a part scan and its respective generated binary mask.

(a) (b)

Figure 21: (a) Instance of porosity from labeled sample (b) Generated binary mask from instance.

For some instances generating a well-fitted mask was much more of a challenge. Since images were taken with varied brightness, the color of the background was not the same for every image. To resolve this issue we took the average color of the background of each image to be the baseline instead of having a constant color for each of the image. The implementation of acquiring the average background color can be found in extract_mask in dataset.py in the GitHub repository [36]. The difference between the two approaches can be seen in Figure 22 below.

(a) (b) (c)

Figure 22: (a) Polygonal annotation with pure yellow background (b) Polygonal annotation with averaged color background c) Binary mask generated from annotation.

The data set was split into three sections: a training set, a validation set and a test set. The test set is a randomly chosen 5% of the data that is used for final model evaluation. The test set consists of 77 images. The images in that set are not used for training and optimizations. The rest of the data was split into a train and validation sets with 80/20 split ratio. The train set consists of 577 images and the validation set of 144 images.

4.1.2 Addressing class in-balance

Since our data set contained significantly more lack of fusion instances than keyhole instances the set had to be re-balanced. To do so we obtained the distribution of lack of fusion instances areas and split this class into three sub-classes: "small lack of fusion", "medium lack of fusion" and "large lack of fusion" based on their sizes. We assigned instances with area smaller than 2492.50 pixels to "small lack of fusion porosity", and the instances with area smaller than 14661.50 pixels to "medium lack of fusion porosity" class, all the other instances are assigned to "large lack of fusion porosity" class. Figure 23. gives the count of instances of lack of fusion porosity that fall under each of these thresholds.

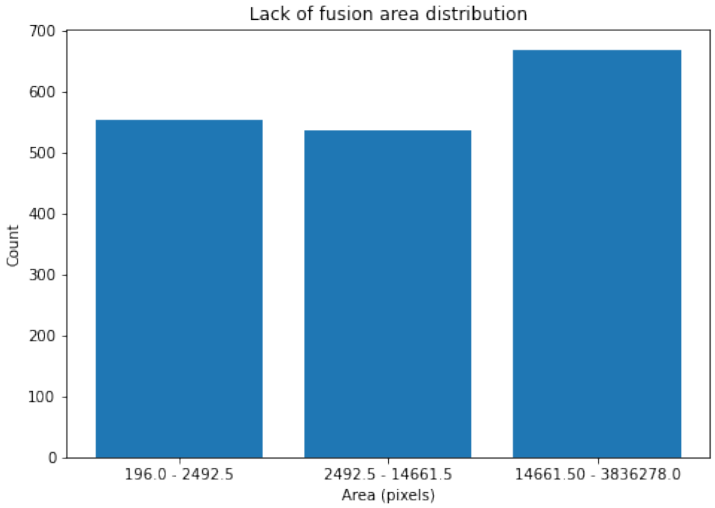


Figure 23: Lack of fusion porosity area distribution.

Figure 24. gives the total instances of each type of defect that occurred across all sets.

Small lack of fusion porosity	Medium lack of fusion porosity	Large lack of fusion porosity	Keyhole porosity
552	537	669	608

Figure 24: Total instances of each defect type across all samples.

4.2 Mask R-CNN

For the training and classification tasks of our model, Mask R-CNN was selected for its flexibility and the success of other models trained on data similar to our own. Matterport Inc's implementation of Mask R-CNN [1] was used as the base model for this project. This particular implementation differed in a few ways from the originally proposed Mask R-CNN model in literature; namely the omission of bounding box ground truth instances and support for input images of a variety of sizes. The underlying neural network architecture of the model was preserved for the purposes of our model, however a number of hyperparameters were adjusted over the course of development to optimize for the given data set.

Figure 25 below gives a schematic of the architecture of the final Mask R-CNN model for defect classification.

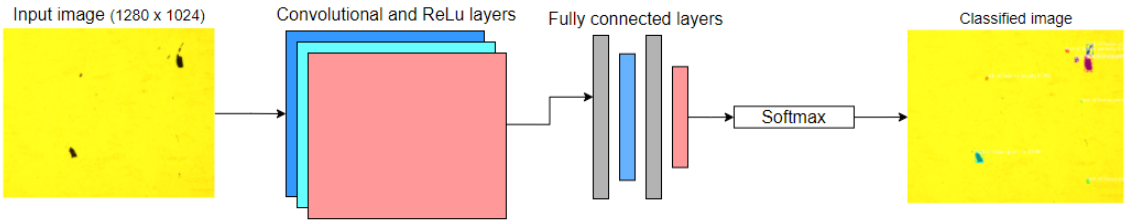


Figure 25: Mask R-CNN defect classification network architecture.

