Robust Deep Learning Algorithms for Selective Laser Melting Monitoring

by

Bodi Yuan

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Summer 2019

Robust Deep Learning Algorithms for Selective Laser Melting Monitoring

Abstract

Robust Deep Learning Algorithms for Selective Laser Melting Monitoring

by

Bodi Yuan

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Sara McMains, Chair

The objective of this research is to enable real-time in-situ monitoring for the Selective Laser Melting (SLM) process, by providing diagnostic feedback from monitoring that can be used to automate and adjust SLM system parameter settings. The ultimate goal is to improve SLM product quality and manufacturing productivity. We propose a deep learning approach to monitoring that takes in-situ videos as input data fed to convolutional neural networks (CNNs). We describe the entire monitoring framework, including running SLM experiments, collecting SLM video data, image processing for ex-situ generated height maps, generating labels for in-situ data from ex-situ measurement, and training CNNs with labeled in-situ video data. Experimental results show that our approach successfully recognizes the desired SLM process metrics (e.g. size, continuity) from in-situ video data.

In order to train effective CNNs, besides collecting extensive SLM video data, we also need to label it. We have automated the process of generating labels from ex-situ measurements of the corresponding finished SLM experimental output. The ex-situ measurements provide high-precision height maps for the product surface, to which we apply our proposed image processing algorithm to calculate process quality metrics as labels.

However, our proposed automated labeling approach requires high-precision height maps, which are generated from an expensive Structured Light Microscope. It might not be readily available to other researchers and institutions, or not enough machine time may be available to label all experiments for which there is video. Thus there might not be enough labeled data to train effective CNNs. This research also combines semi-supervised learning with our original approach to address this problem. Semi-supervised learning method enables other researchers to address the problem without requiring a huge amount of labeled data.

In addition, in practice, another issue is label noise. Even though the data labels were generated using high-precision height maps, the labels are not perfect and might still contain incorrect labels, known as “noisy” labels. We propose novel approaches to improve neural networks’ performance when they are trained under label noise. The proposed approaches can be easily combined with other existing approaches that address the label noise problem to further improve the prediction accuracy, with very few additional hyperparameters that need to be tuned. Experimental results demonstrate that our approaches can significantly improve CNN models' prediction accuracy when training neural networks with noisy labels.

To my family and friends.

Contents

List of Figures

List of Tables

Acknowledgments

The past four years, my unforgettable journey, comes to an end. At this moment of accomplishment, I would like to take this opportunity to extend my thanks to everyone who supported, helped, and advised me through my Ph.D. life.

Foremost, I was lucky enough to meet my Ph.D. advisor, Professor Sara McMains, at the very beginning of this journey and worked under her supervision since then. Sara has been a truly dedicated mentor who cared so much about my work. We discussed research questions hours after hours in group meetings. We worked together revising our paper drafts days after days before paper submission deadlines. We explored new research directions weeks after weeks for our new research proposal. She is not only my advisor, but also my important friend. We celebrated together when I passed the qualifying exam, and she has been a constant source of support and encouragement through all the up and downs of the past four years. I owe her my eternal gratitude.

I am also extremely indebted to my dissertation committee members, Professor Alexei A. Efros and Professor Hayden Taylor, for their invaluable advice and patient guidance at all stages of this work. Profound gratitude also goes to my other qualifying exam committee members, Professor Tarek Zohdi and Professor Trevor Darrel, for their insightful comments and extensive discussions around my work.

A Berkeley Fellowship funded me for the beginning stages of the Ph.D. program. I will forever be grateful to Lawrence Livermore National Lab (LLNL) for their generous funding in the past two years. My Ph.D. work wouldn't be possible without help from various colleagues at LLNL, especially Brian Giera, Gabe Guss, Manyalibo Matthews, Aaron Wilson, Stefan Hau-Riege, and Phillip DePond. Brian Giera is my mentor at LLNL and has been so dedicated to his role. He continuously supports my work, provides me meaningful and helpful feedback, and connects me with other colleagues. It is a great pleasure and honor to work with him and also other researchers at LLNL!

I also thank my lab mates, Youngwook Paul Kwon, Hannah Budinoof, Xiang Li, Sara Shonkwiler, and Jeronimo Mora, for their helpful discussions and feedback. In addition, lots of friends, especially Weidong Zhang, Jianyu Chen, Zining Wang, and Yang Gao, have been so supportive and caring over the years. Wishing you all the best in the future!

Last, but by no means least, I would like to express my deepest love to my parents and my girlfriend, Chaoran Guo. They have been extremely supportive and have strong faith in me at all times. I love you all.

Chapter 1

Introduction

1.1 Selective Laser Melting (SLM)

Selective Laser Melting (SLM) is one of the most widely used Additive Manufacturing (AM) techniques. SLM uses a high-powered laser to melt metal powders layer-by-layer (Figure 1.1), in order to produce an object with the desired shape. Compared to traditional subtractive manufacturing, SLM can process a variety of materials and has a relatively low production cost. However, one big challenge of SLM is that the quality of SLM products is sensitive to many factors, such as laser characteristics (power, speed, spot size, etc.), metal powder characteristics (size, purity, thickness, etc.), and build environment (ambient temperature, oxygen level, etc.) [124]. With so many related factors that can affect an SLM product's quality, it is difficult to find an initial good set of operating parameters or optimize the existing system parameters. In addition to that, even when using fixed system settings, SLM product quality may still vary and is hard to control. The lack of SLM process repeatability is a barrier for industrial AM manufacturing progression.

In order to improve SLM automation and product quality, diagnostic feedback from monitoring during the SLM building process can be used to automate parameter settings. Currently, using a variety of sensors, a great amount of SLM process data can be easily generated. This data could enable researchers to train machine learning models to monitor the SLM process; however, to implement machine learning in practice requires labels for the collected training data, and traditional manually labeling will be too time-consuming and labor-intensive. This is common for SLM related problems and even in other AM areas, since oftentimes data is easily generated but difficult to label.

1.2 SLM Monitoring

We envision that a real-time SLM monitoring system could provide helpful feedback for identifying defects, adjusting laser parameters, and compensating for variability in the powder and build environment.

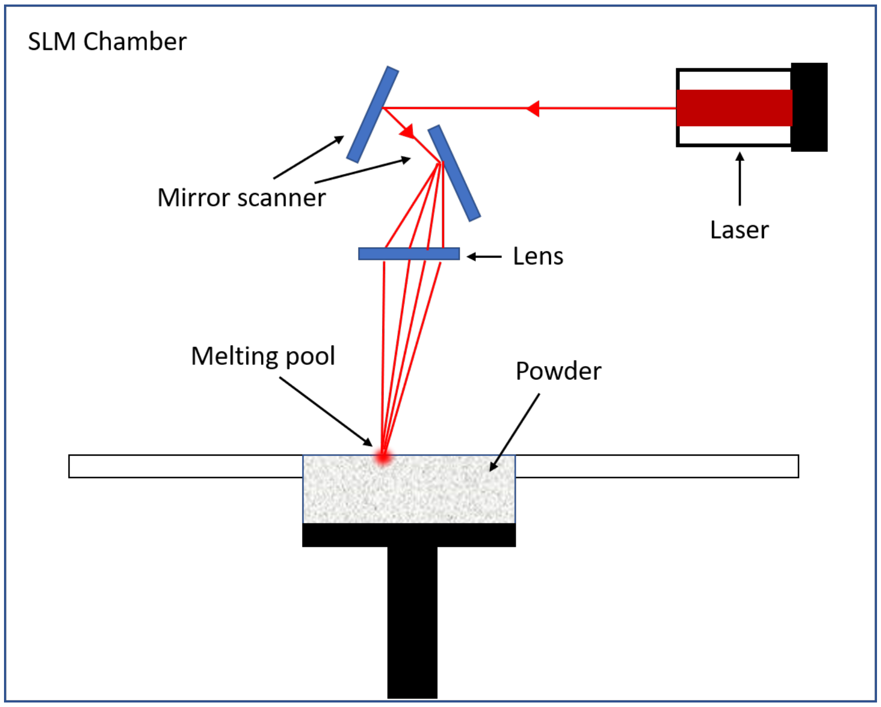


Figure 1.1: *SLM system*

In our work, in-situ videos recorded during the SLM build process are used as input data points from which we train convolutional neural networks (CNNs) to recognize, in real time, desired process metrics. We demonstrate a two-step approach of applying machine learning to SLM. First, our collaborators at LLNL ran experiments to collect over one thousand in-situ SLM videos that we labeled, making use of post-build measurements via our ex-situ height map analysis algorithm. Our labeling algorithm provides detailed labels, including both average width and width standard deviation of each printed track. Next, we train a Convolutional Neural Network on a portion of the labeled video data. With the remaining in-situ SLM data, we test our neural network and find that it predicts track widths and standard deviation using only 10 ms videos, without the need for ex-situ measurements. We evaluate our prediction results based on the coefficient of determination of the predicted labels and the ground-truth labels. The Convolutional Neural Network architecture we designed is general and also accurately predicts the continuity of tracks without changing the model architecture. For the continuity classification, we evaluate our trained model on overall prediction accuracy. The experimental results show that our proposed method can predict accurate outputs using only the in-situ videos.

1.3 SLM Monitoring from Semi-supervised Learning

To train a well-performing CNN for monitoring SLM (and other additive manufacturing systems), it is challenging to collect labeled video data, though relatively easy to collect unlabeled data. Therefore, we also explore using a semi-supervised learning technique to reduce the effect of having relatively little labeled data. We train a semi-supervised CNN model based on the temporal ensemble method [75] with a small amount of labeled data and a large amount of unlabeled data. The same as our supervised learning approach, we evaluate our semi-supervised model on a labeled test dataset, using the coefficient of determination for evaluating regression and accuracy for evaluating classification; in all cases the addition of unlabeled data improved performance, sometimes significantly.

1.4 Label Noise

Besides insufficient training labels, another common issue in practice is label noise. Different from traditional datasets such as those that researchers use to train neural networks, datasets collected "in the field" are rarely perfectly labeled.

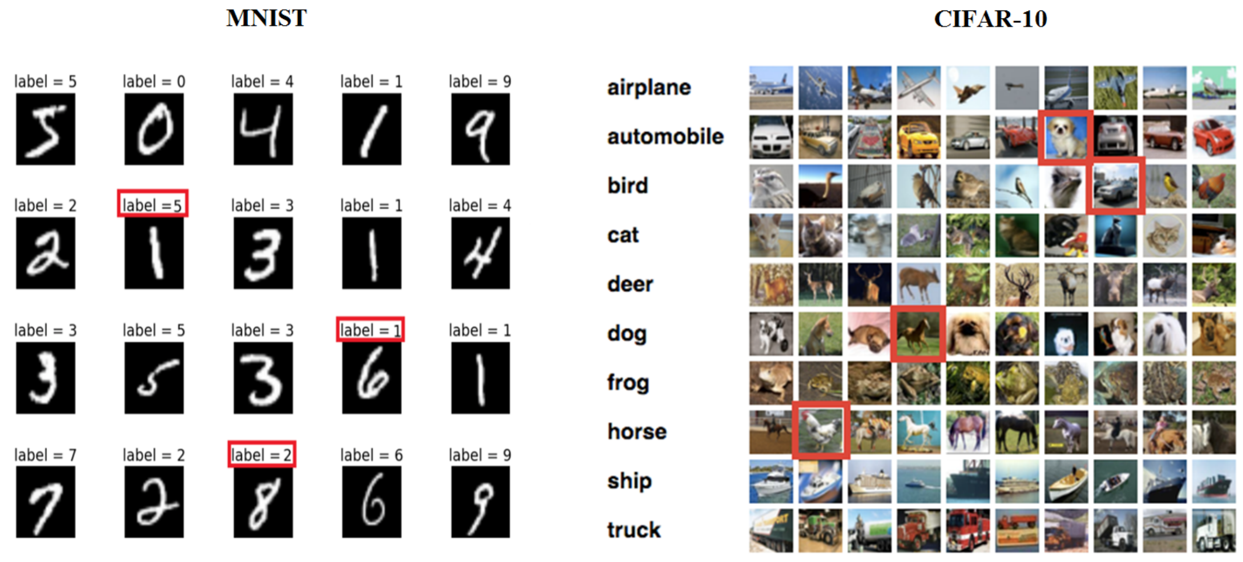


Figure 1.2: *Examples of training data sets corrupted with noisy labels*

To address the problem of incorrect labels in training data for deep learning, we propose two novel and simple training strategies, Iterative Cross Learning (ICL) and Consensus Refinement Learning (CRL), that significantly improve the classification accuracy of neural networks when the training data has noisy labels. Testing our methods on public datasets such as MNIST [76] and CIFAR [66] with partially shuffled labels (Figure 1.2), our methods significantly improve the classification accuracy of existing methods when the data labels have noise, especially in heavy noise situations.

1.5 Summary of Contributions

This dissertation will discuss an entire SLM monitoring framework, an in-situ video labeling algorithm from ex-situ measurements, semi-supervised monitoring, and addressing the label noise problem.

We describe our proposed SLM in-situ monitoring framework and also our proposed height map image processing algorithm for labeling experimental SLM tracks. The proposed method can be used for not only SLM monitoring, but also other AM technology monitoring. The major contributions related to AM monitoring are:

- Our approach predicts the average width, width standard deviation, and continuity of SLM tracks from in-situ videos with high accuracy.
- Our proposed SLM monitoring approach enables providing feedback in real time for SLM product quality control and automating system parameter setting, in order to improve SLM product quality and manufacturing productivity.
- Our proposed image processing algorithm provides accurate pixel-level classification results for ex-situ height maps, which enables us to train machine learning models with in-situ data for a variety of track metrics, such as track width, continuity, etc.

Our SLM monitoring approach is a data driven approach. Our collaborators collected a great amount of data comprised of SLM in-situ videos and height maps. The dataset has been organized and labeled at a height-map pixel-level scale, enabling dataset re-use for further research.

Having lots of data but lacking labels is very common in AM. Semi-supervised learning alleviates this problem and has a potential to be more broadly used in this area. We also implemented and analyzed a semi-supervised approach to SLM monitoring. Our major contributions related to semi-supervised learning in AM are:

- We successfully designed and implemented a semi-supervised learning framework for SLM monitoring, which includes data collection, partial data labeling, CNN model training, and results verification. The experimental results show that the semi-supervised learning approach outperforms the supervised learning approach with the same amount of labeled training data. To our knowledge, we are the first to apply a semi-supervised learning approach to any AM monitoring task.

Besides the insufficient labels problem, label noise is another important problem in practice. We focus on classification problems with training label noise and propose effective approaches. Our major contributions related to the label noise problem are:

- We propose two simple but effective training algorithms, Iterative Crossing Learning (ICL) and Consensus Refinement Learning (CRL), to address label noise. Our proposed approaches consistently improve neural networks' prediction performance in the presence of incorrect labels, for all underlying models tested.
  - ICL: When using ICL, we randomly partition the original dataset into two subsets on which independent networks are trained, then retrain after introducing random labels for training data on which they disagree. This helps to alleviate the problem of models overfiting training label noise.
  - CRL: Our experiments show that if the original training data is noisy, even though the consensus dataset will be smaller, the same network architecture trained on this more accurately labeled data still achieve better performance when we use our proposed CRL approach.
  - CRL: We define a new metric, "consensus failure rate," that we use to analyze our proposed Consensus Refinement Learning's effectiveness. The theoretical analysis shows the principles of how Consensus Refinement works.
- Both of ICL and CRL have no restriction on the underlying network architecture and only a few extra hyper-parameters, enabling others easily to combine existing methods that address the label noise problem by modifying the network architecture with our approaches to get even better performance.

Prior to details of these results, we provide an overview of the literature in the next chapter.

Chapter 2

Background and Related Work

2.1 Additive Manufacturing

Additive Manufacturing (AM), commonly referred as "3D-printing" technology, builds up 3D components by adding materials, in contrast to traditional subtractive manufacturing. AM technologies are made possible by Computer Aided Design (CAD), which provides 3D object models as input for an AM product. AM technologies have been widely used in many industries, including automotive, medical, and aerospace.

AM-related technologies have been researched since the 1970s, and have been developed into numerous different types of AM processes (as shown in Figure 2.1), such as Direct Energy Deposition (DED), Powder Bed Fusion (PBF), Fused Deposition Modeling (FDM), Material Jetting, Vat Photopolymerisation, and so on. These different processes are used depending on the different material and precision requirements.

Compared to traditional manufacturing, AM has unique advantages. AM enables one to build much more complex parts with fewer manufacturing steps. For simple-shape and complex-shape parts of the same size, with AM there may be little to no difference in build time or cost. As for designers, AM also provides much more flexibility. Designers can design any complex models with CAD and make changes easily. Moreover, AM is able to work on a variety of different materials.

However, AM still has its own challenges. First of all, high-precision additive manufacturing machines are expensive. Even with expensive machines, the finished product's surface accuracy is often not high enough to satisfy industry requirements. It usually requires post-processing to improve the product surface quality. In addition, the AM product mechanical properties can be poor. The finished product might have defects such as unintended pores (voids).

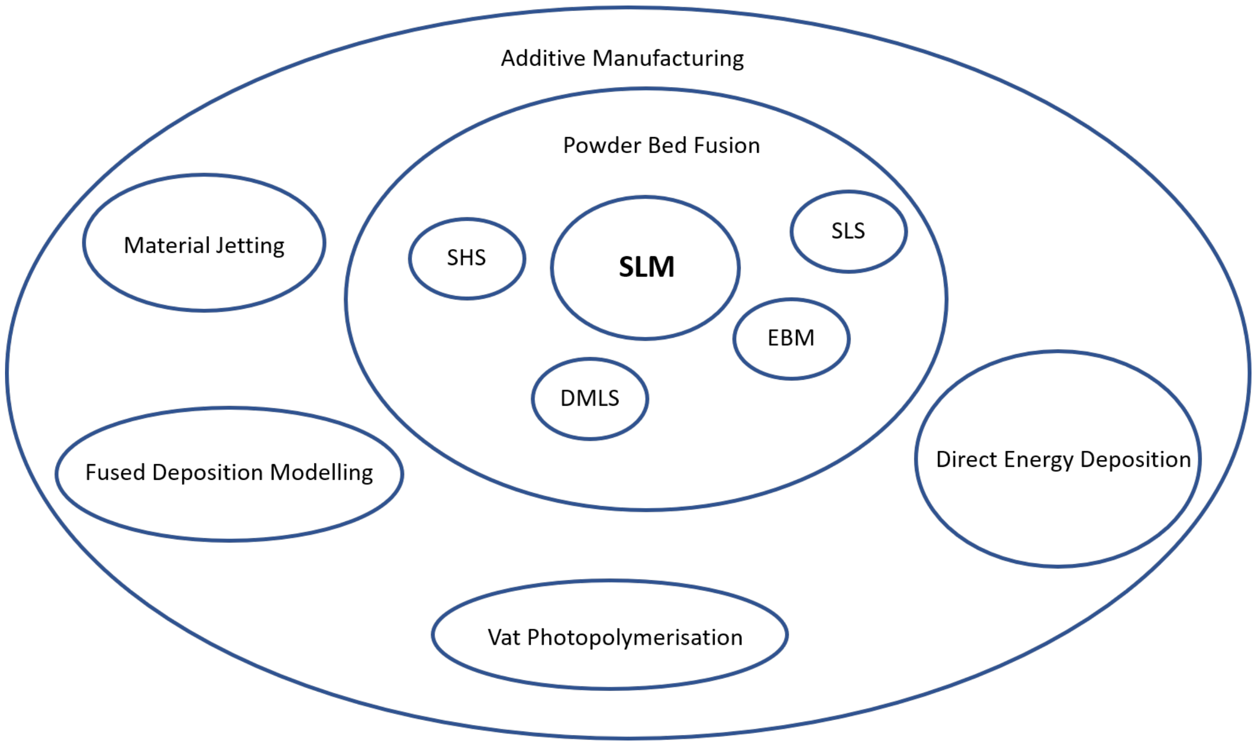


Figure 2.1: *Different types of AM technologies*

2.2 Powder Bed Fusion (PBF)

Powder Bed Fusion (PBF) [13, 61, 62] is an additive manufacturing technology that produces metal or polymer parts, layer by layer, by melting powdered metals, alloys, or polymers, with a heat source such as a high-power laser [139], a beam of electrons, or a thermal printhead. Considering the beneficially large set of metallic and polymer powders available to PBF for fabricating objects of virtually any shape, the technique is versatile and ideally suited for applications such as rapid prototyping and light-weighting [37], since PBF also has a fast building speed and relatively low cost among other AM technologies. Depending on the requirement, PBF can work on a variety of materials, such as metal, polymers and so on.

PBF comprises various types of techniques, such as Selective Laser Melting (SLM), Selective Laser Sintering (SLS), Electron Beam Melting (EBM), Selective Heat Sintering (SHS), and Direct Metal Laser Sintering (DMLS). Though these different PBF techniques have different usages, can work on different materials, and use different physical mechanisms, all of them have very similar building process. Thus, though we focus on the research related to SLM process, our work can be easily applied to other PBF techniques as well.

2.3 Selective Laser Melting (SLM)

One of the most popular PBF technology is Selective Laser Melting (SLM), which shows promise to revolutionize the fabrication of metal parts. However, the physical process of SLM is complex, making it hard to control product properties by adjusting the system settings [36, 61]. Regan et al. [99] focus on correlations between build environment and product quality, but the results only describe the rough trend of the correlations rather than a clear mathematical relation.