4.3 Transfer Learning

Since the amount of data available to us was relatively small we decided to take advantage of transfer learning, which is a technique where instead of starting with randomly initialized weights, the already trained model can be used as a starting point for the new model [14]. Transfer learning has been successfully used with Mask R-CNN for defect detection task by Attard et al. [37], Han et al. [21] and Xu et al. [38] . Before training on our data set, we loaded weights from architecture that was already trained to detect objects in images. Common Objects in Context (COCO) [39] is a large-scale object detection, segmentation, and captioning data set. Before training, we initialized our model with COCO weights so that is it already trained to defect various objects.

4.4 Ensured Longevity

To ensure that the project is sustainable we will provide a clear documentation outlining the functionality of the model. We will also use Jupyter notebooks for our project's source code, divided into code blocks. This not only makes the code easy to understand but also allows for the model to be easily altered and redeployed since the notebook can be uploaded to Google Colaboratory, commonly known as "Google Colab", and ran without the need for setting up any local environment.

5 Results

5.1 Dataset

For our final model we decided to create a balanced with only low and medium porosity samples by sub-sampling the entire available data set. We chose to include sets J0, J1, J3, J4, J4R, K0, K0R, K4, Q3,Q5, R0, R6, R7. Table 26 shows the number of labels for each defect type. This decision was made to reduce the training time and allow for more iterations of the model during the limited project time.

Lack of fusion porosity	Keyhole porosity	Total number of instances
709	640	1349

Figure 26: Total number of instances of each defect type across selected samples

Additionally, due to memory constraints and the chosen model implementation, all of the images had to be down-scaled by 50%. After down-scaling, the largest image in the data set measured 1452 x 1514 pixels. The script to downscale the images can be found in the Github repository [36] alongside with the script to transfer the labels to the smaller images.

5.2 Hyperparameter tuning

Hyperparameter values can significantly impact the performance of a neural network [40]. Research has shown that this is also true for Matterport's implementation of Mask R-CNN as shown by [37]. To optimize the hyperparamters we decided to conduct a grid search over some of the hyperparamters of Mask R-CNN. The chosen hyperparameters are: learning rate and steps per epoch. For other hyperparameters the default parameter value was used. All of these models were trained on 250 epochs. We tried 6 combinations of these hyperparameters:

Learning rate	Steps per epoch	mAP
0.01	100	0
0.001	100	0.202
0.0001	100	0.233
0.00001	100	0.254
0.000005	100	0.3525
0.00001	300	**0.3826**

Figure 27: The mean Average Precision of the model based on different hyperparameter values.

5.3 Final results

The three main goals of this project were: 1) to obtain, clean and prepare the data set to be used in a deep learning model, 2) ensure proper localization of defects and finally 3) train a model that would successfully classify pores in the image.

5.4 Defect localization

Our model was effective in identifying which regions on the samples to denote a defect. Figure 28. below gives an example of effective region identification of defects on a region of a sample.

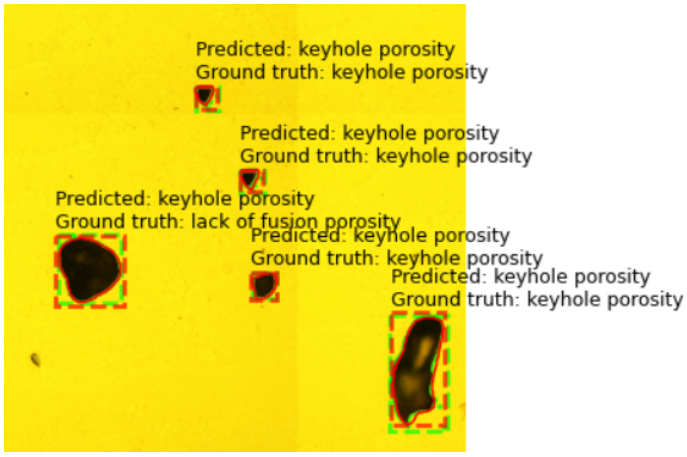


Figure 28: Example classifications on region of sample: red denotes predicted area, green denotes ground truth. All regions shown were accurately generated by the model.

Figures 29. and 30. give the loss as a function of epochs of bounding box generation on the training and validation sets, respectively, and Figures 31. and 32. give loss for mask generation for the training and validation sets.

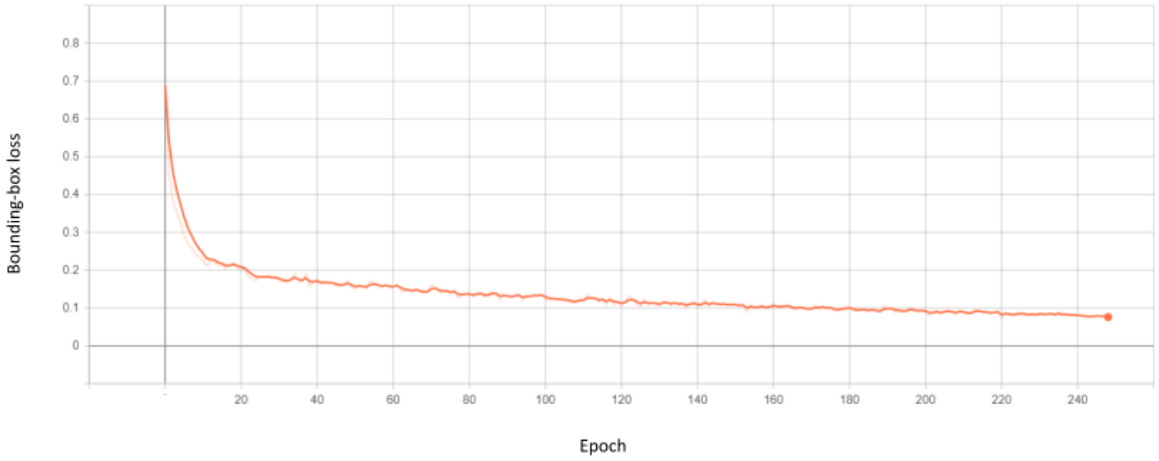


Figure 29: Bounding-box loss in each epoch on the training set

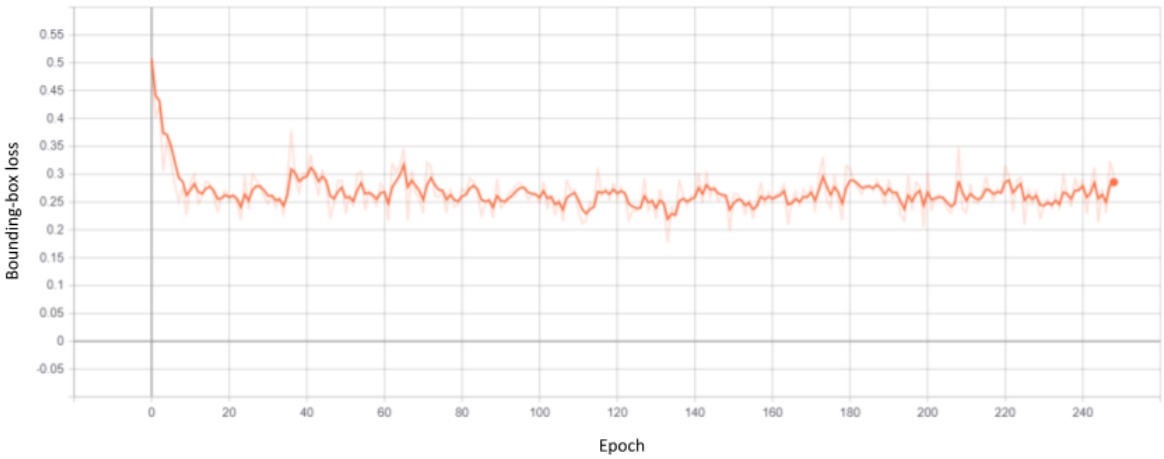


Figure 30: Bounding-box loss in each epoch on the validation set

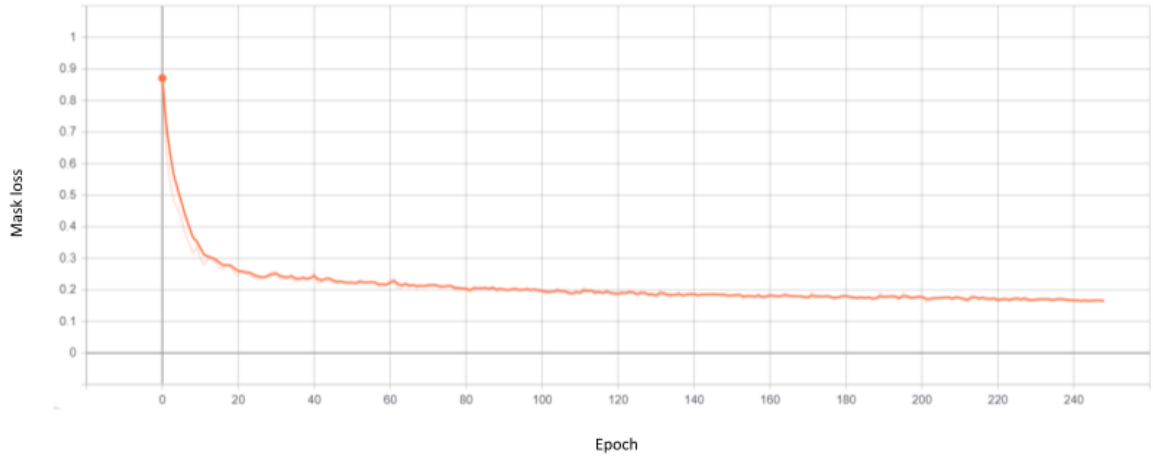


Figure 31: Mask loss in each epoch on the training set

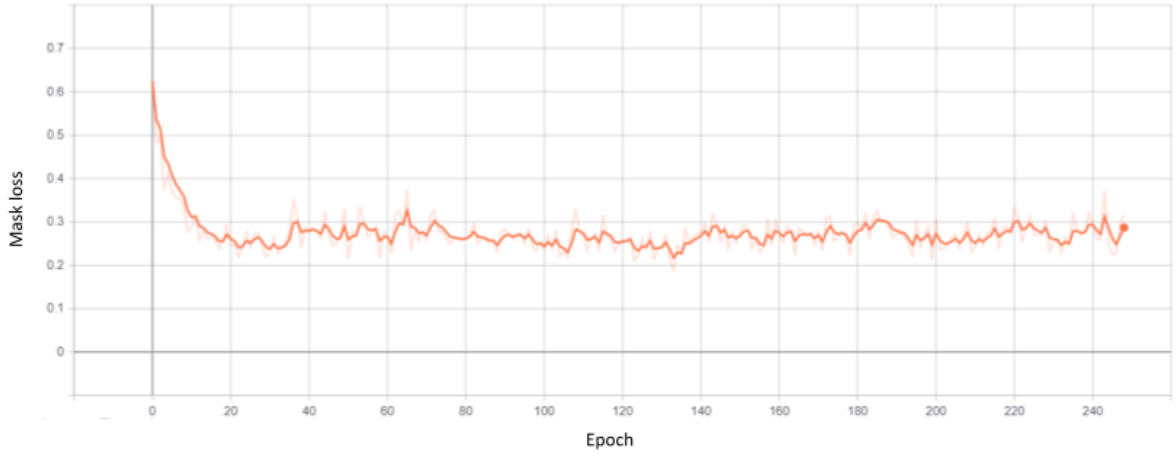


Figure 32: Mask loss in each epoch on the validation set

5.5 Defect classification

A final mean average precision score of 0.3826 was achieved utilizing 14 different samples for training. Figures 33 and 34 give classification loss as a function of epochs on the training and validation sets, respectively.