In fact, SLM product quality is affected by a variety of system build parameters. The final material properties of parts made via SLM are extremely sensitive to the powder properties [123] (e.g., powder shape, flow characteristics, porosity, laser absorptivity [111, 15], etc.) and laser parameters (e.g., beam size, power, scan rate, etc.) [55, 56]. As such, it is a significant challenge to identify the optimal operating parameters to rapidly and reliably produce parts with the desired properties without defects [9]. Furthermore, many types of SLM defects arise due to inherent variability in the powder properties [22], bed thickness non-uniformity [115], and laser parameters and scan paths that result in improper powder melting [26]. Many researchers identify the key input variables and metrics to analyze the quality of finished products. Spears et al. [124] state that there are over 50 different SLM process input variables (grouped into four categories: laser and scanning parameters, powder material properties, powder bed properties and recoat parameters, and build environment parameters) that affect the quality of the finished product. Given so many input variables, it is not only hard to analyze the impact of each variable, it is also hard to run experiments to test each variable independently. Berumen et al. [12] argue that there are five determining areas of the building process: powder quality, temperature management, process gas atmosphere, melt pool behavior, and documentation process. This variety also suggests the complexity of SLM product quality control. Although physics-based models offer some insights into optimized parameter selection and underlying mechanisms of SLM, the required computational expense limits their application for real-time uses. This suggests the critical necessity for in-situ SLM monitoring.

Moreover, especially with improper system build parameters, SLM finished products might have lots of different kinds of defects [2, 20, 41, 58, 87] or unsatisfied mechanical properties [81]. These issues can cause serious problems when the product is being used. For instance, an over-powered slow-moving laser can vaporize the metal, leading to “keyhole” voids, while an underpowered fast-moving laser might not melt powder at all [15]. Kusuma [72] studies only the effect of laser power, scan speed, and laser energy density, showing that even if only these three variables change, the relation between the finished product metrics and the studied variables is complex and not easy to predict. Slotwinski et al. [123] focus on the effect of powder properties and study two different powder materials: stainless steel and cobalt-chrome. Their experimental results show that a variety of characteristics of powder properties, such as powder size, shape, density, crystallographic form, etc., have effects on the finished product quality.

Even after optimizing the build parameters and having a good initial set of parameters [128], SLM monitoring system enables feedback and automated process control during a build [124, 32], in order to improve the final product quality and production efficiency.

2.4 SLM Monitoring

Currently, most PBF process monitoring approaches, including for SLM, rely on a variety of non-contact sensors [60], such as optical, thermal [102] and acoustic [134, 118] sensors. These different types of sensors enable researchers to address the monitoring problem from different signals. Overall speaking, optical sensors can provide image data, which has more richness than single-point input data from thermal and acoustic sensors. Moreover, visual imaging equipment is appealing for SLM monitoring systems because it is relatively inexpensive and provides non-contact sensing [124]. Thus, most researchers, including us, focus on using optical image data. Aminzadeh et al. demonstrated layer-by-layer detection of fusion defects from images using a Bayesian classifier [5]. That work proposed to detect real-time events such as material ejected by applying manually-set thresholds to high speed near-infrared images of the melt pool. Fox et al. [35] increased the image resolution relative to the standard deviation of measured track width, which results in improved predictions of the final track width and topography. Kruth et al. [69] proposed to reduce laser power proportionally to an integrated signal from a photodiode calibrated against a CMOS camera, and got results with smoother overhang structures. Abdelrahman et al. [1] used images of the print bed taken after laser melting, a level sets method that can detect intentionally created defects. Machine vision algorithms can identify pore defects [3], and multi-fractal image analysis can characterize layers with balling, cracks and pores, and no defects [138].

SLM monitoring methods typically fall into two categories: in-situ that occur in real-time, or ex-situ [86, 4, 38, 1, 138] that provide precise, automatic, robust quality detection and measurement from off-line data in certain circumstances. However, an unavoidable problem of ex-situ methods is that they require an entire part to be fully built before it is assessed for quality. Even if there is an obvious defect that arises during the build process, an ex-situ method inherently cannot detect it in time to correct it. Therefore, compared to in-situ methods, ex-situ methods are not only slow, but also incur additional time and material losses due to failed builds. For these reasons, we focus only on in-situ methods.

There are several different strategies to monitor the SLM process in real time. Grasso et al. [44] gives four major categories for in-situ monitoring depending on the feature of interest: (i) the powder bed, (ii) the layer slice, (iii) the track along the scan path, and (iv) the melt pool.

For the powder bed monitoring, Li et al. [83] proposed a stereo vision based method for powder bed monitoring, which can calculate the powder bed 3D surface topology from two aligned cameras. Compared to a single camera monitoring system, Li's method will be more computationally expensive and there might be 3D reconstruction errors when calibrating the two cameras. Erler et al. [31] proposed monitoring the thickness of the powder bed to recognize structural defects. This approach requires an expensive surface height sensor that can measure surface height in-situ, and the height map generation is slow. Scime, et. al. [114] leverages CNN architectures, as we do here, to identify the location of possible powder spreading anomalies for a variety of material systems.

For layer slice monitoring, Foster et al. [34] proposed a layer-wise powder bed monitoring method using a single camera and regular color ("electro-optical") images. It takes and analyzes images for each powder layer before the laser scans and fuses the powder for that layer. Kleszczynski et al. [65] proposed using high resolution gray-scale images during the build process to recognize unsatisfied material properties of the entire product. However, both methods can only provide high-level monitoring for product overall qualities rather than detailed information during the SLM process.

Track scanning methods [57, 10, 28] take temperature data as input signals. However, common problems of this approach are that a temperature sensor's sensitivity is low and the data is noisy, resulting in imprecise monitoring results.

The predominant approach is in-situ melt pool monitoring, which is the method we use in our research. Lott et al. [85] designed an optical system for in-situ melt pool monitoring with an overhead camera monitoring the melt pool, but without experimental verification. Craeghs [24] proposed analyzing the measured melt pool image data to find the position of the frame with maximum average intensity, which may correspond to overheating. Clijsters et al. [23] proposed using high-speed optical sensors to record the intensity of the melt pool, and recognize the overheating of the melt pool and detect positions of possible pores. Repossini et al. [108] focuses on a specific feature of the melt pool, spatter analysis, and develops spatter-related descriptors to classify different energy density conditions. Grasso et al. [45] sets up a monitoring system similar to ours, which also includes in-situ monitoring (for under-melting and boiling via infrared imaging) and ex-situ labeling methods (via x-ray image data), and then uses conventional image processing methods and a statistical learning technique, Principle Component Analysis (PCA). Grasso et al. [46] integrates the acquisition of infrared images with conventional image processing methods for in-situ monitoring. All of these approaches have many experimental parameters whose values are likely tied to their specific system and dataset and may not be transferable to other tasks and systems.

2.5 Convolutional Neural Networks (CNNs)

In recent years, deep convolutional neural networks [77] have shown exceptional performance on many image and video recognition, segmentation, and classification problems. The first time that CNNs were successfully applied to a research problem was to recognize images at hand-written digit (0–9) in the MNIST dataset [78]. With relatively limited training data and computational resources in the 20th century, the proposed CNN, containing only two convolutional layers, still showed excellent performance on the classification dataset, MNIST [76]. However, this original CNN only worked well on simple tasks and datasets with small numbers of classes. In 2012, a deep convolutional neural network, AlexNet [68], was introduced and showed much better classification performance than conventional computer vision methods on the very large dataset, Imagenet [27], with over 1 million images in 1000 categories. Benefiting from much more training data and computational resources including better computers and powerful GPUs for parallel computing, it became possible to train much deep neural networks, which resulted in better performance. Even though a variety of new CNN architectures were proposed after AlexNet, most of them have a special architecture design [127, 53] or only address a specific problem, sush as 3D object detection [88, 103], or face recognition [113]. Currently, VGGNet [122] and ResNet [48] are the most widely used CNNs because of their standard architecture design and even better performance than AlexNet's. In our work, we have choosed to use VGGNet, ResNet, or simplified versions of these as our underlying models.

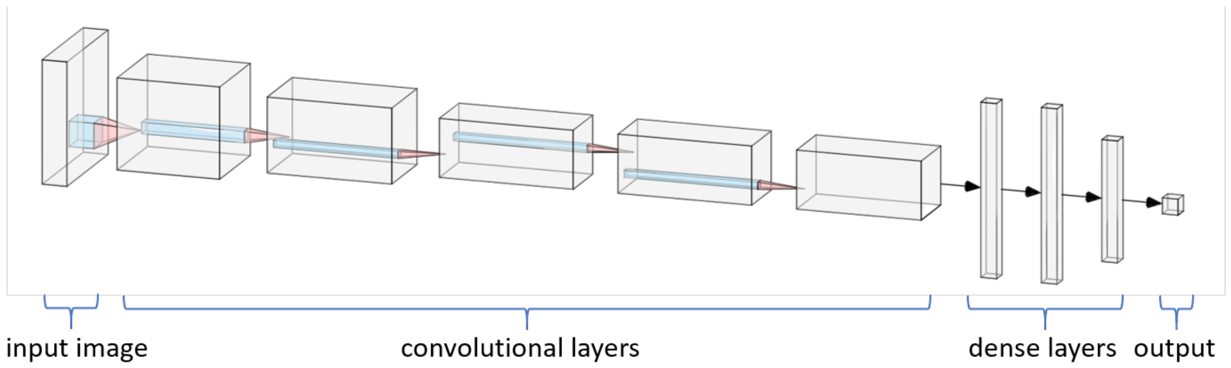


Figure 2.2: *An example of CNN architecture. [Figure credit: NN-SVG]*

In addition to improved performance compared to conventional computer vision methods, CNNs are also convenient because relevant image features are automatically extracted by the model itself [142]. This allows us to address several different problems from the same dataset even with the same CNN architecture just by training with different labels, whereas conventional computer vision methods typically require significant manual oversight to extract different features and tune algorithm parameters.

Commonly, CNN architectures consist mainly of convolutional layers (conv-layers) and dense layers (also referred as fully connected layers) [77], as shown in Figure 2.2. Conv-layers process convolution operations and encode the input data into lower dimensional feature spaces. Dense layers predict the output from the encoded features; neurons in dense layers are fully connected with each other. Some CNN architecture design techniques are currently widely used in all kinds of CNNs; we also used them in our work. We briefly describe some important ones here: activation, pooling, and dropout. Some fundamental activation functions such as ReLU [93], sigmoid, and tanh [120], are used to increase the non-linearity of the neural networks. Activation layers usually follow the conv-layers and dense layers. Pooling layers [68], such as max-pooling and mean-pooling, can reduce the dimension of the data. Pooling layers usually follow the activation layers. Dropout is a regularization technique and can reduce overfitting. Dropout layers are before the activation layers and commonly used after only dense layers without conv-layers before them.

Besides the architecture design, a variety of training related techniques have also been developed. Backpropagation [78, 49], shorthand for "the backward propagation of errors," is almost always used in training neural networks. It is a method to calculate gradients for updating neural networks' weights in each layer. The most commonly used classification loss function is the cross-entropy loss [119, 25], which measures the likelihood of input data belonging to each category. In our work, we use the Adam optimizer [64], which is an extension of stochastic gradient descent (SGD) [16]. Adam has an adaptive learning rate, thus it is more tolerant of learning rate range than SGD.

2.6 Semi-supervised Learning

Current convolutional neural networks rely critically on large labeled training datasets. Several popular examples benefit from vast publicly-accessible labeled datasets. Collecting data is generally relatively cheap and easy; however, labeling data requires time-consuming human labor [105]. The same is true for our problem: collecting in-situ videos is automated and the laborious part has traditionally been labeling video data. Semi-supervised learning trains with a large amount of unlabeled data and a small amount of labeled data [107]. Therefore, semi-supervised learning [143, 21] is a promising approach since it reduces the need for large labeled datasets while not suffering from the level of performance loss expected from unsupervised learning, as shown in Figure 2.3. In semi-supervised learning, the dataset can still be huge but only a very small portion of the dataset might be labeled. This will significantly reduce the expense of the labeling task and increase the generalizability of deep learning for practical use.

Co-training [14, 97, 133, 82, 70] is a conventional semi-supervised learning method. It trains multiple classifiers based on different descriptive features of a dataset and takes agreement of those classifiers to iteratively augment the labeled dataset by adding labels to unlabeled data, which is similar to our idea. But the most important constraint of co-training is that it requires that original data has different independent features for different classifiers to learn, which is not always available in practice. Similarly, Yarowsky [140] proposed a algorithm to start training classifiers with a small set of labeled training examples, then incrementally augment labeled data with unlabeled data and repeat this process. Lee [80] proposed a simple method to generate pseudo-labels for semi-supervised learning. It picks the class that has the maximal predicted probability as the label for each unlabeled data point and then trains networks in a supervised fashion. Major problems of this approach are lack of robustness and easily overfitting the labeled training data. Kingma et al. [63] proposed a deep generative model to improve semi-supervised learning performance by generative approaches, which can generate new samples that follow the same probabilistic distribution of a given training dataset. Rasmus et al. [106] proposed a ladder network, which is a powerful semi-supervised neural network for image classification tasks. The ladder network makes

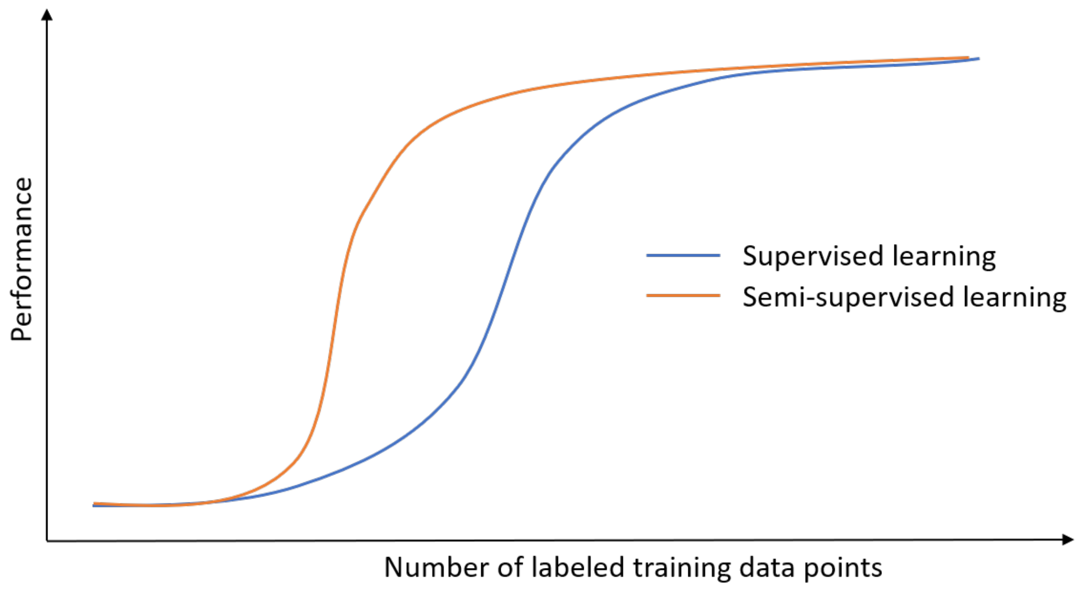


Figure 2.3: *Semi-supervised learning and supervised learning*

use of both supervised learning with labeled data and unsupervised learning with unlabeled data simultaneously. Inspired by ladder networks, Laine et al. [75] proposed the temporal ensemble approach, which has even better performance than ladder networks. We also used temporal ensembles for SLM monitoring in our work. More details are described in Chapter 6.

2.7 Label Noise

In large datasets, besides insufficient labeled data, another labeling related issue is label noise. The current state-of-art in visual object recognition still relies on supervised-learning [50, 67, 107, 116, 142]. These supervised learning systems require the correctness or the cleanliness of the labeled data in order to achieve the expected performance.

In most research work, it is assumed and also optimized under the assumption that the labeled datasets have all correct labels without noise. However, in practice, data often has noisy labels [18, 94, 144], from being manually mislabeled, or mislabeled by inaccurate but fast and cheap automated algorithms. Moreover, in semi-supervised learning as described in the above section, if a portion of the input is well-labeled and the remainder is originally unlabeled, noisy labels may come from assigning random labels to the unlabeled data. In many situations, it is hard to get a completely clean dataset, and it requires much effort to manually clean the data. Figure 7.1 shows an example of noisy labels. Without data cleaning, the existing label noise will significantly lower the trained models' performance.

To understand label noise's effect on neural networks, Nettleton et al. [96] compared the different degrees of noise and how they affect four supervised learners. Rolnick et al. [110] presented extensive experimental results and gave theoretical analysis on cross-entropy loss [42]; both theory and practice indicated that neural networks have some robustness to label noise. Moreover, Veit et al. [131] show that residual networks [48] especially tend to be even more robust to noisy labels. The ubiquity of noisy labels has led to studies on the preprocessing stage to clean the data in order to train the supervised learning systems [8, 17].

There are several popular strategies to address the label noise problem. One strategy is to improve the neural networks' architecture. Sukhbaatar et al. [126] augmented the underlying network with an extra linear layer that learns the label noise distribution. Some other work is similar to or based on this approach [54, 101, 11].

Another strategy is to modify the loss function. Patrini et al. [101] introduce a loss correction approach, which achieves a high experimental classification accuracy, but their results also suggest that the approach doesn't provide much improvement to the original cross entropy loss function. Azadi et al. [6] optimized the stochastic Alternating Direction Method of Multipliers algorithm to make CNNs more robust to label noise. Natarajan et al. [95] studied the effect of mislabeled data for the classification problem and presented two approaches specifically targeting the binary classification problem with noisy labels by estimating the original loss function and using surrogate loss functions. The first one constructs an unbiased estimator for the original loss function without knowing the noise distribution; the other one estimate the original loss function using noise distribution information. Liu et al. [84] propose re-weighting data points' importance, which can also be thought of as modifying the loss function. Miao et al. [89] propose a label-noise-robust boosting algorithm with non-convex loss functions. In contrast, Rooyen et al. [130] propose using a convex loss, which specifically targets uniformly random distributed label noise on classification problems.

Other researchers tackle the label noise problem by using semi-supervised learning, training on a small portion of cleanly labeled data as well as a much larger amount of noisy labeled data. Fergus et al. [33] proposed a graph-based semi-supervised learning label propagation method, and other semi-supervised training frameworks addressing label noise have been recently proposed [121, 30]. Mnih and Hinton [92] formulate loss functions and use a semi-supervised learning approach to pretrain the networks. Then they train deep neural networks for different kinds of label noise and successfully improve models' performance. Lee [80] proposed a network that is trained in a supervised manner but reduces the effect of noisy data with a "De-noising Auto-Encoder," which is a neural network encoding input data points in lower dimension space and recovering the input data points from the low dimension representation. Xiao et al. [135] proposed a training framework using the noise distribution information; the training loss calculated depends on the probability of training labels being incorrect. Veit et al. [132] proposed an approach that first pre-trains a network using the large noisy dataset, and then fine-tunes the network using both the clean data and the reduced-noise data.

More similar to our proposed methods in this dissertation (as described in Chapter 7 and Chapter 8), other researchers use new training frameworks designed to address label noise. Vahdat [129] proposes a new training framework and a conditional random fields [74] model that represents the correlations between noisy and clean labels. Conditional random field is a framework for building probabilistic models, originally proposed by Lafferty et al. for modeling sequential and structured data. Vahdat's method shows impressive performance on the CIFAR-10 dataset with noisy labels. But this approach requires extra "auxiliary information" about the noise distribution that is not always available in practice. Reed et al. [107] address label noise by using a bootstrapping method. During the training, by changing weights for data points, models will pay more attention to data points that have labels consistent with model predictions.

Chapter 3

SLM Monitoring Framework

Our goal is to provide in-situ monitoring for the SLM process via in-situ videos. To accomplish thus, we need to set up experiments, collect data, train models, and test models. Thus, using hardware designed and also data collected by our collaborators at LLNL, we developed algorithms for our experiments. In this chapter, we will overview our proposed SLM monitoring framework.

3.1 SLM Monitoring Framework

The entire framework (Figure 3.1) includes: running SLM experiments, collecting SLM in-situ video data, generating height maps from ex-situ measurement, image processing for height maps, generating labels for in-situ data from ex-situ measurement, training CNNs with labeled videos, and testing the trained CNNs.

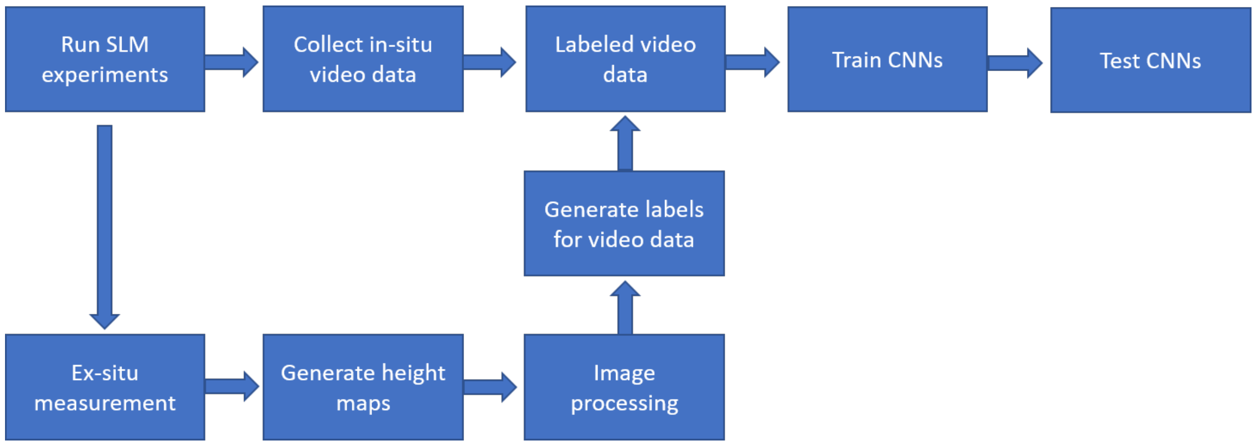


Figure 3.1: *The flowchart of the entire SLM monitoring framework*

We propose to train CNNs to recognize desired metrics (e.g. track size, continuity) from in-situ videos. First of all, we need to run SLM experiments to collect SLM videos as training data. In our work, we use single-layer straight SLM tracks for experiments and analysis. Our collaborators built over one thousand weld tracks in total and set up a high speed camera to collect in-situ video data while the tracks were built. The SLM scanning process was ran with different laser parameters, including speed and power. All the tracks were built with the same spacing and uniformly distributed on the plate, which will be convenient for future analysis. Once the data has been collected, to train neural networks, the next step is to label the training data. However, just from these in-situ videos, it is hard to recognize the desired metrics via visually checking. Thus, we propose to use an ex-situ approach to automated label these videos. Our collaborators use a structured light microscope to measure the product surface height and generate height maps. Running our image processing algorithm (Chapter 4) with these high precision height maps enables us to calculate the desired metrics of each track. The calculated metrics will be used as labels for our in-situ video data. Once the video data is labeled, we can train our designed CNNs on the training data for the different metrics. Finally, we test our approach on the test dataset.