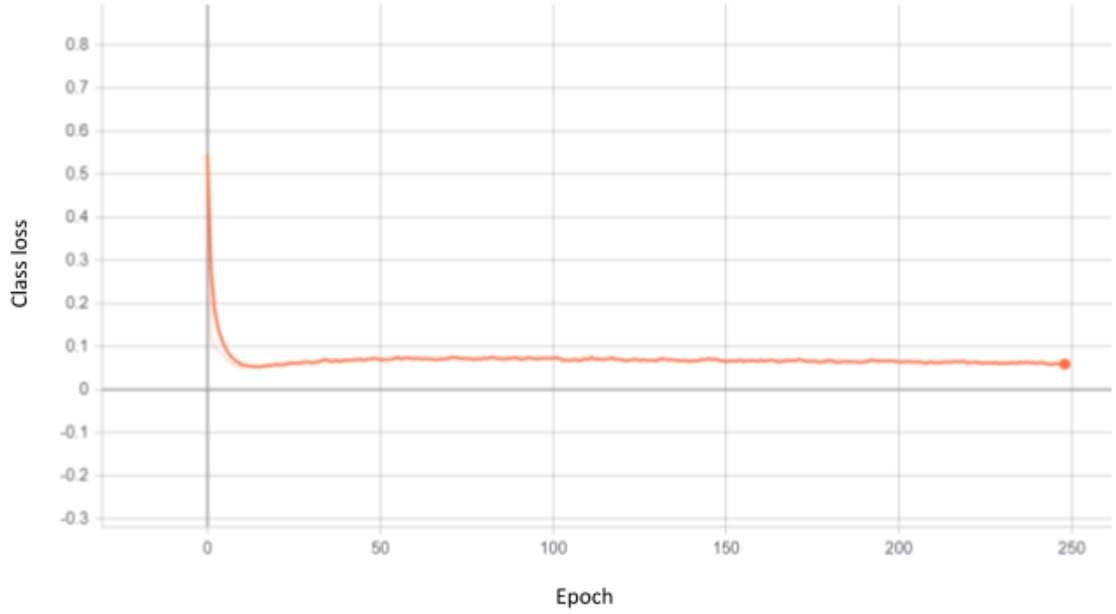


Figure 33: Classification loss in each epoch on the training set

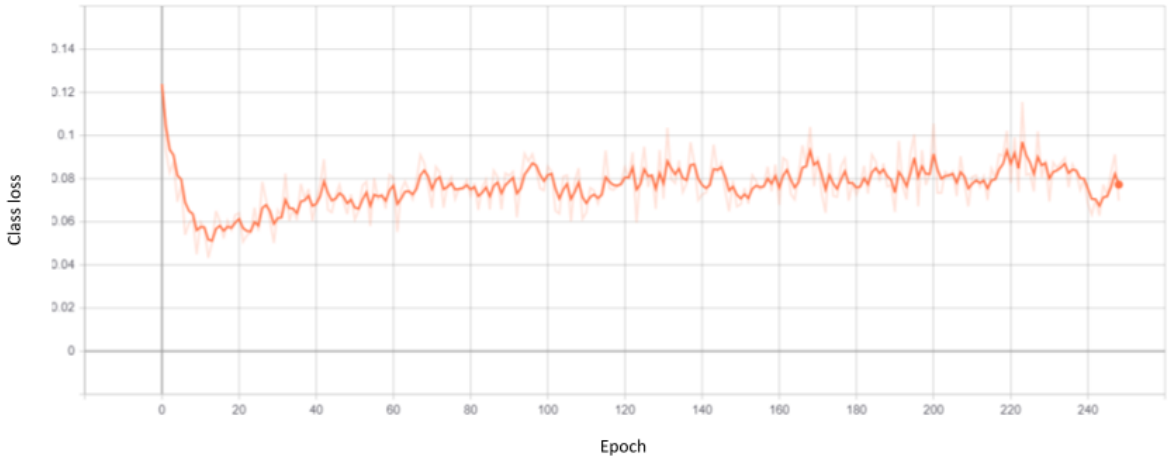


Figure 34: Classification loss in each epoch on the validation set

		Actual		
		Lack of fusion porosity	*Keyhole porosity*	Total
Predicted	*Lack of fusion porosity*	16 11.03%	6 4.13%	**22** **72.73%** **37.50%**
	Keyhole porosity	40 27.58%	83 57.24%	**123** **67.48%** **32.52%**
	Total	**56** **28.57%** **71.43%**	**89** **93.26%** **6.74%**	**143** **69.23%** **32.17%**

Figure 35: Final confusion matrix.

In the confusion matrix displayed above (Figure 35.) the recall and precision scores are shown for each class. The green fields in the last column show the recall score and the green fields on the bottom row show the precision values. The recall values for both lack of fusion and keyhole porosity are quite large, 72.73% and 67.48% respectively. This indicates that for both classes, the number of instances predicted correctly is larger than predicted incorrectly. However, when it comes to precision score, the keyhole porosity precision score is almost perfect (93.16%) which means very few instances of keyhole porosity are classified as lack of fusion porosity. In case of lack of fusion the precision score is very poor - 28.57%. The model more often predicts that a lack of fusion instance is a keyhole porosity than it does correctly classify it.

The challenges associated with the model's performance are indicative of larger problems associated with the data set: namely the wide in-class variance of lack of fusion porosity, as well as similarities in geometries between sets. Figure 36. below demonstrates this very issue: lack of fusion ground truth annotations being classified as keyhole porosity is the biggest detractor to overall model performance.

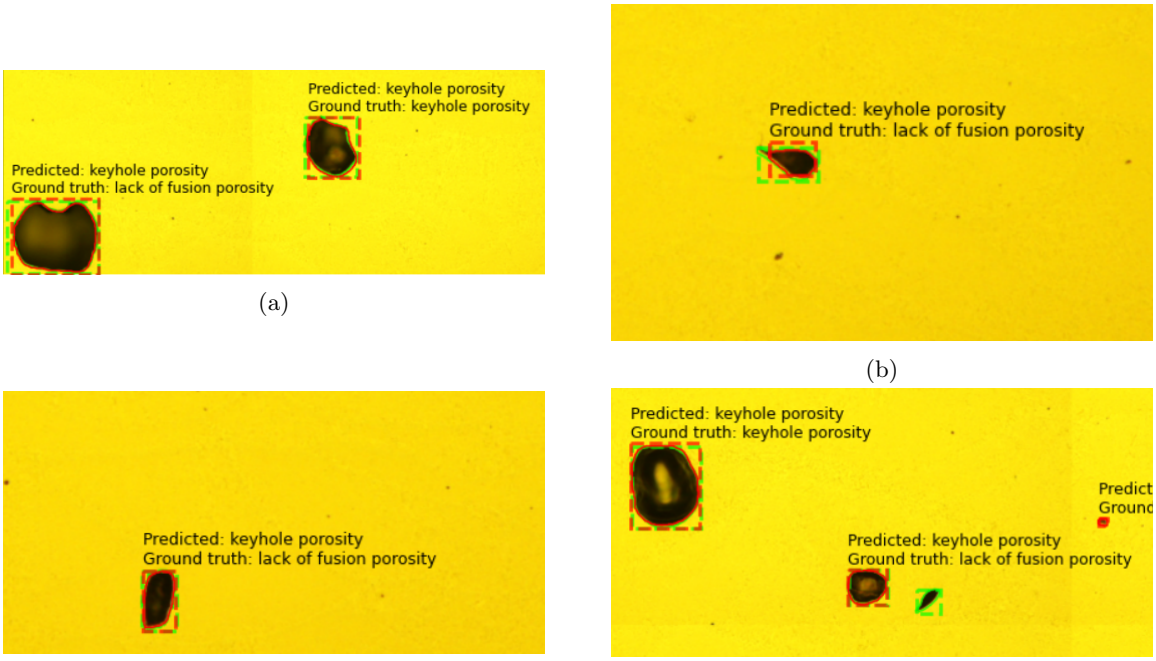


Figure 36: Instances of lack of fusion porosity with circular geometry misclassified as keyhole porosity.