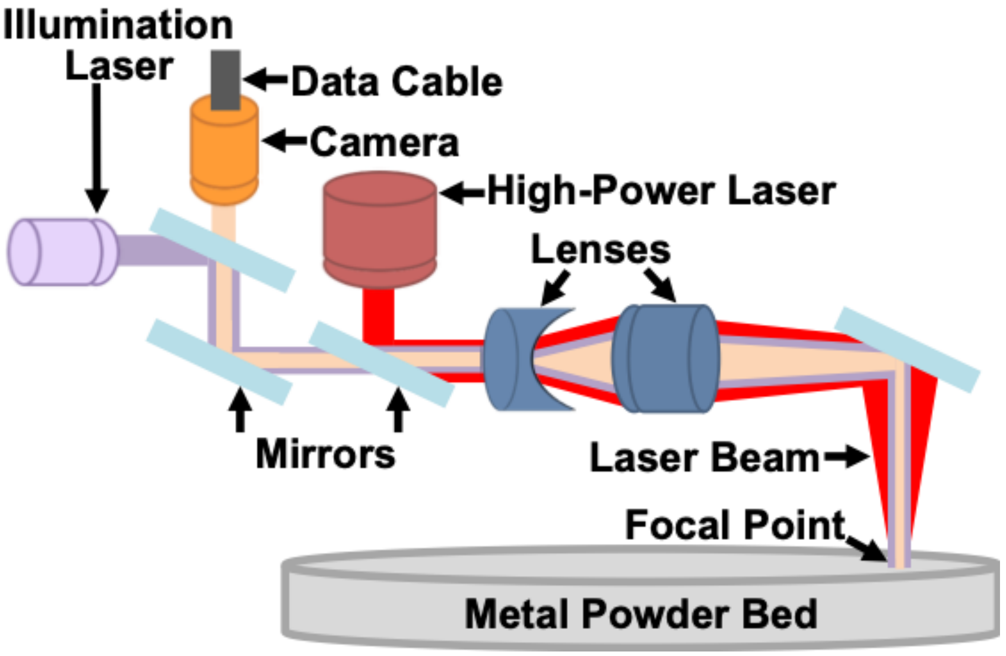


Figure 3.2: *High powered and illumination lasers are co-aligned with a high speed camera whose shared focal point is scanned across the metal powder bed via adjustable mirrors and lenses. Camera video data is saved to a nearby computer for each track site. [Figure credit: Brian Giera]*

3.2 Hardware System Design

An Aconity LAB system from Aconity3D is used for welding single tracks of 316L stainless steel. The system uses a carbon fiber brush to spread metal powder evenly across a stainless steel plate in an argon-purged environment. Galvanometer mirrors scan the high-power laser across evenly spaced track sites on the powder bed. In-situ video data is recorded at a frame rate of 1 kHz using a 10-bit Mikrotron EOsens MC1362 camera incorporated into the optical system diagrammed in Figure 3.2. The illumination laser and Mikrotron video camera are co-aligned with the high-power laser. These components share a common moving focal point, fixed at the melt pool while the laser scans each track site. The camera pixel size is 14 μm/pixel. The experimental machine is shown in Figure 3.3.

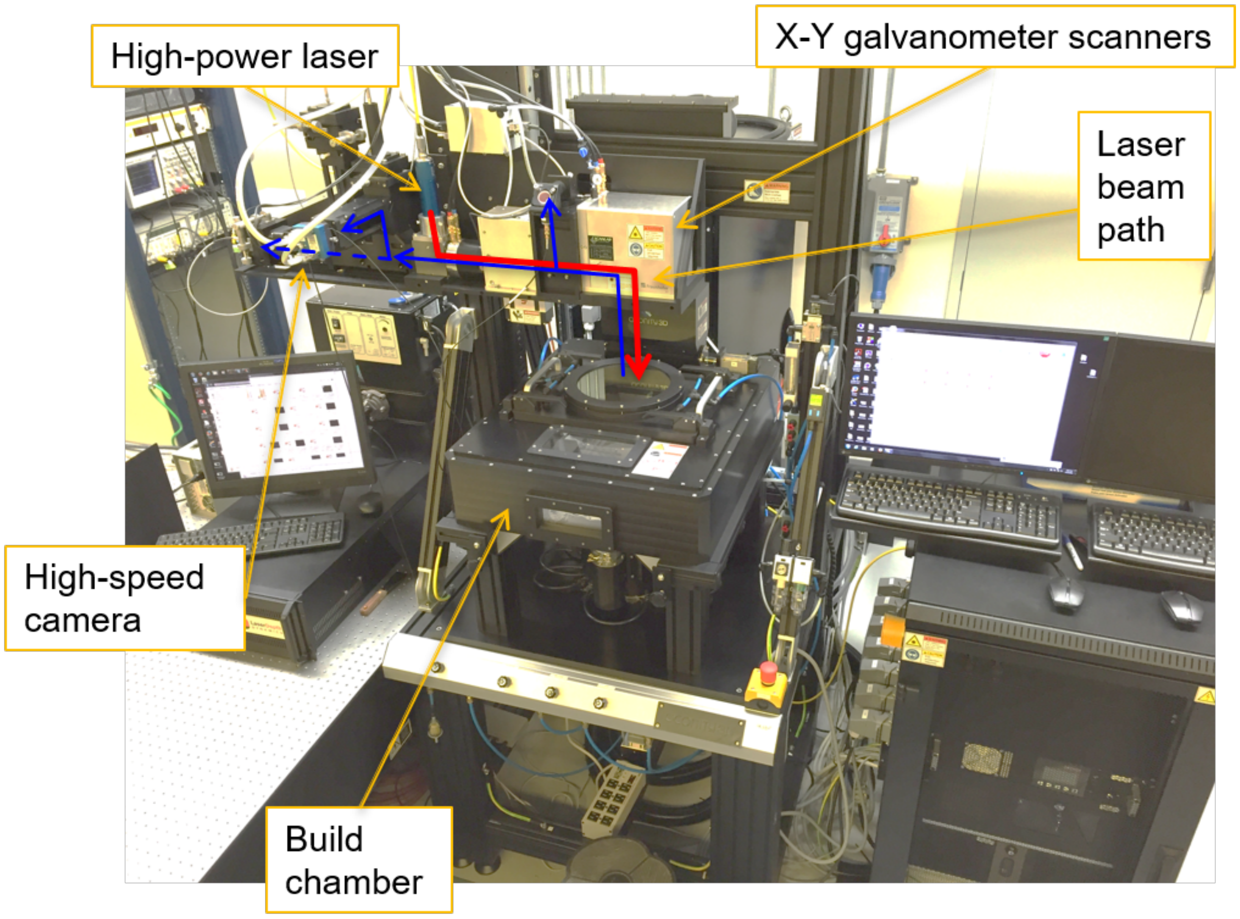


Figure 3.3: *SLM machine and data collection system at LLNL. [Figure credit: Gabriel Guss]*

3.3 Experimental Data Collection

In what follows, SLM experiments are performed while simultaneously recording video; isolated 5 mm long track welds are created under a variety of laser speed and power settings. Note that the experiment parameter combinations are randomly chosen from 11 possible scan speeds and 11 possible laser powers in evenly-spaced increments between 100 – 400 mm/s and 50 – 375 W, respectively, with each set of laser conditions repeated up to seven times. The laser spot size is a fixed 106 μm diameter.

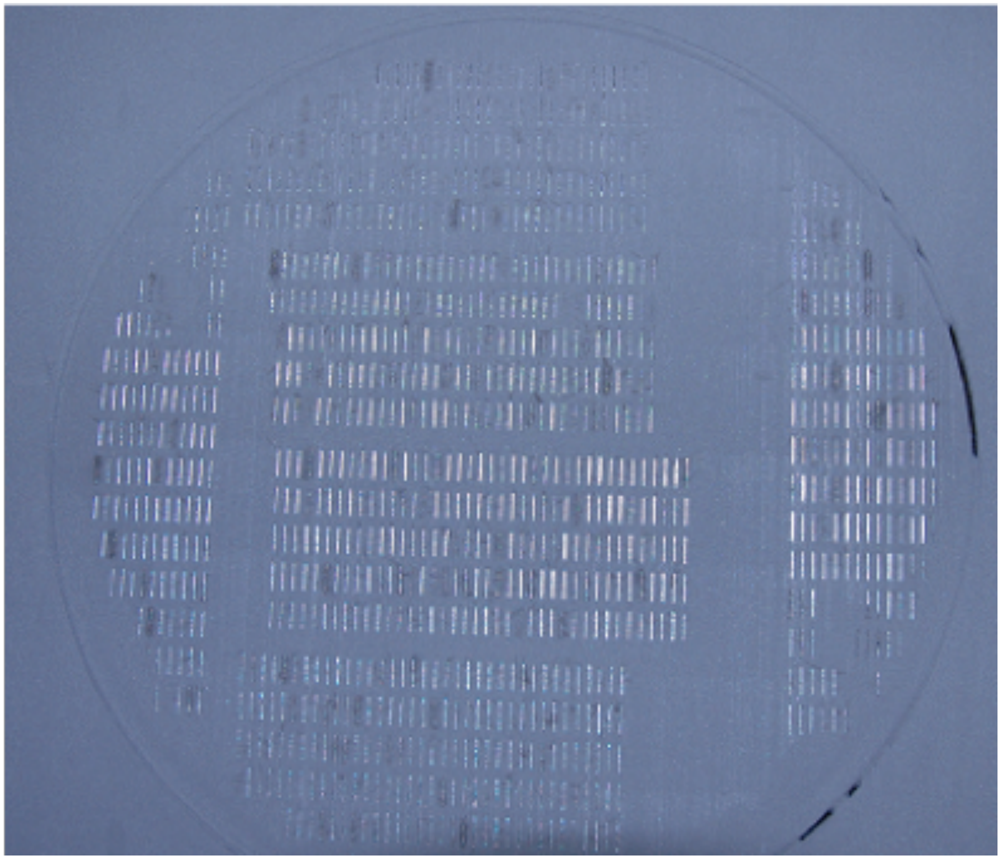


Figure 3.4: *Examples of experimental tracks on one of the build plates. [from LLNL experiments]*

For our SLM experiments, we use 316L stainless steel powder. The powder is spread to ~50 μm thickness on top of a 180 mm diameter stainless-steel plate (only a single layer is printed). The finished experimental tracks are shown in Figure 3.4 and a grayscale electro-optical image (images captured by a high-resolution camera equipped with a telephoto zoom lens) is shown in Figure 3.5.

The in-situ video for each individual track has 12 – 50 frames depending on the scan speed, each frame of size 256 × 256; an example is shown in Figure 3.6. These videos

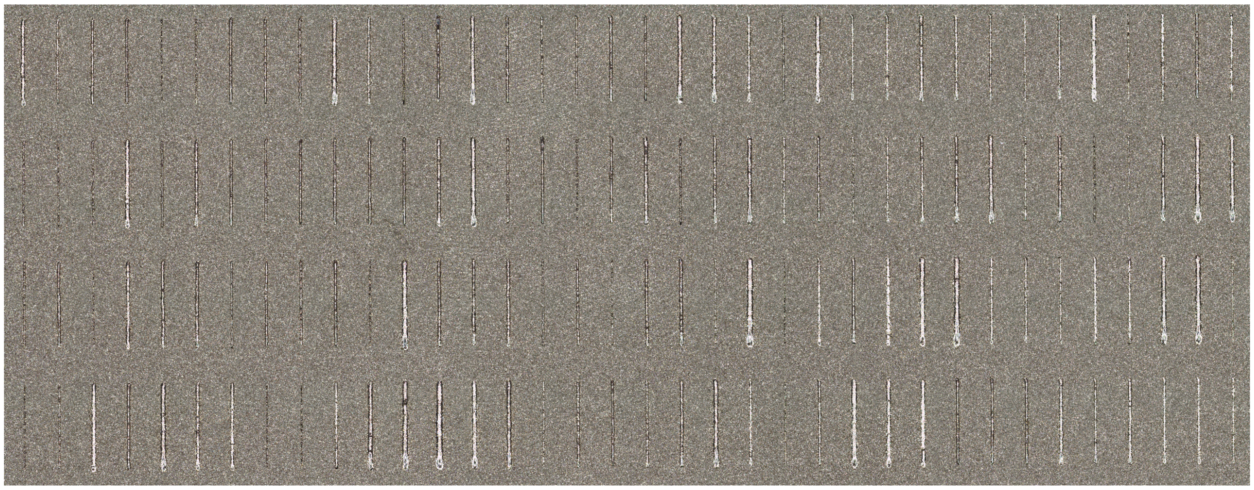


Figure 3.5: *Electro-optical image of experimental tracks. [from LLNL experiments]*

need to be pre-processed and then will be used as our training dataset and test dataset. There might be some frames without a clear laser spot such as the first and the last two frames in Figure 3.6. This is because the camera recording is not perfectly coordinated with laser scanning. This issue will be addressed when pre-processing the videos, as described in Section 5.1.

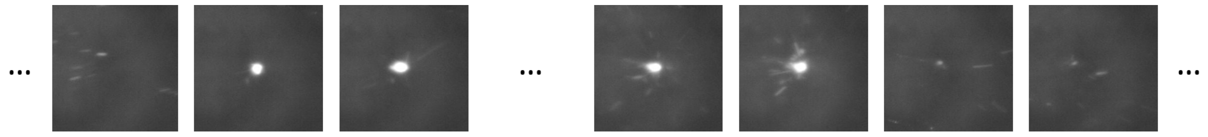


Figure 3.6: *An example from our collected video data*

It is difficult to directly determine a clear relationship between these video frames and our desired track metrics. Thus, after laser scanning and unwelded powder is removed, height maps of bare laser tracks are generated and analyzed with the height map analysis algorithm (described in Chapter 4) to determine the per-pixel average track width, standard deviation of the track width, and give each track a Boolean continuity label that identifies whether or not a track is continuous. Then each in-situ video is assigned three labels and used in training or test sets for our machine learning algorithm. A single supervised machine learning architecture is used to predict these three ex-situ measurements from in-situ video data, as described in Chapter 5.

Chapter 4

Height Map Processing

In the previous chapter, we introduced our SLM monitoring framework and discussed how the in-situ video data was collected. Using the collected in-situ videos described in the previous chapter, we hope to train neural networks on these kind of videos and learn the relations between short videos and desired track metrics. The training process requires meaningful labels; however, it is difficult to obtain labels directly from a sequence of video frames. We proposed to generate labels from ex-situ measurement of the corresponding finished tracks.

Our collaborators use a structured light microscope to measure the surface height of the plate. This measurement provides us high precision height maps for all the tracks. Then we treat height maps as images and run our proposed image processing algorithm to separate the background pixels and track pixels. With pixel-level segmentation on each track, it becomes easy to characterize each track. Based on our segmentation results, we can calculate a variety of track metrics such as the track's width, standard deviation of width, length, continuity, and so on.

4.1 Height Maps

Our collaborators performed ex-situ measurements of corresponding laser printed tracks on the top of the steel plate. After all tracks were completed, unwelded powder is removed from the plate and then a Keyence VR3000 3D microscope is used to generate a height map of all laser tracks. The entire plate is scanned in less than 4 hours. An example of a height map is shown in Figure 4.1. Height maps are generated with 40x magnification via the microscope. Each pixel in the height map corresponds to 29.5 μm of real physical size in both the horizontal and vertical directions.

The measured height maps provide detailed information for each track. The height maps are much more precise and provide more information than the original electro-optical images (Figure 4.1). Even the polishing marks on the background plate are visually clear in the height maps. With such high-resolution height information, we can characterize the tracks' properties with our proposed image processing algorithms described in the next section.

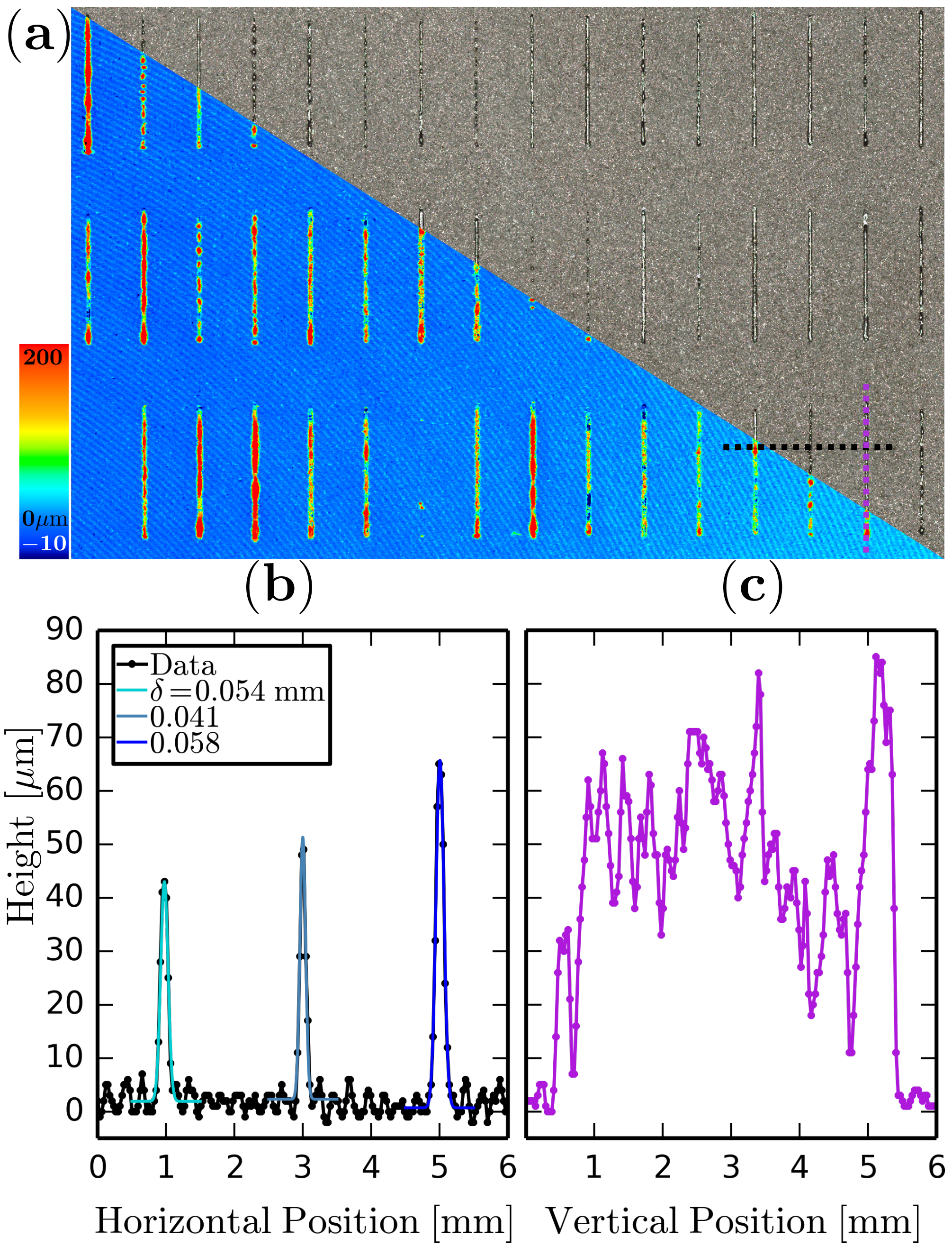


Figure 4.1: *(a) An example of an original electro-optical image and height map measured from structured light; (b) the actual height corresponding to the horizontal dash line in (a); (c) the actual height corresponding to the vertical dash line in (a). [Figure credit: Brian Giera]*

In the height map, since tracks are formed by fusing the metal powder together and then unfused powder is removed, track pixels usually have height higher than the background plate. However, if the laser power is too strong, it might actually etch the background plate, causing the surface height to be even lower than the background height. To differentiate, we propose an image processing algorithm to distinguish tracks from the background.

4.2 Image Processing Algorithm

Given a height map as input, we hope to automatically classify each pixel as a "track pixel" or "non-track pixel." To be even more accurate, the "non-track pixel" has two categories, "background pixel" and "etching pixel." By correctly identifying all the "track pixels," then we can calculate metrics for each track. The key idea here is to remove the background and then analyze each track.