Overall our model has achieved a 69% overall accuracy score. The F1-scores for lack of fusion porosity and keyhole porosity are 0.41 and 0.78 respectively.

$$overall\_accuracy = 69.23\% \tag{11}$$

$$F1_{lof} = 0.4105 \tag{12}$$

$$F1_{keyhole} = 0.7833 \tag{13}$$

In both loss graphs we can see that even though training loss kept steadily falling until it reached total loss of 0.5 at the end of epoch 250, the validation loss stopped decreasing after epoch 20 and oscillated between 1 and 1.5.

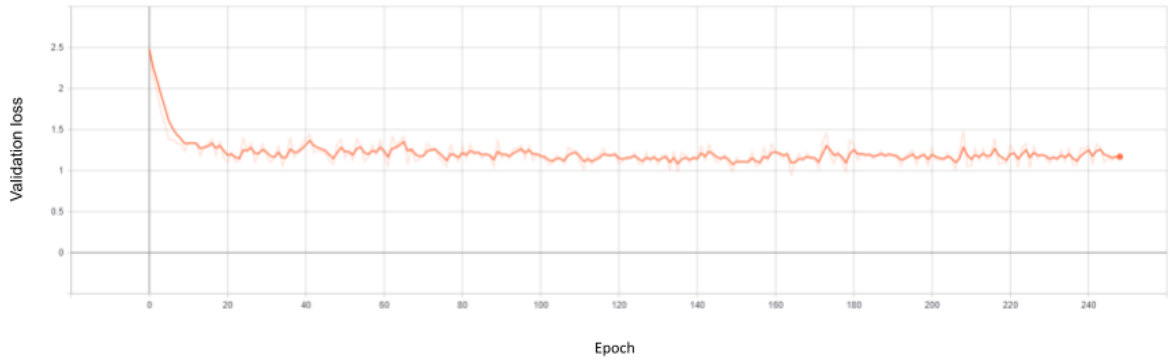


Figure 37: Loss in each epoch on the validation set

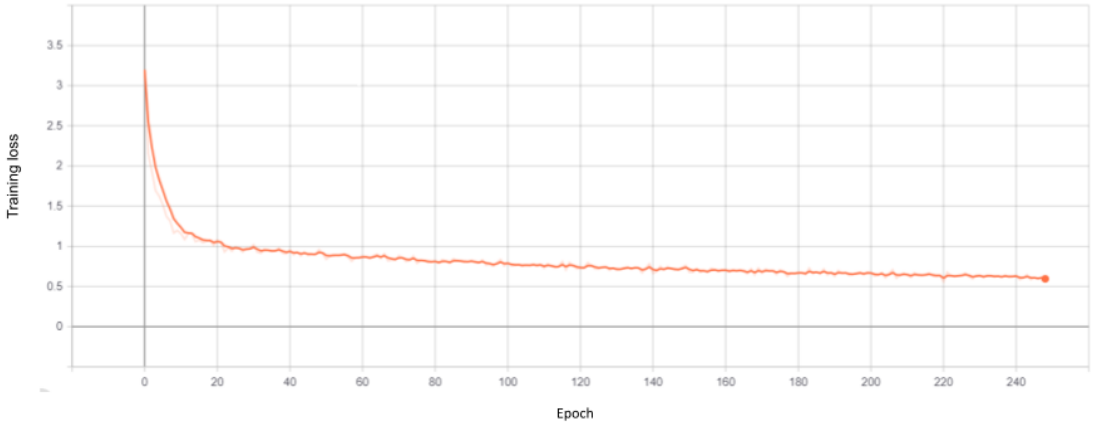


Figure 38: Loss in each epoch on the training set

6 Conclusions and Future Work

6.1 Project Outcomes

One of the main outcomes of this project is the clean and labeled data set containing broad variety of sample morphology ranging from fully dense samples to samples with extreme porosity. This outcome was fully realized through the collaborative effort of students preparing and imaging the samples, in addition to our exhaustive data cleaning and image stitching. The data set can be utilized in different types of image classification models in the same problem domain. The developed Mask R-CNN model achieved the overall classification accuracy score of 69.23%, and performs particularly well on keyhole porosity with precision score for that class being over 90%.

6.2 Future Work

Since using machine learning and deep learning is still quite new in the area of additive manufacturing, there are a lot of potential directions in which this project can be taken.

6.2.1 Data Augmentation

Since our data set contains only 798 images, our model is more prone to over-fitting. Small data sets often do not contain enough variability and this causes the models trained with them to not be able to generalize and preform poorly on the test sets [41]. Data augmentation is a technique that allows to increase the amount of training data by applying various transformations on the training data. The common methods for data set augmentation include geometric transformations, like image reflection, rotation, cropping, noise injection, scaling and translation photometric transformations, which includes color space transformations [42]. The following four augmentation methods could be applied to the original data set:

1. Horizontal axis flipping
2. Vertical axis flipping
3. Gaussian transform
4. Impulse (salt-and-pepper) noise

The Python script utils/data_augmentation.py that could perform such augmentations can be found in our Github directory [36]. Even though we were not able to extend our training set through augmentation due to time and resource constraints, we believe that this should be one of the first steps taken ion effort to improve the performance of the model.

6.2.2 High Porosity Classification

Our attempts focused on distinguishing keyhole porosity from small and medium lack of fusion porosity. This decision was made as high and extreme lack of fusion pores have incredibly hard geometries and they are large, resulting in much larger images that encompass the entire pore. However, since our data set includes these types of pore, a new model could be developed to include them in the classification.

6.2.3 Porosity Prediction

Since our data is annotated and mask extraction provides the shape and size of the pore, it could be used to predict the porosity of the entire sample.

6.2.4 Manufacturing parameters optimization

Another possible avenue to expand the project is to create the model that could predict the optimal processing parameters for the metal. This would allow the engineers to decrease the number of trials needed and make the entire process more cost-efficient.

6.2.5 User Interface

Our solution lacks in the usability aspect as it lack a visual interface that could be used by the lab engineers. Such interface could allow the user to upload a batch of data and run our model on it. Predictions then could be evaluated and if necessary changed by the user.

6.3 Conclusions

Even though our model's performance is significantly worse than the performance of the models that can be found in literature, we believe that this project laid the foundation for several projects that connect the worlds of Materials Science and Computer Science. For this reason, this paper outlines not only our successes but also failures and various obstacles that had to be overcome. We also bring attention to and provide solutions to some common image recognition challenges like quality mask extraction for a bounding-box and class in-balance. We hope that this paper will be a valuable resource for anybody working on image recognition tools for the field of Material Science.

References

[1] W. Abdulla, Mask r-cnn for object detection and instance segmentation on keras and tensorflow, https://github.com/matterport/Mask_RCNN, 2017.

[2] "Additive manufacturing — general principles — terminology," International Organization for Standardization, Standard, Dec. 2015.

[3] Hubs, Additive manufacturing trend report 2021, https://f.hubspotusercontent10.net/hubfs/4075618/Additive%20manufacturing%20trend%20report%202021.pdf.

[4] R. Liu, Z. Wang, T. Sparks, F. Liou, and J. Newkirk, "13 - aerospace applications of laser additive manufacturing," in Laser Additive Manufacturing, ser. Woodhead Publishing Series in Electronic and Optical Materials, M. Brandt, Ed., Woodhead Publishing, 2017, pp. 351–371, ISBN: 978-0-08-100433-3. DOI: https://doi.org/10.1016/B978-0-08-100433-3.00013-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780081004333000130.

[5] M. Salmi, "Additive manufacturing processes in medical applications," eng, Materials, vol. 14, no. 1, pp. 191–, 2021, ISSN: 1996-1944.

[6] Data pipeline, Feb. 2022. [Online]. Available: https://hazelcast.com/glossary/data-pipeline/.

[7] D. Systems, 3d printing - additive. https://make.3dexperience.3ds.com/processes/directed-energy-deposition.

[8] H. Kotadia, G. Gibbons, A. Das, and P. Howes, "A review of laser powder bed fusion additive manufacturing of aluminium alloys: Microstructure and properties," eng, Additive manufacturing, vol. 46, pp. 102 155–, 2021, ISSN: 2214-8604.

[9] H. M. Khan, Y. Karabulut, O. Kitay, Y. Kaynak, and I. S. Jawahir, "Influence of the post-processing operations on surface integrity of metal components produced by laser powder bed fusion additive manufacturing: A review," eng, Machining science and technology, vol. 25, no. 1, pp. 118–176, 2020, ISSN: 1091-0344.

[10] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach (4th Edition), 4th ed. Pearson, 2020, ISBN: 9780134610993.

[11] A. L. Samuel, "Some studies in machine learning using the game of checkers," IBM Journal of Research and Development, vol. 3, no. 3, pp. 210–229, 1959. DOI: 10.1147/rd.33.0210.