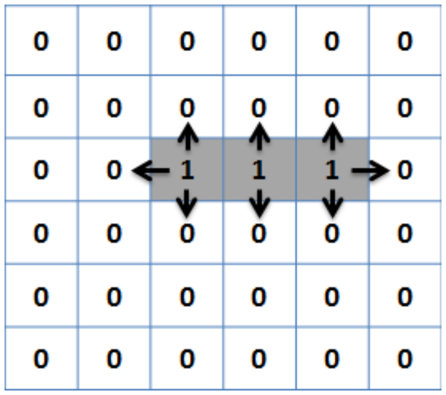
Our proposed image processing algorithm first determines the height maps' orientations and rotates each height map to the same orientation. Since the background plate surface is not perfectly level, to avoid the effect of global variations in the background height, we divide the entire height map into patches, with each patch containing a single track. Finally, running our two-round background removal algorithm with a minor additional filtering step addressing a few remaining noisy pixels, we will remove the background and get a clean segmentation result. Knowing the pixel level segmentation information, we can then easily calculate a variety of track metrics.

4.2.1 Height Map Orientation Automation

The height maps might have horizontal tracks or vertical tracks depending on how the plate was oriented during scanning. If we can rotate input tracks to the same orientation, either horizontal or vertical, these tracks will be much easier to analyze and for future processing. Therefore knowing the original orientation of the tracks is important and helpful; we can calculate pixel gradients to determine whether the tracks are vertical or horizontal.

The first step in determining track orientation is to binarize the input image with a simple global threshold, which will tend to generate a noisy binary image; the goal is to identify the portions that are tracks, although there will also be lots of noise. We choose the threshold by calculating the mean of the whole height map and adding a small constant (we use 0.01 mm) to make it less noisy.

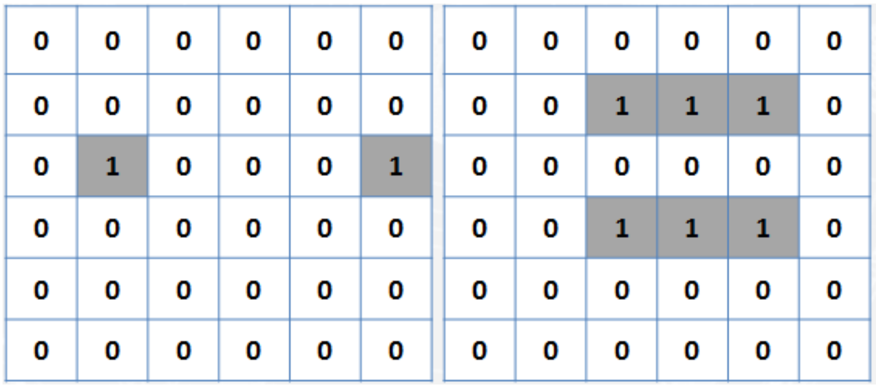
From that binary image, the next step is to calculate the gradient (Figure 4.2) of all the valid pixels in both the x-direction and y-direction, respectively, using the kernels $[-1, 0, 1]$ and $[-1, 0, 1]^T$ to do the convolution for the x-gradient and y-gradient, respectively. The x-gradient and y-gradient can tell us whether the track is horizontal or vertical, since there should be one direction with higher value, corresponding to the track orientation. Thus, we can easily compare the sum of the absolute values of the x-gradients to the sum of the absolute values of the y-gradients for all pixels in the image.



0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	1	1	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Figure 4.2: *Calculation of the gradient*

In the Figure 4.2 and 4.3 example, the sum of absolute values from the x-gradient is 2 and the sum of absolutes value from the y-gradient is 6. From that information, we can infer that the tracks are likely horizontal in the figure.



0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

0	0	0	0	0	0
0	0	1	1	1	0
0	0	0	0	0	0
0	0	1	1	1	0
0	0	0	0	0	0
0	0	0	0	0	0

Figure 4.3: *Left image: absolute value of x-gradient; right image: absolute value y-gradient*

4.2.2 Dividing Height Map into Patches

In each height map, all the tracks are uniformly distributed and the spacing constants between tracks (offsets in x and y) are known. For each track, we can divide the entire height map into patches and each patch contains only a single track. Thus, we just need to focus on the pixels in that patch to analyze that single track. We would like put each track near the middle of each patch to avoid having different parts of the same track appear in different patches. Therefore, the first step is to find a start-point, where we put the first patch as a reference point. Then the other patches' positions could be easily calculated according to the spacing constants.

At first, we can start from a default start-point, such as (0,0), the upper left corner of the image. We temporarily treat this as the top left corner of the first patch and generate uniform patches in the whole image based on the known spacing constants. These patches are temporary patches, which are used for calculating the correct start-point. All of these temporary patches will later be shifted the same distance in the same direction to move the tracks to the middle of the patches.

In each patch, then we compute a local "shift vector," which is the vector from the default start-point of the patch to the correct start-point, and then average all of these vectors as a global "shift vector." Then we can find the correct start point coordinates by adding this global "shift vector" to the original start point. To calculate the "shift vector," we take the vector from the patch center to the center point of the track as the "shift vector." In each patch, we average the positions of all the valid pixels, which are defined as value 1 pixels in the binary patch. The averaged position should be roughly at the center of the track, since most valid pixels are track pixels, despite some noise. However, some patches might have only a few track pixels or even have no track at all. In such situations, the calculated local "shift vector" will be very unstable. Thus, we calculate the global "shift vector" as the weighted average of local "shift vectors," where weights are the total number of valid pixels in each patch. The weight serves to make patches with a track inside more important than other patches containing only noise. In Figure 4.4, the shift vector is $[2, -0.5]$ and the weight is 4.

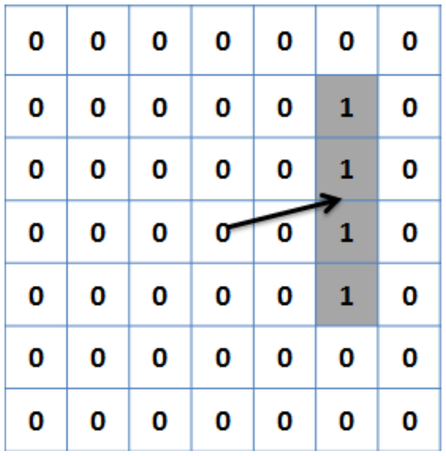


Figure 4.4: *Calculation of the "shift-vector" in each patch*

4.2.3 Background Removal and Noise Filtering

We need to remove the background and filter remaining noise in patches to analyze each track. We proposed a height segmentation algorithm that takes each patch as an input and provides pixel level classification results as outputs (Track/Non-track). Since the background is noisy in terms of height, and there are global variations in the height of the underlying plate (it is not completely flat), we propose a two-round filtering method that is robust

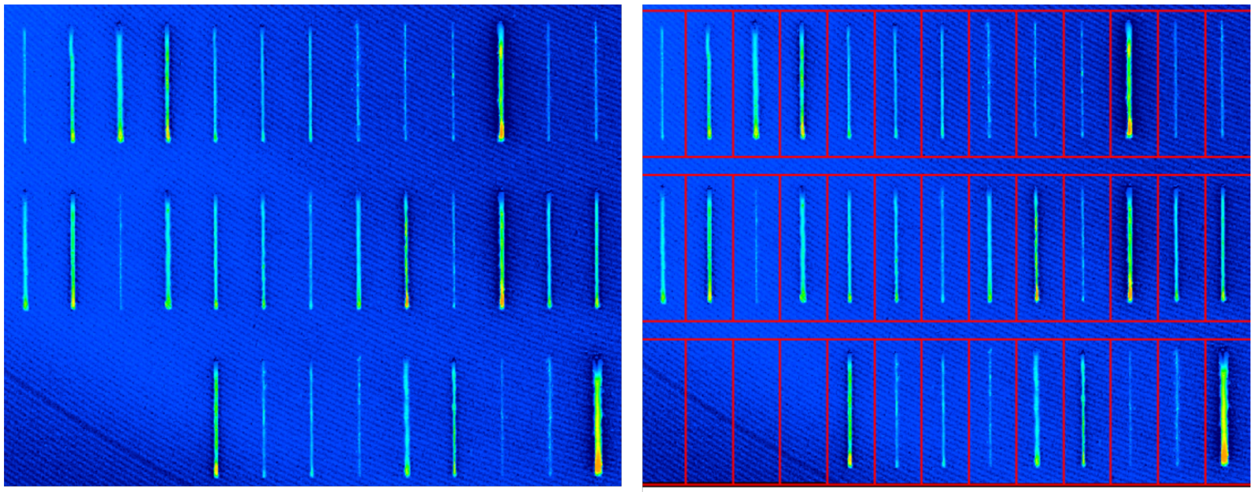
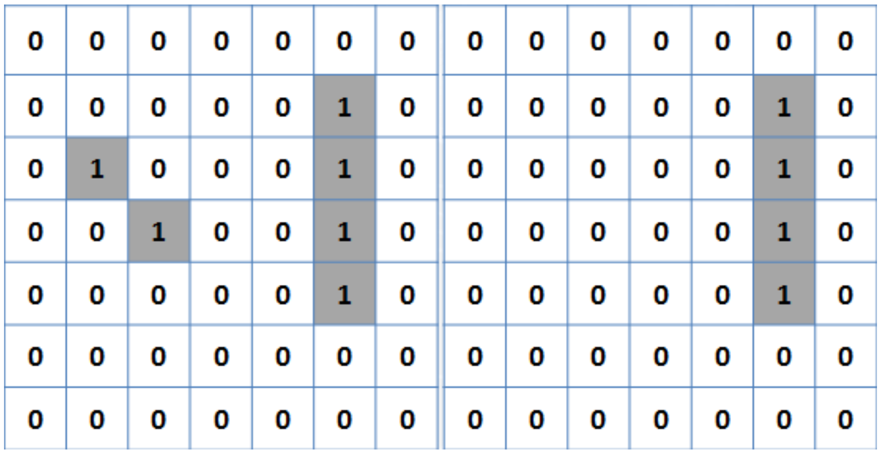


Figure 4.5: *Divide height map into patches*



0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	1	0	0	0	1	0
0	0	1	0	0	1	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	1	0
0	0	0	0	0	1	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figure 4.6: *Left image: binary map after background removal; right image: filtered binary map*

to background noise and able to provide a clean segmentation result. Pseudo code of the algorithm is given in Algorithm 1.

(a) Calculate the average height $\bar{h}_1$ of the height map in patch $\mathcal{H}$. This will generally be just a bit higher than the height of the background plate since track pixels are at most 10% in our experiments.

(b) Re-calculate the average height $\bar{h}_2$ of pixels with height in the range $(\bar{h}_1 - \tau_1, \bar{h}_1 + \tau_1)$. These are likely to be background pixels rather than track pixels.

(c) Binarize the height map $\mathcal{H}$ as $\mathcal{B}$ with the threshold $\bar{h}_2 + \tau_2$, in preparation for final denoising.

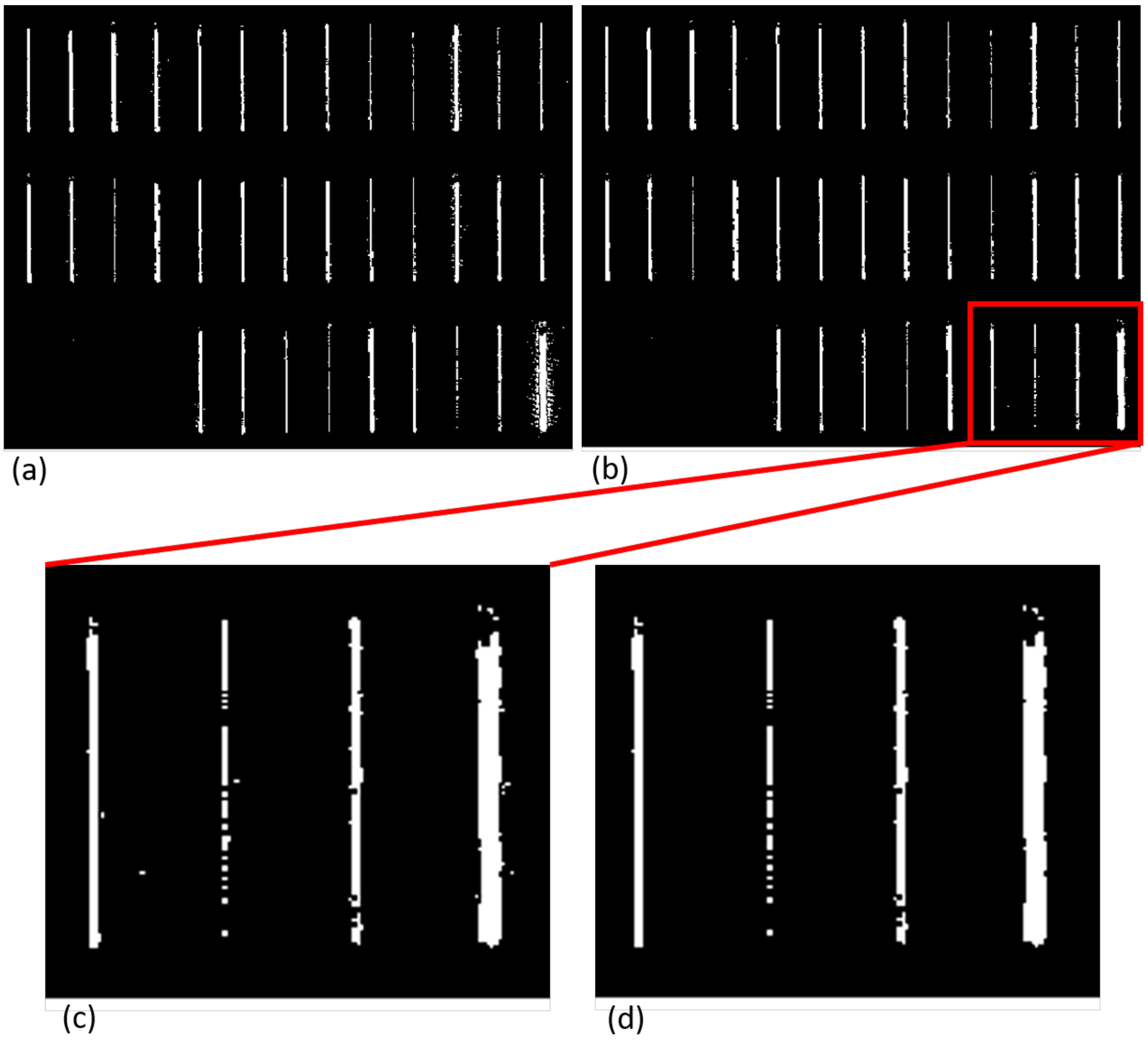


Figure 4.7: *Background removal and filtering. (a) Binarized height map after running the first round of background removal; (b)(c) binarized height map after running the second round of background removal; (d) binarized height map after running post-filtering*

(d) Calculate the summation S_j of each column j in the binary map $\mathcal{B}$.

(e) Set all the pixels in column j to the "non-track" (zero) value if S_j is less than a noise threshold T expressed as a percentage of the desired track length.

(f) Return the output binary map $\mathcal{B}$, which has the same size as the original height map patch $\mathcal{H}$, with the "Track/Non-track pixel" classification result on each pixel.

Algorithm 1 Height map segmentation

```
1: $\mathcal{H} \leftarrow$ input height map patch
2: $H, W \leftarrow$ vertical and horizontal sizes of the height map patch $\mathcal{H}$
3: $\tau_1, \tau_2, T \leftarrow$ constants for threshold
4: // Round 1:
5: $h_1 \leftarrow \sum_i^H \sum_j^W \mathcal{H}_{ij}$
6: $\bar{h}_1 \leftarrow h_1 / (\sum_i^H \sum_j^W 1)$
7: // Round 2:
8: $h_2 \leftarrow \sum_i^H \sum_j^W \mathcal{H}_{i,j} \cdot (\bar{h}_1 - \tau_1 < \mathcal{H}_{ij} < \bar{h}_1 + \tau_1)$
9: $\bar{h}_2 \leftarrow h_2 / (\sum_i^H \sum_j^W 1 \cdot (\bar{h}_1 - \tau_1 < \mathcal{H}_{ij} < \bar{h}_1 + \tau_1))$
10: // Post-filtering
11: $\mathcal{B} \leftarrow$ binarize $\mathcal{H}$ with threshold $\bar{h}_2 + \tau_2$
12: for $j \leftarrow 1$ to $W$ do
13:     $S_j \leftarrow \sum_{i=1}^H \mathcal{B}_{ij}$
14:     if $S_j < T$ then
15:         for $i \leftarrow 1$ to $H$ do
16:             $\mathcal{B}_{ij} \leftarrow 0$
17:         end for
18:     end if
19: end for
20: return $\mathcal{B}$
```

The appropriate values of the constants τ_1, τ_2, and T will depend upon the height of the track pixels relative to the background pixels, and on the variability. We use $\tau_1 = 0.01$ and $\tau_2 = 0.01$ (all heights are in units of pixels) for our setup, in which the typical standard deviation of the background height within a patch is around 0.002, and the difference between track pixel height and background pixel height is over 0.2 (100 times the standard deviation of the background). In other words, these constants are set at about 5% of the height of the tracks above the background, which is also equal to about 5 standard deviations of the background height. We set $T = 5\%$ of the track length, which is 10 pixels out of 200 in our experimental setup. Example output of our filtering algorithm is shown in Figure 4.6. Any automated labeling process must rely upon a relatively precise machining and polishing process for the background steel plate compared to the track height; with the two-orders-of magnitude height:standard-deviation ratio in our setup, there is less sensitivity to the choice of these constants.

Overall, the background removal and filter results are given in Figure 4.7. This results in fairly clean pixel labels.

Note that sometimes there are pixels with heights lower than the background plate height. This is caused by the laser melting not only the powder but also the powder bed, which results in etching. For the current work, these are classified as non-track pixels; identification of etching is left to future research.

4.2.4 Image Processing Results

With running all the previous mentioned image processing algorithm together on the original height maps, we will get the the final results as shown in Figure 4.8.

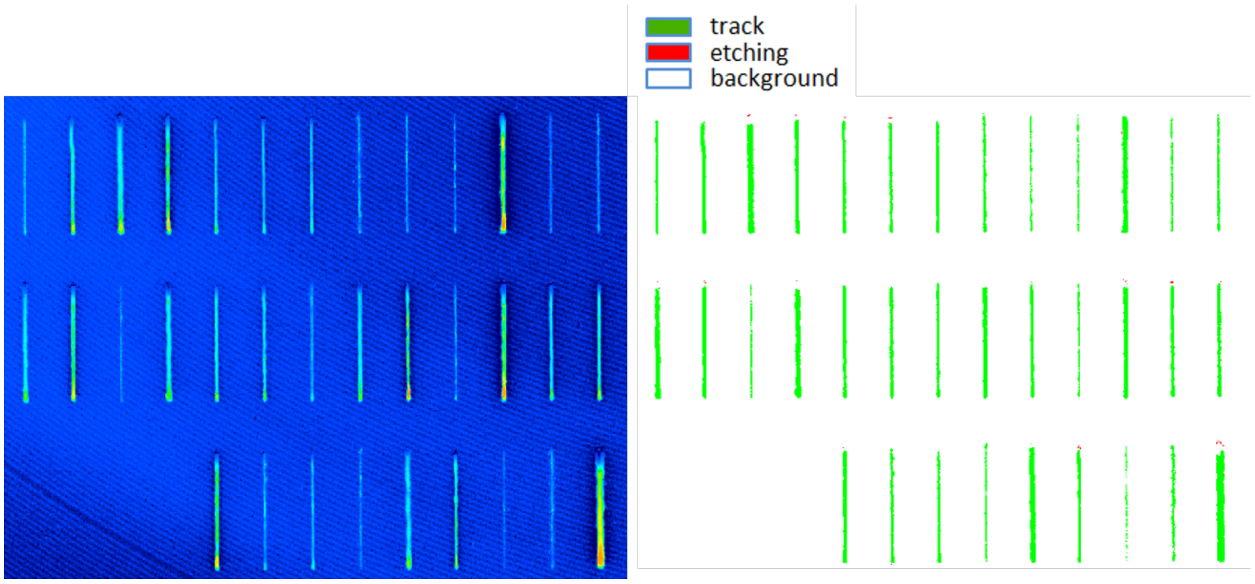


Figure 4.8: *Height map (left) and final results (right) of the proposed image processing algorithm.*

4.3 Calculated Metrics

In the results shown above, the background has been removed and the track has been extracted. Then we need to calculate the information we need. In the following example, we use white pixels to define background, green pixels to define tracks and red pixels to define non-track pixels with lower height than background plate. Since we only care about the track pixels, we can delete rows and columns which are outside of the track as shown in Figure 4.9.