[12] I. C. Education, Deep neural network. neural networks. https://www.ibm.com/cloud/learn/neural-networks.

[13] R. Szeliski, Computer Vision Algorithms and Applications, eng, 1st ed. 2011., ser. Texts in Computer Science. London: Springer London, 2011, ISBN: 1-84882-935-3.

[14] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build eng. Sebastopol: O'Reilly Media, Incorporated, 2019, ISBN: 9781492032649.

[15] S. Ho et al., "Dlam: Deep learning based real-time porosity prediction for additive manufacturing using thermal images of the melt pool," IEEE Access, vol. 9, pp. 115 100–115 114, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3105362.

[16] R. Prabhu, Neural network with many convolutional layers. understanding of convolutional neural network (cnn) – deep learn https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, 2018.

[17] W. Cui, Y. Zhang, X. Zhang, L. Li, and F. Liou, "Metal additive manufacturing parts inspection using convolutional neural network," eng, Applied sciences, vol. 10, no. 2, pp. 545–, 2020, ISSN: 2076-3417.

[18] X. Tao, D. Zhang, W. Ma, X. Liu, and D. Xu, "Automatic metallic surface defect detection and recognition with convolutional neural networks," Applied Sciences, vol. 8, no. 9, 2018, ISSN: 2076-3417. DOI: 10.3390/app8091575. [Online]. Available: https://www.mdpi.com/2076-3417/8/9/1575.

[19] R. Girshick, "Fast r-cnn," eng, in 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, 2015, pp. 1440–1448, ISBN: 1467383910.

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," pp. 2980–2988, 2017. DOI: 10.1109/ICCV.2017.322.

[21] F. Han, S. Liu, S. Liu, J. Zou, Y. Ai, and C. Xu, "Defect detection: Defect classification and localization for additive manufacturing using deep learning method," pp. 1–4, Aug. 2020. DOI: 10.1109/ICEPT50128.2020.9202566.

[22] B. Zhang, S. Liu, and Y. C. Shin, "In-process monitoring of porosity during laser additive manufacturing process," Additive Manufacturing, vol. 28, pp. 497–505, 2019, ISSN: 2214-8604. DOI: https://doi.org/10.1016/j.addma.2019.05.030. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214860419303653.

[23] K. Beck et al., Manifesto for agile software development, 2001. [Online]. Available: http://www.agilemanifesto.org/.

[24] Google, Machine learning workflow — ai platform, https://cloud.google.com/ai-platform/docs/ml-solutions-overview.

[25] Martín Abadi et al., TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[26] F. Chollet et al., Keras, https://github.com/fchollet/keras, 2015.

[27] J. D. Hunter, "Matplotlib: A 2d graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[28] T. pandas development team, Pandas-dev/pandas: Pandas, version latest, Feb. 2020. DOI: 10.5281/zenodo.3509134. [Online]. Available: https://doi.org/10.5281/zenodo.3509134.

[29] S. Gillies et al., Shapely: Manipulation and analysis of geometric objects, 2007–. [Online]. Available: https://github.com/shapely/shapely.

[30] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in Job Scheduling Strategies for Parallel Processing, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60, ISBN: 978-3-540-39727-4.

[31] Project jupyter. [Online]. Available: https://jupyter.org/.

[32] K. Wada, Labelme: Image polygonal annotation with python, https://github.com/wkentaro/labelme, 2018.

[33] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union," Jun. 2019.

[34] E. Zhang and Y. Zhang, "Average precision," in Encyclopedia of Database Systems, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 192–193, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_482. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_482.

[35] J. Hui, Map (mean average precision) for object detection. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, 2018.

[36] C. Broderick and A. Staszewska, Automatizing inspection and defect detection for metal additive manufacturing using deep learning techniques. [Online]. Available: https://github.com/cabroderick/ML-AM-MQP.

[37] L. Attard, C. J. Debono, G. Valentino, M. Di Castro, A. Masi, and L. Scibile, "Automatic crack detection using mask r-cnn," pp. 152–157, 2019. DOI: 10.1109/ISPA.2019.8868619.

[38] Y. Xu, D. Li, Q. Xie, Q. Wu, and J. Wang, "Automatic defect detection and segmentation of tunnel surface using modified mask r-cnn," Measurement, vol. 178, p. 109 316, 2021, ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2021.109316. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224121003158.

[39] T.-Y. Lin et al., Microsoft coco: Common objects in context, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312.

[40] H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," CoRR, vol. abs/2007.07588, 2020. arXiv: 2007.07588. [Online]. Available: https://arxiv.org/abs/2007.07588.

[41] L. Perez and J. Wang,
The effectiveness of data augmentation in image classification using deep learning, 2017. arXiv: 1712.04621 [cs.CV].

[42] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," eng, Journal of big data, vol. 6, no. 1, pp. 1–48, 2019, ISSN: 2196-1115.