Having identified the track pixels within the binary map $\mathcal{B}$, it is straightforward to calculate the relevant track metrics (Figure 4.10), such as the length l, the average width

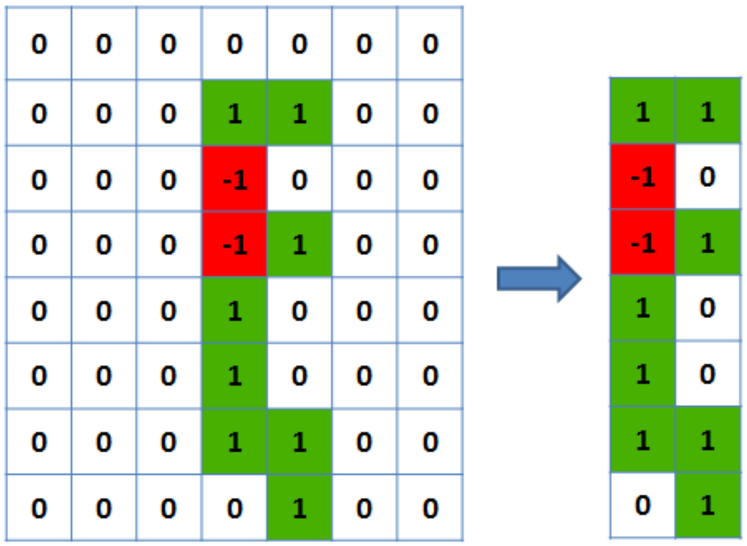


Figure 4.9: *The patch after deleting non-track rows and columns*

δ, both in pixel units, and the continuity $\mathcal{C}$ of the track. Although we focus on the track width and continuity, we can also provide other laser track metrics from the height map, e.g. roughness, average height, etc. We calculate the following:

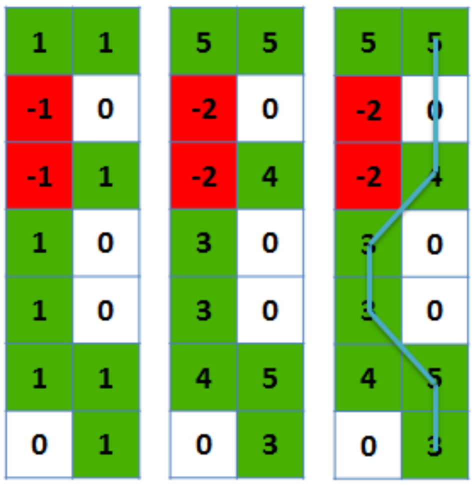


Figure 4.10: *The trinary map, the corresponding height map and the "ridgeline" of the track.*

(1) Length l of the track. We define it by the number of rows of the patch after deleting background rows and columns:

$$l = I' - I^{\star},$$

where $I^\star$ is the index of the first row that has track pixels, and I' is the index of the last row that has track pixels. In the example shown in Figure 4.10, the length is 7.

(2) Average width $\bar{\delta}$. For the CNNs, we will treat predicting the average width $\bar{\delta}$ as a regression problem, so the metric is given as positive real numbers. In this example (Figure 4.10), the width $\bar{\delta}$ is 1.14.

$$\bar{\delta} = \frac{1}{l}\sum_{i=I^\star}^{I'}\sum_{j=1}^{W}\mathcal{B}_{ij}.$$

(3) The standard deviation of width, σ_δ. In this example (Figure 4.10), σ_δ is 0.69.

$$\sigma_\delta = \sqrt{\frac{1}{l}\sum_{i=I^\star}^{I'}(\sum_{j=1}^{W}\mathcal{B}_{ij} - \bar{\delta})^2}$$

(4) Continuity: $\mathcal{C}$, we define it as a Boolean and treat it as a binary classification problem. If there exists any row between $I^\star$ and I' such that the row has no track pixel, then this track has a break. In this example (Figure 4.10), the Continuity is *FALSE.*

$$\mathcal{C} = \sum_{i=I^\star}^{I'}((\sum_{j=1}^{W}\mathcal{B}_{ij}) == 0)$$

(5) The average height $\bar{h}$ of the "ridgeline" of the track. Taking the pixel with the maximum height from each row to represent the height of that row, average them as the height of that track. In Figure 4.10, the height $\bar{h}$ is 3.29.

$$\bar{h} = \frac{1}{l}\sum_{i=I^\star}^{I'}\max_j(\mathcal{H}_{ij})$$

(6) The standard deviation of the "ridgeline" height, σ_h. $\bar{h}$ is calculated from (5). In this example (Figure 4.10), σ_h is 1.70.

$$\sigma_h = \sqrt{\frac{1}{l}\sum_{i=I^\star}^{I'}(\max_j(\mathcal{H}_{ij}) - \bar{h})^2}$$

(7) The area of the track pixels, α. In this example (Figure 4.10), the area α is 8.

$$\alpha = \sum_{i=I^\star}^{I'}\sum_{j=1}^{W}\mathcal{B}_{ij}.$$

It can also be calculated by:

$$\alpha = \bar{\delta} l$$

(8) *The average height of background pixels $\bar{\mathcal{H}}$ in the original whole patch without considering track pixels or etching pixels.*

$$\bar{\mathcal{H}} = \frac{1}{HW} \sum_{i=0}^{H} \sum_{j=0}^{W} \mathcal{H}_{ij}$$

4.4 Automated Labeling

Even though all of these calculated metrics can be used as training labels, in the current work, we only focus on the most important two metrics, size and continuity of the track. To be more specific, we train convolutional neural networks to predict the average width and width standard deviation of each track and whether there is a break in each track. The average width can reflect the basic information of the product; the break in the track is common and also a basic type of defect. In figure 4.11, we show examples of our labeling results for tracks.

4.5 Conclusion

The entire image processing and labeling algorithm enables us to label these in-situ videos using automated analysis of ex-situ measurements. In most situations, it is easy to collect in-situ data but difficult to obtain labels for the data. Thus, it is usually difficult to directly use machine learning algorithms for SLM in-situ monitoring. With our proposed methods, we successfully correlate in-situ and ex-situ measurements, then make predictions with machine learning algorithms.

According to what we have described in this chapter, we are able to label tracks from height maps. Since our goal is to train CNNs to recognize the desired metrics from in-situ video rather than ex-situ measured height maps, the last step of labeling is to correlate all these labels to the corresponding videos instead of height maps.

Overall, we generate hundreds of average width and continuity labels via the proposed algorithm. Moreover, we also show that our algorithm can generate not only these two types of labels, but also other metrics as necessary. These labels are then used for training convolutional neural networks, as described in later chapters. With even broader impact, since this approach to generate labeled training sets for in-situ machine learning-based detection algorithms does not rely on specific experimental characteristics, is should be extensible to other additive manufacturing technologies.

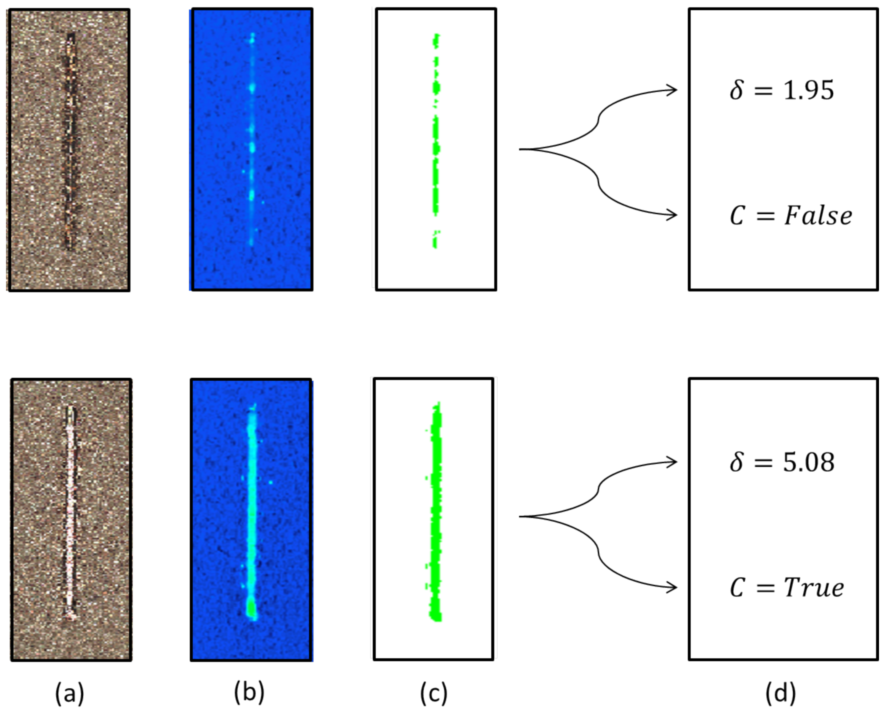


Figure 4.11: *Example tracks:* ***(a)*** *regular electro-optical images,* ***(b)*** *height maps measured by structured light,* ***(c)*** *segmentation results by proposed image processing algorithm,* ***(d)*** *labels from calculated track metrics*

Chapter 5

In-situ Video Monitoring

In this chapter, using the data labels from the previous chapter, an original convolutional network is trained that processes in-situ high speed video data to predict properties of tracks. We demonstrate a proposed machine learning approach to monitoring the SLM manufacturing process that enables on-the-fly assessments of laser track welds. First, in-situ video melt pool data acquired during SLM is labeled according to the (1) average individual track width, $\bar{\delta}$, (2) standard deviation of individual track width, σ_{δ}, and also (3) whether or not the track is continuous, $\mathcal{C}$, measured post-build through the ex-situ height map analysis algorithm. This procedure generates three ground truth labeled dataset for supervised machine learning. Using a portion of the labeled 10-millisecond video clips, a single Convolutional Neural Network architecture is trained to recognize these three distinct laser track metrics. Thus, we also demonstrate using CNNs with the same architecture to perform both regression and classification. This approach should benefit any other SLM system or any additive manufacturing technology, where height-map-derived properties can serve as useful labels for in-situ sensor data.

5.1 Video Preprocessing

Our collaborators at LLNL performed bead-on-plate SLM experiments in which isolated 5 mm track welds are created under a variety of laser speed and power settings using the Aconity 3D SLM system, as diagrammed in Figure 5.1. The raw video data collected are video segments, with each segment corresponding to a single SLM track. The video frames are 256×256 pixels with 1k Hz frame rate; each video segment might contain several hundred frames. Only 12 to around 50 frames, depending on the laser scanning speed, record the laser spot and the scanning process, whereas all the other frames just record the background before or after the laser scanning. Since there are so many unrelated frames without laser spot, the collected raw video data can not be directly used for training CNNs. Thus, we proposed a simple and effective video processing method to remove those extraneous video frames and create a practical video dataset as following:

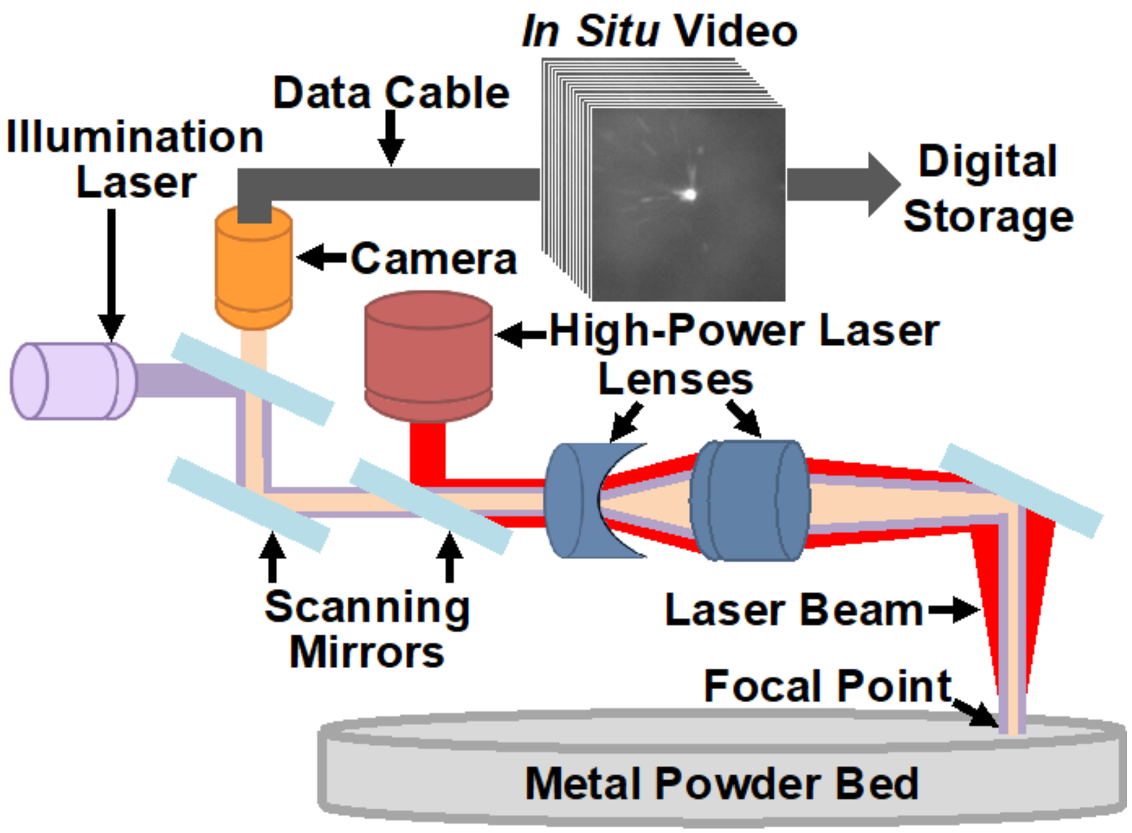


Figure 5.1: *Schematic of video capture system. Adjustable mirrors trace the path of a focal point shared between the co-aligned high-speed camera, illumination laser, and high-power laser. The camera records in-situ video taken during each welding event. [Figure credit: our collaborators at LLNL]*

- Since the laser spot is always roughly in the center of each frame, we set the region of interest (ROI) to be a centered 80×80 box.
- Calculate the average intensity I_{F_i} of all the pixels in each frame F_i.
- Calculate the average intensity I_{ROI_i} of all the pixels in each ROI_i.
- If I_{ROI_i} is larger than I_{F_i}, keep the frame; otherwise, remove the frame.
- Take the longest sequence of consecutive frames as the video data point for training CNNs.

The video dataset contains 870 individual videos, labeled according to the ex-situ height map analysis algorithm that provides each video a label of $\bar{\delta}$ and σ_δ for regression, and track continuity $\mathcal{C}$ for classification. To do this, 700 randomly selected videos are used to

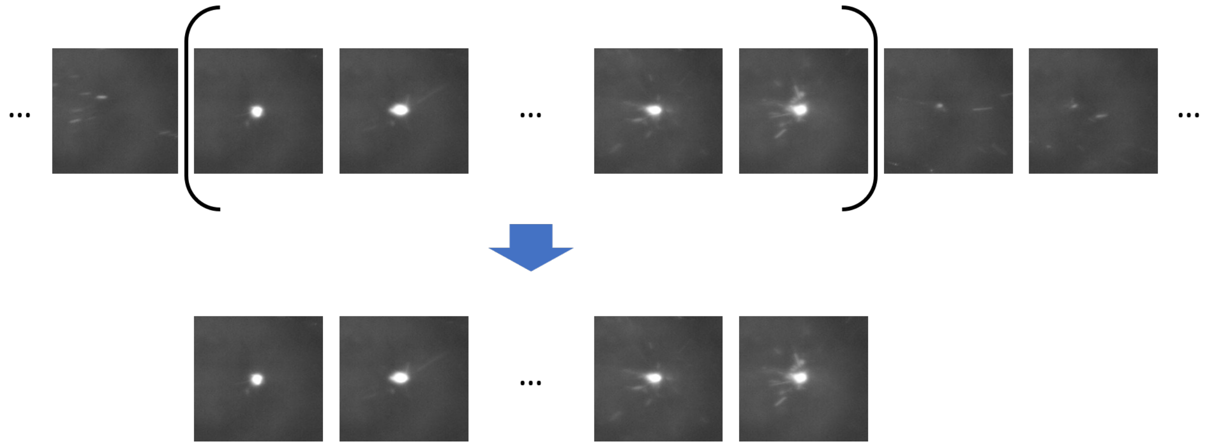


Figure 5.2: *An example of removing video frames which contain no laser spot from raw video data*

train candidate machine learning models and the remaining videos are used to test the fully-trained models. The model architecture was developed using only the $\bar{\delta}_{measured}$ labels, but was trained separately on all three labels, and the learning rate hyperparameter re-tuned. The standard squared difference loss function is used for the regression predictions, while cross entropy loss is used for classification.

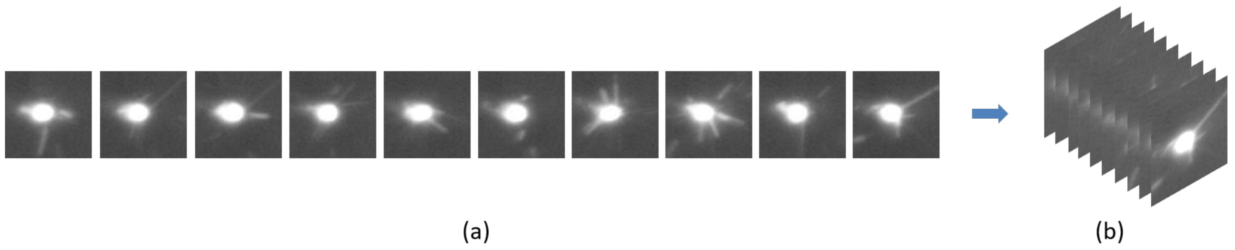


Figure 5.3: *Example of in-situ video data **(a)** ten continuous 64 × 64 frames of in-situ videos **(b)** single data point from concatenation of ten continuous frames*

A convolutional neural network architecture requires videos to be a fixed frame length and resolution. For the fastest scan speed, there are only 12 frames of video and the first and last frames are omitted to ensure end-of-track artifacts did not affect the results. Thus, only the 10 middle frames are chosen from each video, i.e.the middle 10 ms of video. Furthermore, the frames for each track are center-cropped into 64 × 64 pixel size images, eliminating an even larger portion of the background from the videos so that the neural network trains on the most relevant region of the video that encompasses the laser spot. An example of in-situ video data is shown in Figure 5.3.

5.2 Data Analysis

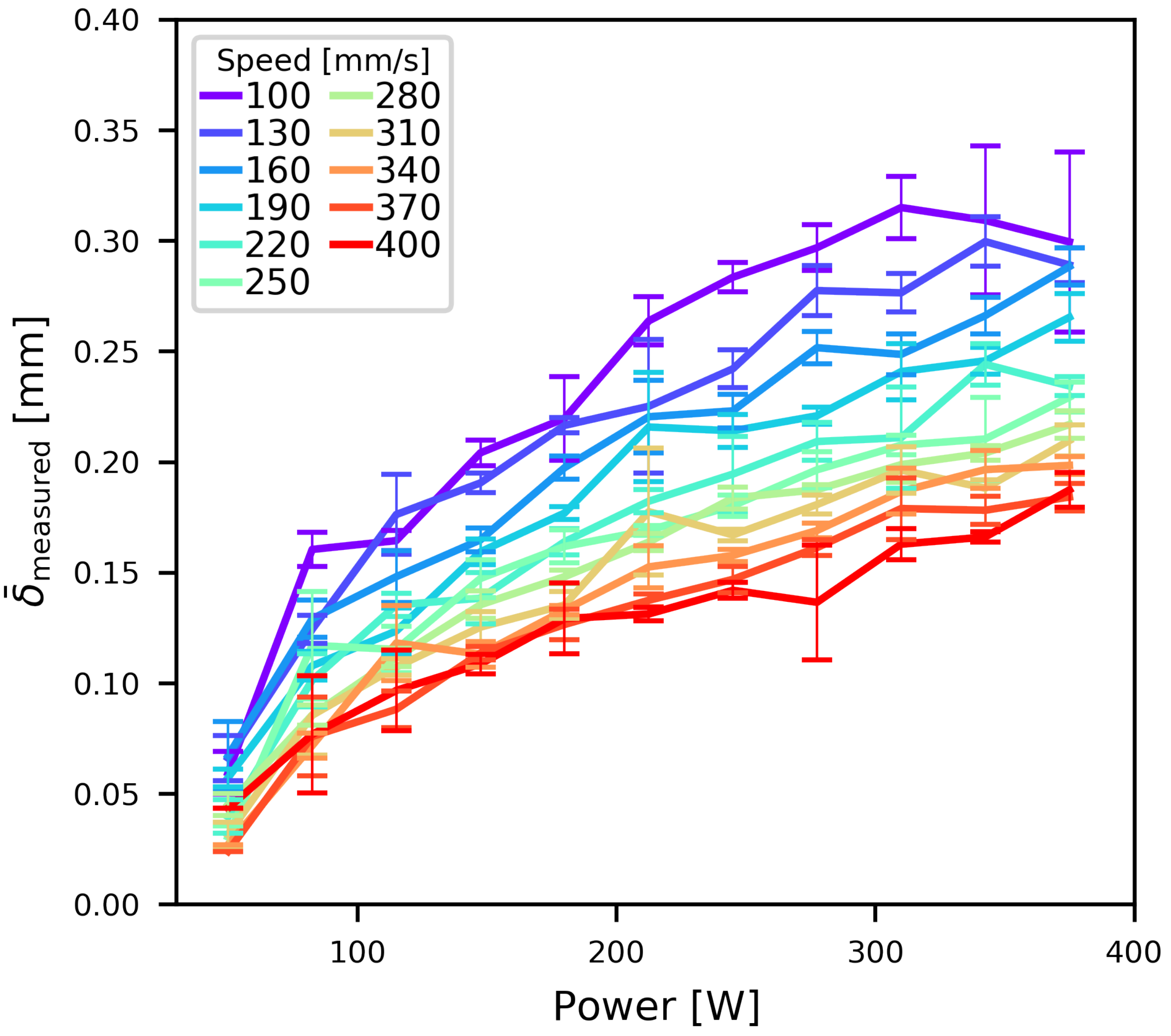
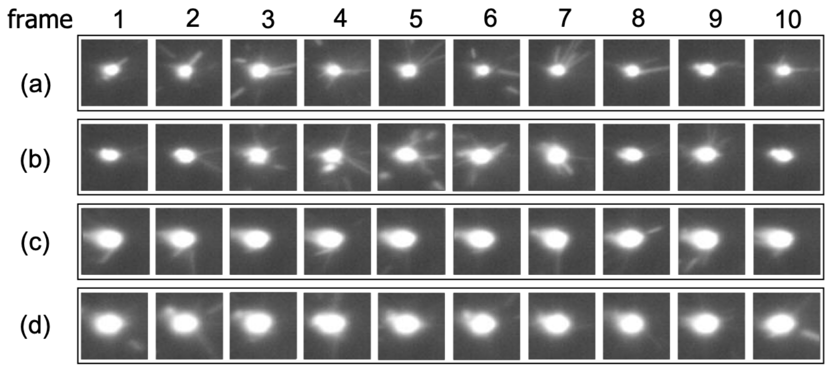


Figure 5.4: *Width measurements in terms of different laser speeds and laser powers.*

After creating and scanning all tracks, the ex-situ height map analysis algorithm (Chapter 4) is used to measure the tracks. Approximately 80% of all tracks are continuous. Figure 5.4 shows measured track widths for all laser speed and power combinations studied, plotted as the average value of $\bar{\delta}_{measured}$ with error bars corresponding to the standard deviation of repeat width measurements. Tracks are widest at high laser powers and slow scan speeds. The trends in Figure 5.4 agree with previous findings from experiments [73, 72, 137, 136] and simulation [59, 59] that show the melt pool width increases with increasing laser power-to-speed ratio, i.e. increasing volumetric energy density. Figure only shows a rough trend of measured track widths, $\bar{\delta}_{measured}$, throughout the dataset collected. However, error bar values suggest no clear function that can accurately calculate track widths with given laser parameters. Thus, an in-situ detection technique based solely on empirical fits of these laser parameters may not reliably capture natural variances within the process.



	$\bar{\delta}_{\text{measured}}$ [mm]	σ_{measured} [mm]	*break*
(a)	0.104	0.0186	true
(b)	0.153	0.0234	false
(c)	0.195	0.0591	false
(d)	0.263	0.0825	false

Figure 5.5: *Examples of different video data points. (Laser conditions: (a). Power* 147.5 *W Speed* 310 *mm/s; (b). Power* 147.5 *W Speed* 160 *mm/s; (c). Power* 342.5 *W Speed* 310 *mm/s; (d). Power* 342.5 *W Speed* 160 *mm/s)*

Track measurements are used as ground truth labels for each video taken during the welding process. Figure 5.5 shows the exact 10 middle center-cropped frames from the in-situ videos and corresponding labels that are used to train the machine learning model for four different example laser conditions in our dataset. The bright area in the center of all video frames corresponds to incandescent light emitted from the melt pool. The video capture system alignment (Figure 1) ensures that the center of the melt pool appears in the same location of every frame. Figure 5.5 displays video segments in order of increasing $\bar{\delta}_{measured}$, in pairs of laser powers, i.e. 147.5 W (a,b) and 342.5 W (c,d), and scan speeds, i.e. 310 mm/s (a,c) and 160 mm/s (b,d). Melt pool size and shape differences between videos at each distinct laser power setting are evident. At larger powers, the melt pool increases in size and aspect ratio, as expected. Intuitively, the relative size of the melt pool appears to correlate with larger values of $\bar{\delta}_{measured}$. Frame-by-frame variations in the quantity of bright pixels also appear to correlate with increasing $\sigma_{\delta_{measured}}$. For instance, there are more saturated pixels in the melt pools at high power in Figure 5.5 (c,d) than at low power Figure 5.5 (a,b). The track width may also correlate with the degree of spatter, which is more pronounced for smaller $\bar{\delta}_{measured}$. It is hard to distinguish from visual inspection alone whether the melt pool is a better indicator of $\sigma_{\delta_{measured}}$ than the spatter. In addition to these features, temporal characteristics gathered from chronological series of frames may yield accurate predictors of resulting track properties. Moreover, there may be additional salient features besides characteristics of the melt pool spatter that correlate strongly with

$\bar{\delta}_{measured}$ and $\sigma_{\delta_{measured}}$. It is unclear from inspection how one would identify track continuity from these video frames.

5.3 Methods

From the post-processed height maps, $\bar{\delta}_{measured}$ and $\sigma_{\delta_{measured}}$ are computed for every track and these are used to assign ground truth labels for the corresponding in-situ videos, thereby assembling a training set for the two regression models. Each track is also assessed for continuity, i.e. a track is discontinuous if at least one gap of non-track pixels exists along its length, otherwise a track is continuous. The continuity labels serve as a training set for a binary classification model. The entire plate (containing around one hundred tracks) is scanned in less than 4 hours after which the ex-situ height map analysis runs in less than 10 seconds. Since a large training set of in-situ video data is amassed relatively quickly, it is feasible to develop machine learning algorithms capable of assessing the in-situ data.

We use a convolutional neural network (CNN) to address this regression problem. The proposed CNN (figure 5.6) has 3 convolutional (conv-) layers with 32 feature maps of 3×3 kernels, 64 feature maps of 3×3 kernels, and 64 feature maps of 3×3 kernels respectively. The first two conv-layers, CONV-1 and CONV-2, are both followed by ReLU [93] activation and a 2×2 mean-pooling layer [79]. The third conv-layer, CONV-3, is followed by ReLU activation. Next a dense layer of 1000 hidden units, FC-1, is fully connected with CONV-3, followed by a dropout layer with 0.5 dropout rate. Then two more fully connected layers, FC-2 and FC-3, with 200 and 20 hidden units respectively are connected with FC-1, and each is followed by sigmoid activation. FC-3 is fully connected with the final output layer of a single unit, which corresponds to the track width value. During the training, we choose the Adam optimizer with a learning rate of 10^{-4}, batch size of 20, and max training epoch number of 300. We apply L2 regularization to all the weights for suppressing over fitting. The regularization coefficient is set as 10^{-4}. Our CNN model is configured and trained with the TensorFlow library using a TITAN X NVIDIA GPU.

To set the number of layers (the depth of the model), an initial architecture was configured with six convolutional layers, which contains sufficient capacity to learn $\bar{\delta}_{measured}$ from the video data. Then the number of layers was reduced sequentially until the model did not exhibit overfitting. (Overfitted models provide excellent predictions for a training set, but do not generalize well to new dataset.) Based on several candidate model training sessions, three convolutional layers could generalize sufficiently to the test data, as discussed below. Although it is more common to use max-pooling to reduce dimensionality between convolutional layers, mean-pooling was found to outperform max-pooling for learning to predict the track width. For this application, mean-pooling may help the CNN learn about the differences in melt pool size to generate predictions for the track width average, $\bar{\delta}$. If this is the case, the overall summation of the video pixel values would seem to be more relevant than the summation of max pixel values. Though the model architecture was not optimized based on track continuity or standard deviation of width measurements, reasonable results were

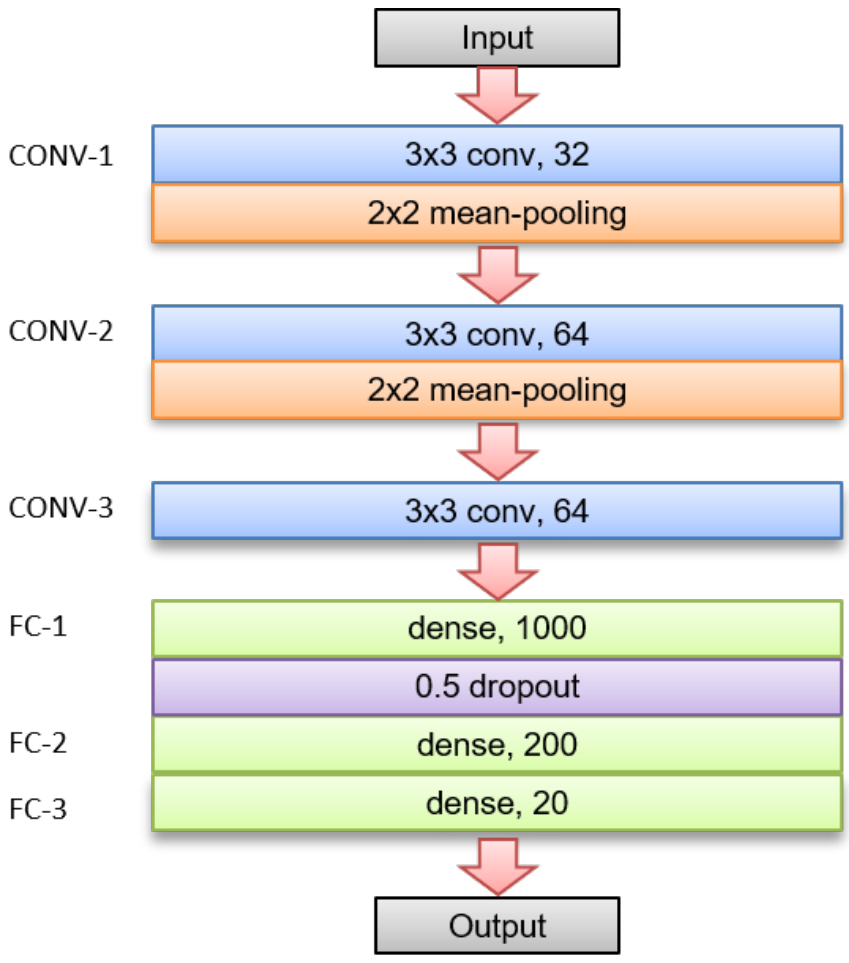


Figure 5.6: *Our proposed CNN architecture*

obtained when retraining the same CNN model (developed with $\bar{\delta}$) using σ_δ and $\mathcal{C}$ labels as discussed below.

5.4 Results and Discussion

Since the relevant, visualized distinguishing features in the in-situ video are not readily obvious, it is not straightforward to decipher the mapping between the video and track properties using only traditional video processing techniques. Rather than manually identifying relevant indicators within the entire in-situ data set, machine learning is used to train our neural network model to learn a suitable mapping between video segments and measured average track width and standard deviation and continuity.

Once trained, the video regression CNN model generates predictions of the average track width, $\bar{\delta}_{predicted}$, standard deviation, $\sigma_{\delta_{predicted}}$, and track continuity, $\mathcal{C}_{predicted}$. The accuracy of track continuity classification is 93.1%. Regression model performance is assessed by comparing measured versus predicted values for both training and test sets in Figure 5.7 and Figure 5.8. A narrow distribution of points around the line of equality (black line) indicates favorably robust model performance, while $\bar{\delta}_{predicted} = \bar{\delta}_{measured}$ or $\sigma_{\delta_{predicted}} = \sigma_{\delta_{measured}}$ for *all* predictions signifies problematic overfitting. The CNN exhibits variance, i.e. training set predictions outperform the test set, as indicated by the tighter distributions of predicted values along the equality line. Furthermore, the CNN model predictions of $\bar{\delta}_{predicted}$ outperform those of $\sigma_{\delta_{predicted}}$ according to the respective coefficients of determination by $R^2_{\bar{\delta}} = 0.93$ and $R^2_{\sigma_\delta} = 0.70$.

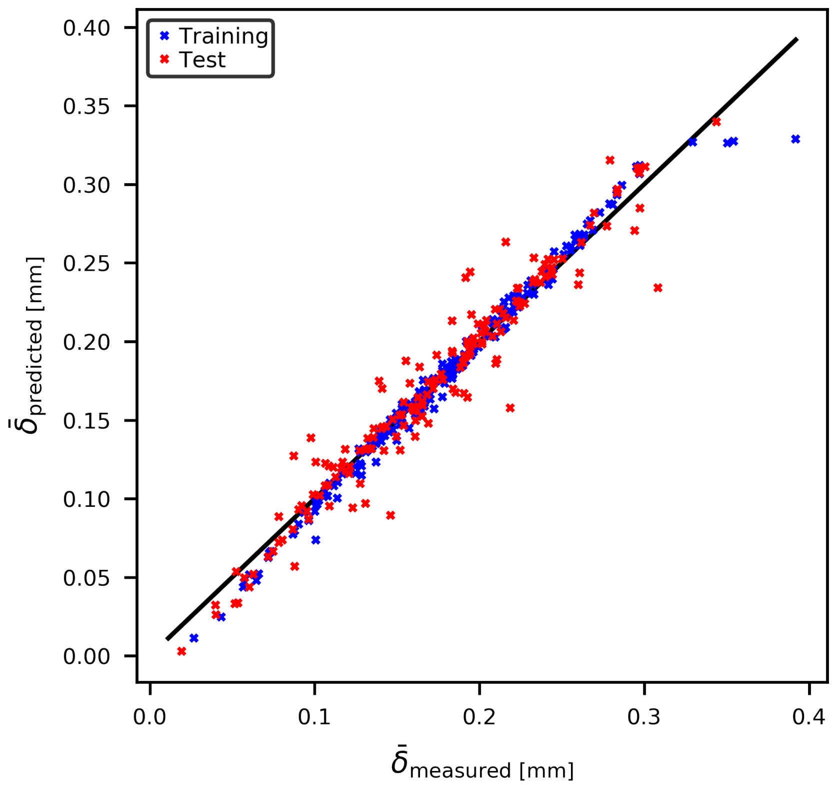


Figure 5.7: *Width prediction results from in-situ videos*

The discrepancy in model performance of predicting average track width versus standard deviation may be due only in part to the fact that the model architecture is developed using width data alone. Given the numerous available choices of model architectures and combinations of hyperparameters, different model configurations than chosen here may result in more accurate $\bar{\delta}_{predicted}$. However, it seems likely that the standard deviation of width is inherently more difficult to predict than the width given the size and/or quality of our

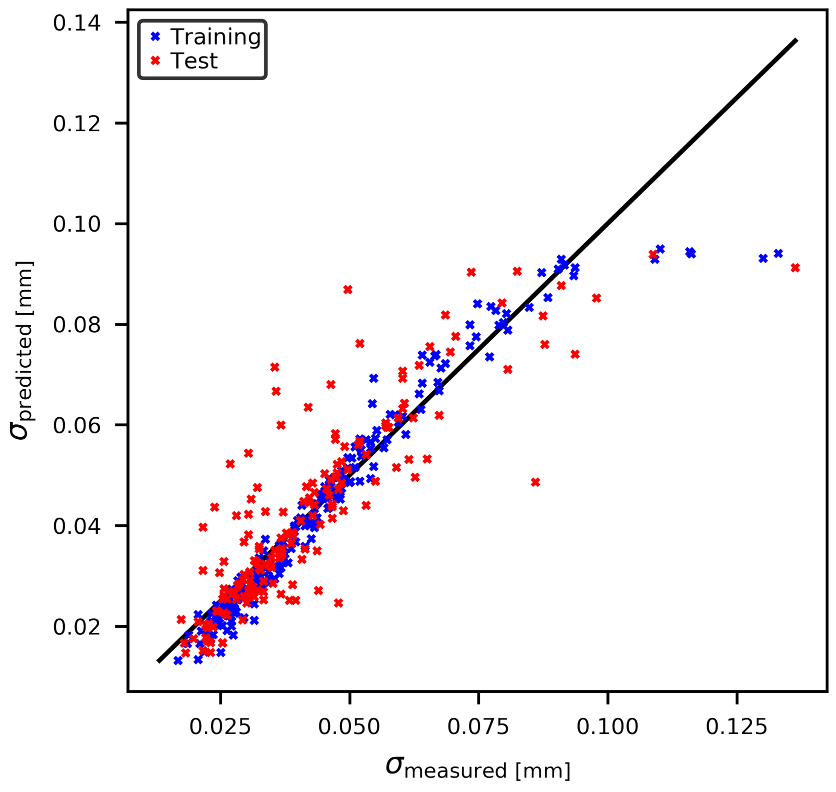


Figure 5.8: *Width SD prediction results from in-situ videos*

dataset. Outliers in Figure 5.8 correspond to the slowest laser scan speeds, 130 and 100 mm/s. Thus, the 10 middle frames (of the 38–50 total frames per video collected at these conditions) the model uses do not contain information of the properties along the entire length of the track. This is a consequence of the CNN architecture, which requires a fixed frame number irrespective of laser operating parameters. Recurrent neural networks that generate predictions from input videos of variable length may help to alleviate this issue. Regardless, for any machine learning model that exhibits high variance, a larger dataset ensures higher quality predictions [47, 7]. Indeed, worse results (not shown) were obtained than in Figure 5 when the model was trained on a small subset of our data. Thus, by following the experimental procedure described here, additional videos can be collected, labeled via the rapid ex-situ algorithm, and used to retrain the CNN to obtain more accurate $\bar{\delta}_{predicted}$, $\sigma_{\delta_{predicted}}$, and $\mathcal{C}_{predicted}$. Nevertheless, using only 10 ms videos, the CNN predicts track SLM widths and continuity and (to a lesser degree) width standard deviation without the need for time-consuming height map derived ex-situ measurements.

Going beyond this work, the training set can be expanded and the machine learning model can be modified to enable improved $\bar{\delta}_{predicted}$ and $\sigma_{\delta_{predicted}}$ or possibly other height map derivable quality metrics, such as surface finish. Moreover, it is worth pursuing whether other ex-situ measurements (e.g., mechanical properties, microstructure, residual stress, part density, etc.) of SLM printed objects are detectable from in-situ data. An important requirement for in-situ detection should involve predicting track properties in cases of multi-scan prints, e.g. parallel adjacent tracks, non-parallel tracks involved in complex strategies, etc. Semi-supervised machine learning algorithms, which will be described in the next chapter, are even more useful and productive since sometimes it is not desirable or possible to label all (or any) in-situ data. Transfer learning techniques may help when ex-situ measurements are difficult to obtain, e.g. x-ray computed tomography, and where complementary physics-based simulations are available for only a subspace of the overall operating regime. While a deeper investigation into the model may reveal something about the features it uses to make predictions, it is unlikely to uncover important characteristics of the underlying physics of the SLM process given the black box nature of CNNs at present. While our current model requires 10 ms video clips, faster detection rates may be possible without compromising prediction accuracy. Machine learning-based models generated with this approach can enable in-situ quality detection and real-time process monitoring essential to rapid closed-loop control.

5.5 Conclusions

A CNN model is developed, trained, and evaluated and shown to be capable of predicting SLM track widths, width standard deviations, and track continuity from in-situ video data alone. Our CNN is trained from as little as 10 video frames with a correlation coefficient of $R^2 = 0.93$ for track width, $R^2 = 0.70$ for standard deviation of track width, and prediction accuracy of 93.1% for track continuity. The algorithm successfully generalizes across multiple tracks created with several combinations of the laser power and speed. Moreover, this approach should benefit any other SLM system or any additive manufacturing technology, where height-map-derived properties can serve as useful labels for in-situ sensor data.

Here, video of SLM tracks is collected using a variety of laser power and scan speed settings; however, it is straightforward to incorporate additional forms of in-situ data, e.g, pyrometer readings, acoustic sensing, etc., which may boost prediction quality. Irrespective of the exact SLM system configuration and chosen operating parameters, in-situ data can be labeled using our ex-situ height map analysis algorithm. After labeling the in-situ dataset with ex-situ measurements, the model is then trained via supervised machine learning that can predict the final properties of SLM track welds on-the-fly. With this approach, it should be possible to label in-situ data via ex-situ measurements for additive manufacturing technologies, e.g. extrusion-based and stereolithographic approaches, other than SLM.

Moreover, even if sometimes our automated labeling approach can not be applied, we can still manually label a small portion of collected data and use semi-supervised learning algorithms to alleviate the problem of insufficient labels. We will introduce a semi-supervised learning approach in the next chapter.

Chapter 6

Semi-supervised Learning

In previous chapters, we described an SLM experimental setup with which video was collected while creating single linear SLM track welds. We labeled these SLM tracks with metrics of interest and used supervised learning to train regression and classification CNN models to predict several metrics from real-time videos. To obtain labels for those in-situ videos, as described previously, height map measurements for each laser printed track were taken using a Keyence VR3000 structured light microscope. However, the structured light machine is expensive and not always available to other individuals or institution, which makes labeling all these in-situ videos become not applicable. Thus, we propose to use semi-supervised learning approach to address this issue, since it only requires a few labeled data points rather than an entire labeled dataset.

In this chapter, we use 2000 SLM in-situ videos, and 700 of them are labeled (randomly selected from the labeled videos described in the previous chapter). Of these 700 labeled videos, the 200 labeled videos are used as the test set and the other 500 labeled videos are used as the semi-supervised training dataset. To recognize these track metrics for SLM in-situ monitoring, we design a semi-supervised in-situ video monitoring framework with convolutional neural networks, which does not require a significant amount of labeled data to train. Note that the data we collect and interpret can also be used for monitoring other metrics, such as track etching, roughness, etc. Our overall framework is easily transferable for monitoring such other metrics as long as a portion of the data is labeled for the metric. Furthermore, our approach is not limited to track width regression or continuity classification.

To our knowledge, we are the first to successfully apply semi-supervised learning to SLM monitoring. Our experimental results demonstrate that a semi-supervised approach is promising and easy-to-implement for SLM monitoring.

6.1 Related Work

The most closely related semi-supervised works to our approach are ladder networks [106] and the temporal ensemble method [75]. The ladder network [106] is a powerful semi-supervised neural network for image classification tasks. The ladder network makes use of both supervised learning with the labeled portion of data and unsupervised learning with unlabeled data simultaneously. The architecture of ladder networks is an autoencoder [51] applied to every layer, not just the inputs. For the labeled data, the ladder network trains in a standard supervised fashion. For the unlabeled data, it trains with an autoencoder in an unsupervised fashion to preserve all the details needed for reconstructing the image. Laine et al., inspired by ladder networks, proposed the temporal ensemble method [75], which is the approach we adapt for our application. The temporal ensemble method can be seen as a simplified version of ladder networks, without the autoencoder. Instead, it makes two passes over the same input with different data augmentation and different dropout [125]. The supervised loss is still the standard cross entropy loss, and the unsupervised loss is the mean square difference between the two passes' output layers. The unsupervised portion helps the model to extract more important features and reduces the effect of overfitting to the small portion of training data that is labeled. It works like an autoencoder but is much more efficient and robust. In our work, we successfully applied temporal ensemble method to our SLM monitoring problem.

6.2 Semi-supervised Approach

Building on previous research on semi-supervised learning algorithms, we make use of the Π model from the Temporal Ensemble method [75] for our CNN model. The Π model has both a supervised training flow and an unsupervised training flow. It is designed for a semi-supervised classification problem. But in our work, rather than only a semi-supervised classification task, we also extend it to a regression problem (predicting track width).

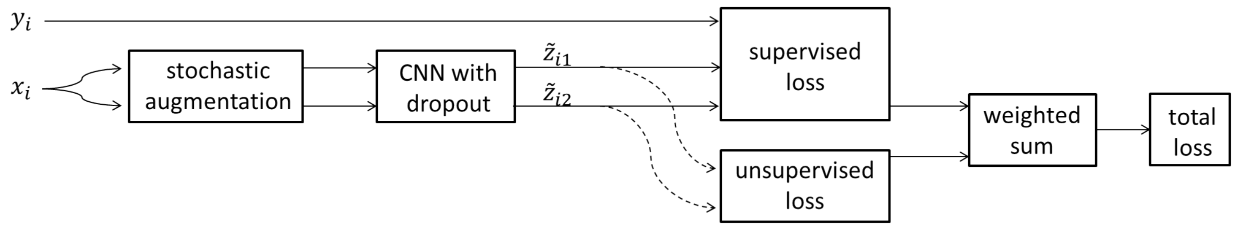


Figure 6.1: *Semi-supervised CNN architecture*

For our problem, the training data $\mathcal{X}$ has N inputs, only M of which have a regression target value (labels $\mathcal{Y}$). L is the set of labeled data indices; for each $i \in L$, x_i has a positive real number label y_i, where $|L| = |\mathcal{Y}| = M$. θ is the set of weights for our convolutional neural networks and the regression value of each data point x_i is $\tilde{z}_i = \theta(x_i)$. During training, data augmentation $\mathcal{A}(x)$ and dropout $\mathcal{D}(x)$ are used, where $\tilde{z}_i = \mathcal{D}(\theta(\mathcal{A}(x_i)))$.

Similar to the Π model, we feed each training data point (video) x_i, whether labeled or unlabeled, into the neural network twice, evaluating it with different data augmentation $\mathcal{A}_1(x), \mathcal{A}_2(x)$, and different dropout $\mathcal{D}_1(x), \mathcal{D}_2(x)$, which will produce different regression outputs $\tilde{z}_{i1}$ and $\tilde{z}_{i2}$, where

$$\tilde{z}_{i1} = \mathcal{D}_1(\theta(\mathcal{A}_1(x_i))), \quad \tilde{z}_{i2} = \mathcal{D}_2(\theta(\mathcal{A}_2(x_i))).$$

The Π model has two passes but only calculates loss from one pass. To make the training more efficient, we instead make the slight change of using both passes' output to calculate the supervised loss $\mathcal{L}_s$ symmetrically (see Figure 6.1). This will take hardly any additional time, since we have to run two passes whether we use the symmetric loss or the original loss, but will work like doubling the training batch size, making our training more efficient overall (it was almost twice as fast in our tests). When we address a regression problem instead of a classification problem, we also make the change of using mean squared difference for the loss instead of cross entropy loss. Specifically, for regression our supervised loss $\mathcal{L}_s$ for each "mini-batch" subset B of data indices is calculated by:

$$\mathcal{L}_s = \frac{1}{|B|} \sum_{i \in (B \cap L)} (\|y_i - \tilde{z}_{i1}\|^2 + \|y_i - \tilde{z}_{i2}\|^2).$$

The unsupervised loss $\mathcal{L}_u$ is still calculated by mean squared difference:

$$\mathcal{L}_u = \frac{1}{|B|} \sum_{i \in B} \|\tilde{z}_{i1} - \tilde{z}_{i2}\|^2.$$

Finally, the total loss $\mathcal{L}$ is the weighted summation of the supervised loss and the unsupervised loss:

$$\mathcal{L} = \mathcal{L}_s + w\mathcal{L}_u,$$

where w is the weight for the unsupervised loss.

We use the same CNN architecture for classification as for regression, but although we use the same symmetric supervised loss, for the classification supervised loss we use cross-entropy loss rather than squared difference:

$$\mathcal{L}_s = -\frac{1}{|B|} \sum_{i \in (B \cap L)} y_i \log(\frac{\tilde{z}_{i1}\tilde{z}_{i2}}{\tilde{z}^\star_{i1}\tilde{z}^\star_{i2}}) + \log(\tilde{z}^\star_{i1}\tilde{z}^\star_{i2}),$$

where $\tilde{z}^\star_{i1} = 1 - \tilde{z}_{i1}$ and $\tilde{z}^\star_{i2} = 1 - \tilde{z}_{i2}$.

We use the same unsupervised loss as for the regression problems, as described above.

6.3 Results and Discussion

We test the semi-supervised learning approach on our collected dataset, consisting of 1000 training data points with 500 labeled and 500 unlabeled, and 200 test data points. Labeled data has two separate labels, average track width (positive real number) and track continuity (Boolean value). We train our model with 500, 100, 50, and 10 labeled data points in both semi-supervised and fully supervised fashion; for all instances of unsupervised learning we also use the entire set of 500 unlabeled videos as additional training data. The 10/50/100 labeled data points are chosen randomly from the full labeled dataset, with the same subset used to train the fully supervised and semi-supervised models we compare. We ran all experiments five times, and report the average performance and the standard deviation. The model architectures are the same (when training the fully supervised model, we just use the supervised loss).

We use the same CNN architecture as described in the previous chapter to address both regression and classification tasks. The CNN has 3 conv-layers and 3 dense layers. The conv-layers have 32 feature maps of 3×3 kernels, 64 feature maps of 3×3 kernels, and 64 feature maps of 3×3 kernels respectively. The first two conv-layers, CONV-1 and CONV-2, are each followed by ReLU activation and a 2×2 mean-pooling layer. The CONV-3 layer is followed by ReLU activation and connected to the first dense layer, FC-1, with 1000 hidden units, followed by a dropout layer with 0.5 dropout rate. FC-2 and FC-3 (each followed by sigmoid activation), with 200 and 20 hidden units respectively, follow the dropout layer. FC-3 is fully connected with the final output layer of a single unit, which corresponds to the relevant track metric as output. We set batch size to 20, and max training epoch number to 300. The regularization coefficient is set as 10^{-4}. We choose the Adam optimizer with a learning rate of 10^{-4} for the regression task, and 10^{-5} for the classification task.

Similar to the original Π model's implementation, we ramped up the unsupervised loss component weight w during the first 50 epochs with a Gaussian ramp-up curve

$$w = W_{max} e^{-5(1-T)},$$

where T varies linearly from 0 to 1 during the ramp-up period and W_{max} is set to 100 for all the experiments.

Each input data video is 10 grayscale frames of 80×80 pixels whose values are in the range $[0, 1]$. With semi-supervised learning approach, we applied standard data augmentation (applying translation, mirroring, brightness and contrast) to all the training data. We crop the frames to 64×64, with pixels shifted vertically and horizontally, by an amount chosen from a uniform random distribution from $[-8, +8]$. We randomly apply horizontal and/or vertical flips. For robustness to changes in lighting conditions (which may even vary from run to run depending on experimental settings), as well as to improve generalizability to other SLM machines, we also performed random brightness and contrast data augmentation for each input video. For brightness, we add a constant chosen randomly from a uniform distribution in the range $[-0.5, 0.5]$ to all pixel values; for contrast, we similarly choose

Number of labeled videos	supervised-only	semi-supervised
500	0.90 (0.02)	0.92 (0.01)
100	0.74 (0.04)	0.81 (0.02)
50	0.56 (0.05)	0.68 (0.03)
10	0.15 (0.07)	0.13 (0.06)

Table 6.1: *Mean R^2 score for average width regression; standard deviations in parentheses*

a uniform random factor c in the range $[0.2, 1.8]$, using it to adjust the contrast $\mathcal{I}_{aug} = c * (\mathcal{I} - \overline{\mathcal{I}}) + \overline{\mathcal{I}}$, where $\mathcal{I}$ is the original image, $\overline{\mathcal{I}}$ is its mean pixel value, and $\mathcal{I}_{aug}$ is the image after augmentation. After these adjustments, we clamp pixel values lower than 0 or larger than 1 to the range limits.

Table 6.1 and Figure 6.2 show our average track width regression experimental results. We report mean and standard deviation of R^2, the coefficient of determination, for randomly selected subsets of 10, 50, 100, and 500 labeled training videos measured on N test data points (in our case, $N = 200$).

Comparing the semi-supervised and fully supervised results, we see that the semi-supervised approach often performs much better than the fully supervised approach when only a small portion of the data is labeled, which demonstrates the viability and data-efficiency of a semi-supervised SLM monitoring system via in-situ videos.

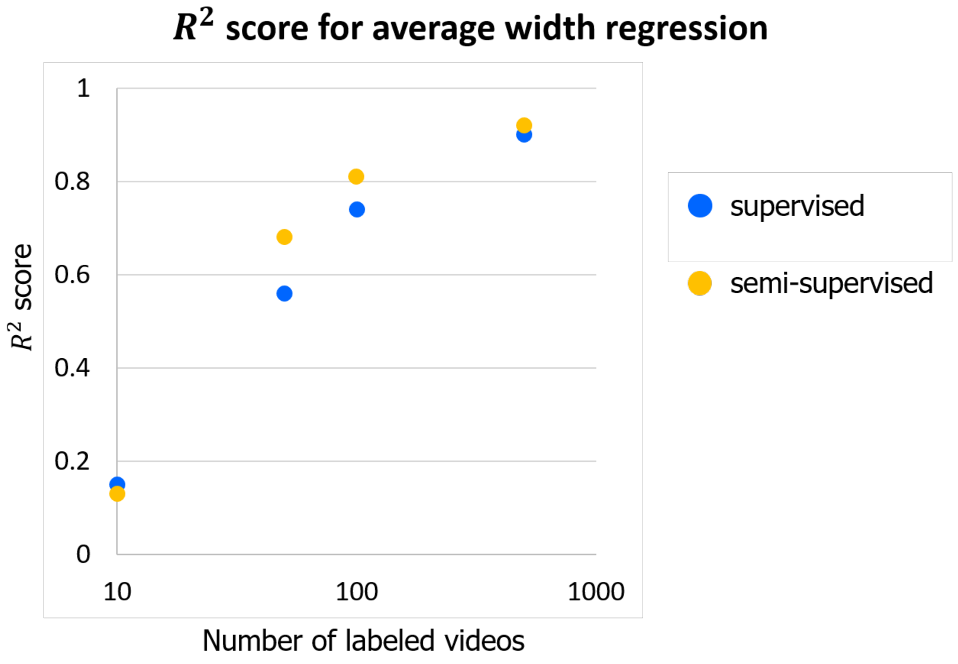


Figure 6.2: *Mean R^2 score for average width regression*

Number of labeled videos	supervised-only	semi-supervised
500	92.2% (0.2)	93.8% (0.2)
100	86.3% (1.1)	92.5% (0.9)
50	85.2% (2.2)	89.8% (1.7)
10	69.1% (4.0)	68.8% (4.5)

Table 6.2: *Mean accuracy for continuity classification; standard deviations in parentheses*

Especially when the number of labeled videos is 50 or 100, the semi-supervised approach significantly improves the performance over supervised, mitigating the disadvantage of having fewer labeled training videos. With only 100 labeled videos, our model can still achieve an R^2 score of 0.81, a 0.07 R^2 score improvement from fully supervised training. Compared to supervised training with five times as many labeled videos, the performance is not much worse. Even with only 50 labeled training videos, we still push the R^2 score from 0.56 to 0.68, which is fairly high for this regression problem. These results imply that people may be able to get away with manually labeling just a small portion of data and training a CNN model in a semi-supervised fashion to address the problem of lack of labels. Moreover, this also demonstrates that semi-supervised learning with temporal ensembles can not only work well for classification, but also work well for regression problems.

For the experiments training with 500 labeled videos, the semi-supervised training result is still better than the fully supervised result, but only slightly. For this dataset, 500 labeled videos is already 50% of the entire training set and it is already enough for supervised training to learn useful features. Thus, the semi-supervised approach does not provide too much improvement. Our semi-supervised approach targets problem when we only have a relatively small amount of labeled data.

However, when we have just 10 labeled training videos, the performance of both semi-supervised and fully supervised models is unsurprisingly poor. With such little labeled data, the training set will be overfit and the model is likely only going to learn very basic and inaccurate features. The R^2 score is close to 0, implying that the model prediction is almost a constant prediction.

Table 6.2 and Figure 6.3 similarly give the mean and standard deviation of prediction accuracy of our binary classification for continuity, also evaluated on the same 200 test data points. About one fifth of the data is "FALSE" for continuity, both for labeled and unlabeled.

The semi-supervised algorithm still performed well for this binary classification problem. It is a remarkable fact that the semi-supervised performance when we have only 100 labeled videos even outperforms the fully supervised performance with 500 labeled videos. Similarly, the semi-supervised performance when we have only 50 labeled videos outperforms the fully supervised one with 100 labeled videos. The semi-supervised approach is particularly advantageous when the amount of labeled data is small.

Overall, our semi-supervised approach performs better than the fully supervised approach no matter if we are solving a regression or classification problem.

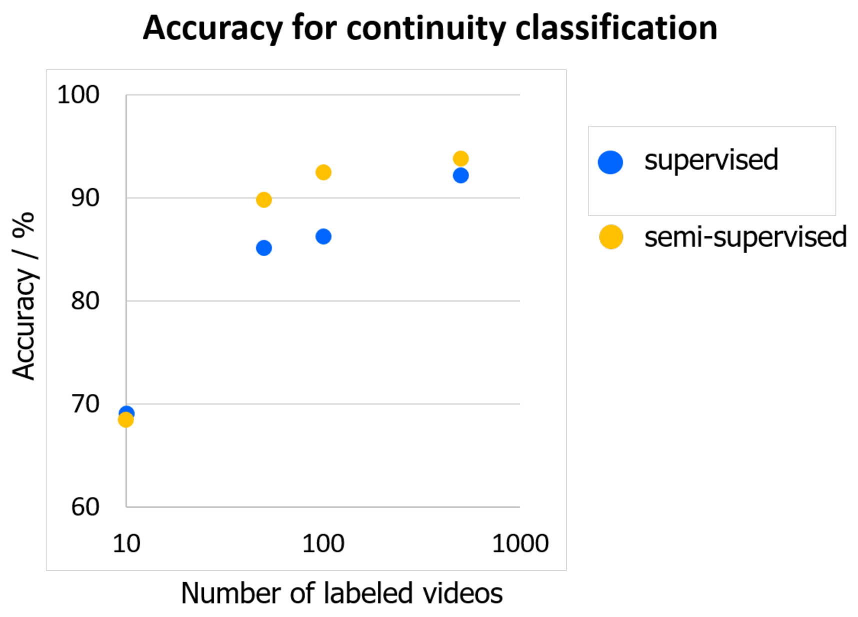


Figure 6.3: *Mean accuracy for continuity classification*

The speed of our trained CNN is more than adequate for real-time monitoring. Measuring its inference time on 100 videos, the average inference time was $1.4ms$, and the maximum was $2.9ms$.

6.4 Conclusion

In this chapter, for around two thousands in-situ videos, we labeled a portion of that data, trained semi-supervised CNN models, and verified our results on a test dataset. We demonstrate that this entire framework enables SLM in-situ monitoring to detect desired metrics in real time. Our framework and the entire approach should be transferable to other SLM systems. To make our approach work well on different systems, it is only required to re-collect video data, measure a small portion of track data and run the track pixel segmentation and classification algorithm, and re-train the model.

We focus on monitoring two characteristics of our experimental tracks, average width and continuity. The experimental results show that our approach recognizes these metrics very well, and also demonstrate that the semi-supervised method requires less labeled data for training.

Besides the metrics discussed in this chapter, this approach can also be easily extended to predict a wider range of defects and metrics, such as etching detection, roughness regression, and even to AM technologies beyond SLM. In this dissertation, we only did monitoring for separated SLM tracks with a single layer. Left to future research is addressing monitoring for SLM with contiguous tracks and multi-layer printing.

Chapter 7

Iterative Cross Learning from Noisy Labels

In the previous chapters, we saw that our deep learning approach shows excellent performance on SLM monitoring. Even though sometimes there are not enough training labels, we can still use a semi-supervised approach to overcome that issue. Besides insufficient training labels, however, another issue is that in many situations (and not only SLM monitoring) it is difficult to get a completely clean data set without extensive effort to manually clean up the data. So in practice, data often has noisy labels, from being manually mislabeled, or mislabeled by inaccurate but fast and cheap automated algorithms, or even because some portion of the data originally without labels is intentionally given random labels because labels are needed for supervised learning. Figure 7.1 shows an example of data sets corrupted with noisy labels.

In this chapter, to address the problem of incorrect labels in training data for deep learning, we study the effect of label noise and propose a novel and simple training strategy, Iterative Cross Learning (ICL), that significantly improves the classification accuracy of neural networks with training data that has noisy labels. ICL is a general strategy, applicable beyond just SLM monitoring. We randomly partition the noisy training data into multiple separate subsets, each of which is used to train an independent network. After these independent networks predict labels for the original data, if the labels agree, we update the label with the predicted result for that data point, but otherwise we update the label with a random label, a key to the success of our proposed method. The process is repeated, possibly with several stages, to gradually improve the performance. Testing our method on MNIST and CIFAR-10 with partially shuffled labels, as shown in Figure 7.1, ICL significantly improves the classification accuracy of existing methods when the data labels have noise, especially in heavy noise situations. Moreover, the proposed method doesn't require any change to the underlying neural networks' structure or loss function, so it can also easily be combined with other existing methods that address noisy labels, improving their performance.

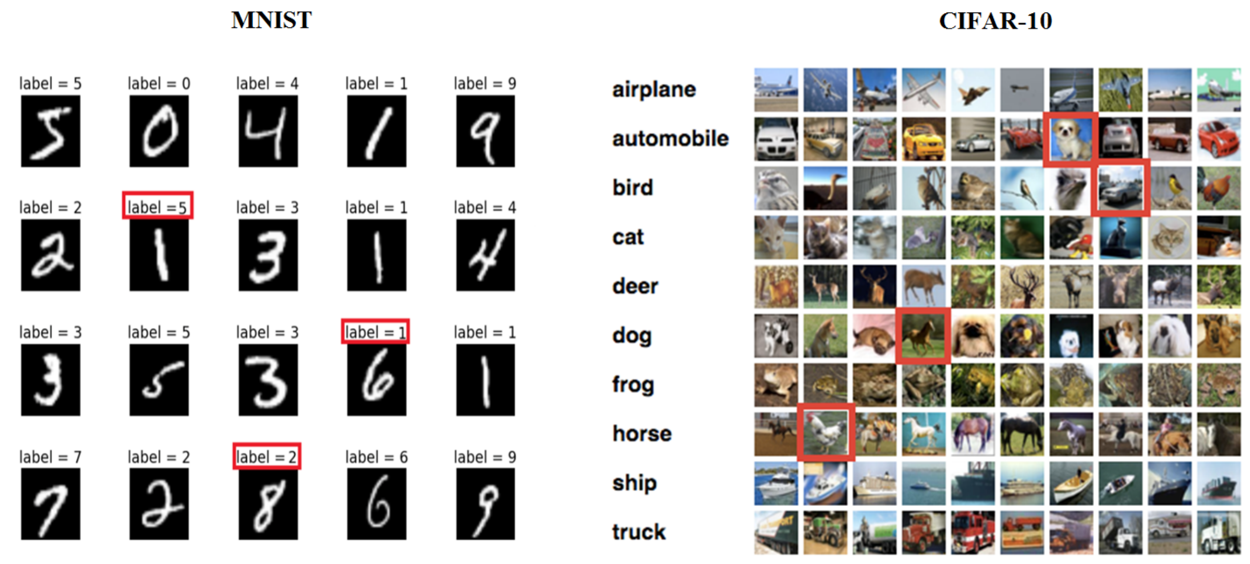


Figure 7.1: *Examples of training data sets corrupted with noisy labels*

7.1 Related Work

Our proposed ICL approach has some common ideas with co-training [14, 97, 133, 82, 70, 90], since both ICL and co-training train multiple classifiers and both are iterative approaches. Co-training for semi-supervised learning requires different views of a dataset, in other words, different features describing the data. It would be ideal for co-training to have two conditionally independent feature sets describing the same data. However, we don't always have that ideal situation, since the two feature sets might not be conditionally independent from each other and multiple feature sets may not be available. Similar to co-training, ICL also makes use of different features, but doesn't require different feature sets. Our proposed method uses a convolutional neural network that learns features [142] by itself rather than manually extracting features like in co-training. We feed our networks different datasets, instead of different features of the same dataset, to make it possible for the different networks to automatically learn some different features. Moreover, co-training's classifier will only gradually generate new labels for data points; it never doubts the labels it has generated, potentially introducing a large bias into the dataset when it outputs incorrect results. Furthermore it cannot correct the labels it has misclassified and cannot address incorrect labels. ICL uses multiple neural networks that learn independently and cross predict labels. It doesn't assume any label is a correct label a priori and it repeatedly updates all the labels in the whole dataset. Thus the features will also be updated after each stage, so ICL tends to have less bias than co-training because the networks are able to correct each other's classifications.

Another hot topic in the deep learning area, generative adversarial nets (GANs) [43, 91, 104, 112], also inspired ICL. The key idea of GANs is to use two separate competitive models, the generator and the discriminator. The competitors compete with each other, gradually learning a better model. In ICL, we also use two separate models, learner 1 and learner 2, but instead of competing, they learn from different content. Then they help each other verify their learning results and identify learning differences to enable more accurate re-learning. Learner 1 and learner 2 in ICL are parallel relations, whereas the generator and the discriminator in GAN are serial relations.

In this chapter, we also compare our approach with some previous work. Sukhbaatar et al. [126] proposed a modification to a convolutional neural network that adds a noise layer to match the noise distribution and achieve the desired performance. Jindal et al. [54] proposed augmenting a standard network with a linear noise model layer at the end that learns the noise distribution; it can be removed after it has helped train the standard network to make accurate predictions in the presence of noise. Meanwhile, since ICL is a general training strategy and doesn't require changing the network structure or the loss function, we can combine ICL with Jindal's appraoch [54] and achieve even better performance when training on noisy labels.

7.2 Label Noise

For a k-classification problem, we will denote a training data point (x_l, y_l), where y_l is the correct label for x_l, $(y_l \in 1, 2, ..., k)$, where k is the total number of classes. However, in practice, some of the training data points might accidentally be (x_l, y_l^*), where the label y_l^* also belongs to $1, 2, ..., k$ but $y_l^* \neq y_l$. For example, in the CIFAR-10 dataset, a dog image might be mistakenly labeled as "cat" or "frog," as shown in Figure 7.1. We call y_l^* a "noisy label."

In practice, noise may also be in the form of non-labeled data points. For the convenience of supervised learning, we can give a random label $1, 2, ..., k$ to each such non-labeled data point.

In this chapter, we compare the performance of existing CNN models trained with and without using ICL on noisy labels. We use uniform noise for our experiments, because the repeatability of non-uniform noise experiments is low. To denote the amount of noise, we use "noise level" η for the fraction of labels "flipped." For a given data point (x_l, y_l), there are two ways to flip its label. We can pick a random label from $1, 2, ..., k$ for y_l^* while ensuring that $y_l \neq y_l^*$, or we can just randomly pick a label from $1, 2, ..., k$ for y_l^*, which means there is a $1/k$ chance that $y_l = y_l^*$. We choose the latter for reporting our results to be consistent with [54]. Using confusion matrix M to represent the noise distribution, each element p_{ij} in M represents the probability of a member of class i being labeled as class j. So for a clean dataset, M will be an identity matrix. For the uniform noise model with noise level η, $p_{ii} = 1 - \eta + \eta/k$ and $p_{ij} = \eta/k$ $(i \neq j)$.

Since ICL is based on the assumption that the neural network can learn useful information even with noisy labels, let us consider their effect. For a k-classification problem with uniform noise level η, each class i will have expected fraction $(1-\eta+\eta/k)$ correct labels, and expected fraction $(\eta - \eta/k) = (\eta/k) * (k-1)$ labels incorrectly set, distributed equally between labels $1, 2, ..., i-1, i+1, ..., k$. Let us consider the relationship between the noise in two classes i and j $(i \neq j)$. There will be fraction $1 - \eta + \eta/k$ of class i correctly labeled, with fraction η/k incorrectly labeled as j. Symmetrically, class j has fraction $1 - \eta + \eta/k$ correct labels and η/k incorrectly labeled as i. Between these two classes, the proportion P of correct to incorrect labels is

$$P = \frac{1 - \eta + \eta/k}{\eta/k} = \frac{1-\eta}{\eta/k} + 1 \ .$$

7.3 Iterative Cross Learning (ICL)

Sometimes even if the labels are noisy, existing CNN models can achieve an error rate lower than the noise level (this is confirmed in our experimental results). Thus, we can use the trained model to re-predict labels to make the labels less noisy. Based on those new labels we predict, if we are still able to train a model with an error rate lower than the current labels' noise level, then we can repeat this process to make the data less and less noisy.

As summarized in Figure 7.2, the key idea of Iterative Cross Learning is to train independent convolutional neural networks from different data, enabling these independent networks to clean up each other's data for the next learning stage. In this thesis, for simplicity we will describe the algorithm for just two independent networks. For a given dataset S with noisy labels, we shuffle and partition the noisy data into two separate training datasets S_1 and S_2 with the same number of data points. Though these two datasets will tend to have similar noise levels and similar noise distributions, the data points in the two datasets will be different.

Two convolutional networks C_1 and C_2 (which need not have the same structure, though for the currently reported results we use the same structure for convenience) will be trained on datasets S_1 and S_2 independently (the inner training loops in Figure 7.2). Even with the same initial data and the same weight initialization, trained networks will not end up with the same weights if the input order is shuffled. Since the two networks will have different randomly initialized weights and are trained on different data, the parameters they learn will definitely be different from each other. The classifier C_1 might be a little better at classifying class i and the classifier C_2 might be a little better at classifying class j. ICL exploits this difference (the outer training loop in Figure 7.2).

When we do the training with a noisy dataset, the performance might decrease after some inner training loop epoch because the network overfits noisy labels. So it is better to monitor the performance during the training and to decide which epoch's model to be used based on the monitoring performance. In some situations, besides the noisy training dataset S, we may have a clean set S_m without noisy labels that can be used to monitor training. For example, in practice, we may expend some human effort to manually clean up a small portion of the data. Since the training labels cannot be trusted, the "training accuracy" and "training loss" calculated from these noisy labels won't be particularly useful.

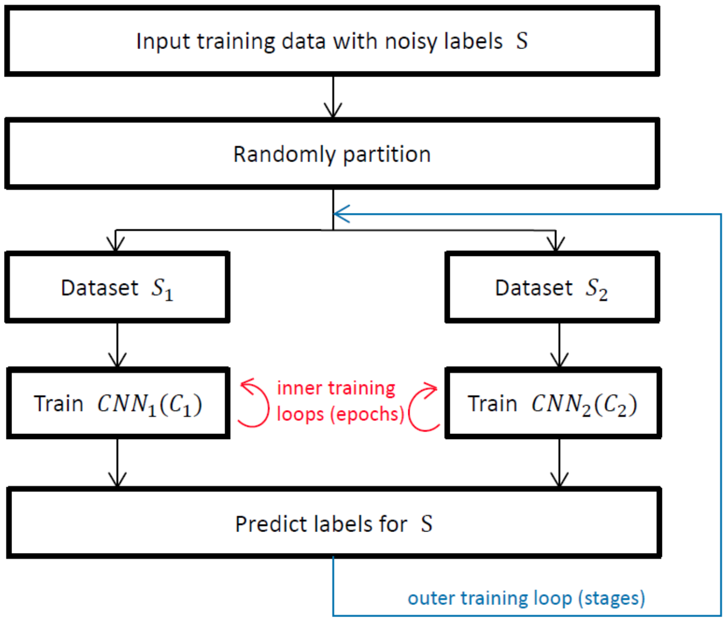


Figure 7.2: *ICL flow diagram*

Therefore, during training, we use instead the monitoring set, which is more trustworthy than the training set, to monitor the performance of the two networks C_1 and C_2. For each network, the model with the best performance on monitoring set S_m during the training process will be used as the training result to predict labels.

However, clean data is not always available or easy to obtain. If we don't have any clean data, we can still use our strategy without monitoring on a monitoring set. Training will just be terminated at a given training epoch number. Then we will use the trained model to predict labels for the dataset. Compared to monitoring on clean data, this will give slightly worse results, since we are not able to determine if a model from an earlier epoch might have been the model with the best performance on the clean dataset. (Algorithm 2 and Algorithm 3 provide the pseudo code for ICL with and without a monitoring set, respectively.)

After training, we use the two networks independently to predict labels for the original training dataset S. If the labels predicted by the two networks are the same, we set the data's label to match the prediction. Otherwise, we update the label with a random label. This technique will make the noise from incorrect predictions more uniformly random and less structured, which can give the trained networks less bias. The two networks learn from their different input at first, but then they also learn from each other, verifying each other's learning results and pointing out learning disagreements to re-learn; hence the name "cross learning."

Algorithm 2 Iterative Cross Learning with monitoring set

```
1: $S \leftarrow$ training set with noisy labels
2: $S_m \leftarrow$ monitoring set with clean labels
3: $S_1, S_2 \leftarrow$ randomly partition $S$ into two separated datasets
4: initialize weights of CNNs $C_1$ and $C_2$
5: initialize Accuracies $Acc_{c1}$, $Acc_{c2}$, $Acc$ and $Acc'$ to 0
6: **repeat**
7:     initialize weights of CNNs $C_1'$ and $C_2'$
8:     $C_1' \leftarrow C_1'$ trained on $S_1$ with monitoring on $S_m$
9:     $C_2' \leftarrow C_2'$ trained on $S_2$ with monitoring on $S_m$
10:    $Acc_{c1} \leftarrow C_1'$'s accuracy on $S_m$
11:    $Acc_{c2} \leftarrow C_2'$'s accuracy on $S_m$
12:    $Acc' \leftarrow \max(Acc_{c1}, Acc_{c2})$
13:    **if** $Acc' <= Acc$ **then**
14:        **return** the more accurate of $C_1$ and $C_2$
15:    **end if**
16:    $C_1 \leftarrow C_1'$
17:    $C_2 \leftarrow C_2'$
18:    $Acc \leftarrow Acc'$
19:    $L_{C_1} \leftarrow$ labels predicted for $S$ by $C_1$
20:    $L_{C_2} \leftarrow$ labels predicted for $S$ by $C_2$
21:    **for** data point $x$ in $S$ **do**
22:        **if** $x$'s label $l$ is the same in $L_{C_1}$ and $L_{C_2}$ **then**
23:            set $x$'s label to $l$
24:        **else**
25:            set $x$'s label to a random label $l \in \{1, 2, ..., k\}$
26:        **end if**
27:    **end for**
28: **until** the max stage number // e.g. 10
29: **return** the more accurate of $C_1$ and $C_2$
```

With a clean monitoring set, we could repeat the whole process until the change in monitoring accuracy between successive stages falls below some user-defined threshold. However, based on our experimental results, we found that most improvements are made in the first several stages, even when it took many more stages for ultimate convergence. So in practice, we terminate the training process if the monitoring accuracy does not increase after a new stage. (Such a decrease in accuracy suggests that the noise level from the two classifiers' prediction is already close to the noise level of the training data.) Without a clean monitoring set, we just repeat the process with a given maximum number of stages. Based on our experiments, depending on how challenging the input is, without a monitoring set the actual error rate may actually get worse again subsequent to the first round of retraining.

Therefore we just set the max stage number to 1 (one round of retraining after updating the noisy labels based on the round zero results). Every time we update a data point with a random label, it brings random noise into the system such that some labels that were previously incorrectly predicted may get a chance to flip and help the networks recover from errors caused by misleading labels. In other words, this random flipping process enables networks to rectify some previous incorrect predictions.

Algorithm 3 Iterative Cross Learning without monitoring set

```
1: $S \leftarrow$ training set with noisy labels
2: $S_1, S_2 \leftarrow$ randomly partition $S$ into two separated datasets
3: **repeat**
4:    initialize weights of CNNs $C_1$ and $C_2$
5:    $C_1 \leftarrow C_1$ trained on $S_1$ with max epoch
6:    $C_2 \leftarrow C_2$ trained on $S_2$ with max epoch
7:    $L_{C_1} \leftarrow$ labels predicted for $S$ by $C_1$
8:    $L_{C_2} \leftarrow$ labels predicted for $S$ by $C_2$
9:    **for** data point $x$ in $S$ **do**
10:       **if** $x$'s label $l$ is the same in $L_{C_1}$ and $L_{C_2}$ **then**
11:          set $x$'s label to $l$
12:       **else**
13:          set $x$'s label to a random label $l \in \{1, 2, ..., k\}$
14:       **end if**
15:    **end for**
16: **until** the max stage number // zero-indexed;
                              // we use 1 as max
17: **return** $C_1$ or $C_2$
```

When we finish the entire training process, we need to choose one model from C_1 and C_2. If we have a clean monitoring set S_m, we choose the one that has better performance on S_m. Otherwise, we just randomly pick one as the final model.

We found that it tends to be slightly better if we only partition the data once for all the stages rather than randomly re-partition in every stage. We speculate the reason could be that some data points may be easier for learning from and some may be harder. Thus the one-time random partition may make the two datasets S_1 and S_2 have slightly different difficulty for classification, which means one dataset could make it easier to train a better classifier from that dataset. Compared to this, repeating random partitioning will eventually on average have two classifiers learn from an average difficulty dataset. So the training result might tend to be a little bit worse than using one-time random partitioning.

Note that the best we can achieve is that we perfectly predict the labels for all the data points in the original training dataset and the performance should be the same as training with clean data. However, in practice, we may not perfectly predict labels, so performance training with noisy data should always be worse than training with clean data.

7.4 Experiments and Results

We test our method on two popular datasets (MNIST and CIFAR-10) using TensorFlow, with uniform noise added. To be consistent with how noise level is measured in the previous work to which we compare our results, we use the same methodology and terminology as Jindal et al. [54]. Since both of the two datasets have 10 different labels $0, 1, 2..., 9$, for a given "noise level" η, we randomly choose η portion of the data and set a random label from 0 to 9 for each. For each dataset, we do experiments with adding 30%, 50% and 70% "noise" to training labels, respectively. (Note that because there are 10 possible labels, 10% of the time the "noise" will just be the original, correct label.)

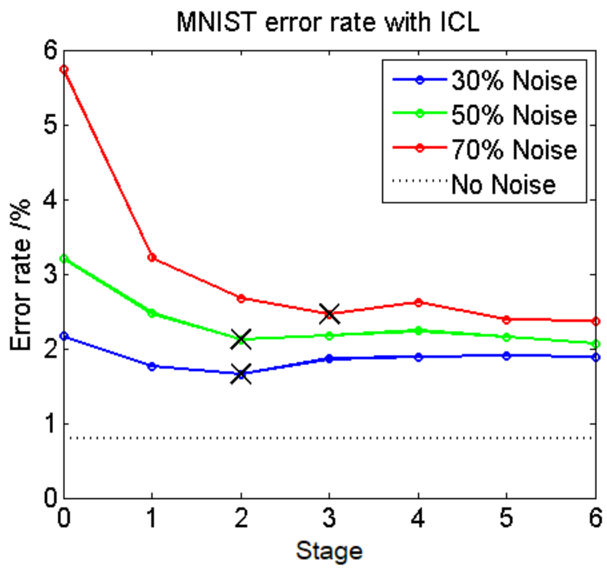
Using ICL, the original training labels will be updated in each stage. The training data will become cleaner and cleaner, which should decrease the error rate. So during the ICL process, the gap between the prediction error and noise level will become smaller and smaller. The improvement after the first several stages will also be small. Finally, after several stages, the error rate tends to be close to the noise level and the monitoring accuracy stops increasing. After the best performance stage, the accuracy starts to fluctuate. ICL's improvement after the first stage is always the greatest.

For each dataset, we compare the performance of the base CNN model, the true noise linear model in [126], and the dropout regularization noise model using [54] to ICL, both ICL combined with the base CNN model and ICL combined with the dropout regularization noise model.

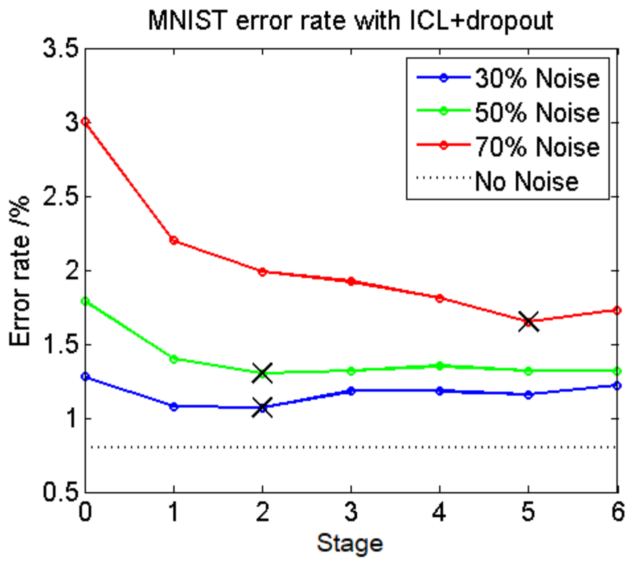
7.4.1 MNIST with clean data for monitoring

The MNIST dataset is a set of handwritten digit images [76]. We use its 50,000 image training set, and randomly divide the other 10,000 images into 5,000 images for monitoring and 5,000 images for testing. Each image has a dimension of 28×28. We use a base CNN that has 2 conv-layers with 32 feature maps of 5×5 kernels and 64 feature maps of 5×5 kernels respectively, and each conv-layer is followed by ReLU activation and a 2×2 max-pooling layer. Next a dense layer of 1024 hidden units is fully connected with the second convolutional layer, followed by ReLU and dropout rate of 0.5, and fully connected with the final output layer of 10 units. We choose the Adam optimizer [64] and a learning rate of 1.0×10^{-4}.

Figure 7.3 and Table 7.1 show our experimental results. Using ICL with the base CNN, the error rate decreases in the first several stages. Stage 0 is the baseline performance without using ICL. For 30% and 50% noise, the model achieves the best performance at stage 2. For 70% noise, the model achieves the best performance at stage 6. But in practice, we would terminate ICL when we find the performance gets worse at stage 4, returning the model from stage 3; its performance is already quite close to the best performance of stage 6. Combining ICL with the dropout regularization noise model [54], the best performance is achieved at stage 2 for 30% and 50% noise, and at stage 5 for 70% noise.



(a) ICL + base CNN



(b) ICL + dropout

Figure 7.3: *ICL on MNIST with clean data for monitoring ("X" marks the stage returned by the algorithm).*

On the MNIST dataset, we find that the accuracy stops increasing significantly after no more than five stages (Figure 7.3). (For comparison, we plot the results for a full six stages.)

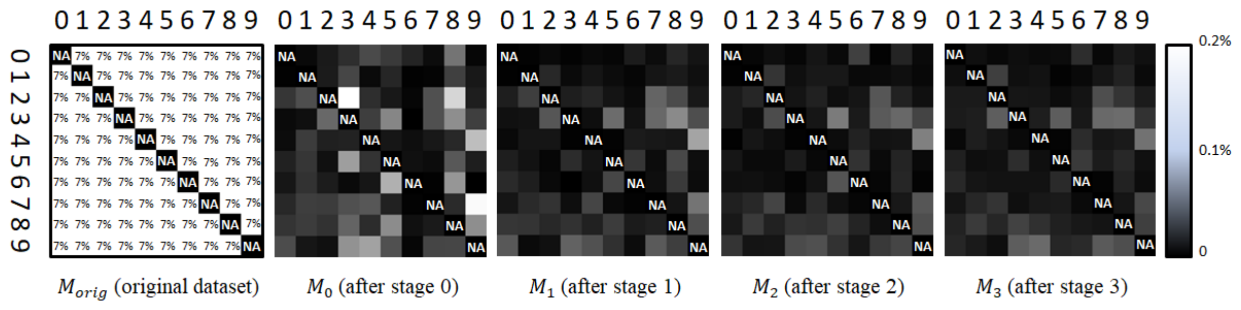


Figure 7.4: *Confusion matrices (base CNN with ICL training on* 70% *noise in MNIST dataset with clean data monitoring)*

In practice, we stop as long as the accuracy stops increasing after a new stage, and return the prior model, which had the best performance.

We also visualize confusion matrices M after each training stage. We give one example in Figure 7.4 of 70% noise to show how the noise distribution changes over training. Note: for clearer visualization, we omit all the diagonal matrix elements (marked NA); otherwise normalizing the confusion matrices will make the intensity of diagonal elements so strong that the other elements will all appear to be equally dark.

From these visualization results we can see that the number of misclassified labels is less and less over training, which is also consistent with the decreasing error rate in the first several ICL training stages. In addition, we can see that the elements of the confusion matrix after stage 0 have the most contrast with each other, which means that labels predicted at that time also have the most structured noise. As seen in the visualizations of later stages in Figure 7.4, our method continues to make improvements at these stages even with structured noise.

"Noise level"	30%	50%	70%
Base CNN	2.17	3.2	5.75
True noise[1]	1.3	2.06	3.31
Dropout[2]	1.25 (1.2)	1.8 (1.92)	3.01 (3.12)
Base CNN + ICL[3]	**1.66**	**2.12**	**2.54**
Dropout + ICL[3]	**1.07**	**1.32**	**1.78**

Table 7.1: *Error rates % for MNIST with clean data for monitoring; "noise level" is the percent of labels randomly reassigned.*

[1]Results using True Noise model as reported in Sukhbaatar et al. 2014 [126]. Since Jindal's model [54] already beats the true noise model, we don't re-implement the true noise model to combine with ICL.

[2]Model proposed by Jindal et al. 2016 [54]. Values out of parentheses are the results we measured using the methods in [54], and the values in parentheses are the results they reported in their paper.

[3]The numbers in bold are results from our algorithm.

No matter which method it is combined with, ICL improves performance (see Table 7.1). Combined with the dropout regularization noise model, ICL achieves the best performance. The higher the noise level, the higher relative performance improvement we see with ICL.

7.4.2 MNIST without clean data for monitoring

Sometimes clean data may not be available for monitoring, so we also test our algorithm without clean data. We used the same 50,000 images for training and 5,000 images for testing. The max epoch number was set to 20. Results are shown in Figure 7.5 and Table 7.2.

Unlike ICL with clean data, we don't have a monitoring set for monitoring each stage, so we just perform one stage of ICL (i.e. max stages set to 1). The performance is a little bit worse than ICL with clean data for monitoring, which is to be expected.

"Noise level"	30%	50%	70%
Base CNN	2.29	3.76	6.60
True noise[1]	1.3	2.06	3.31
Dropout[2]	1.29 (1.2)	2.12 (1.92)	3.96 (3.12)
Base CNN + ICL[3]	**2.00**	**2.54**	**3.58**
Dropout + ICL[3]	**1.17**	**1.50**	**2.34**

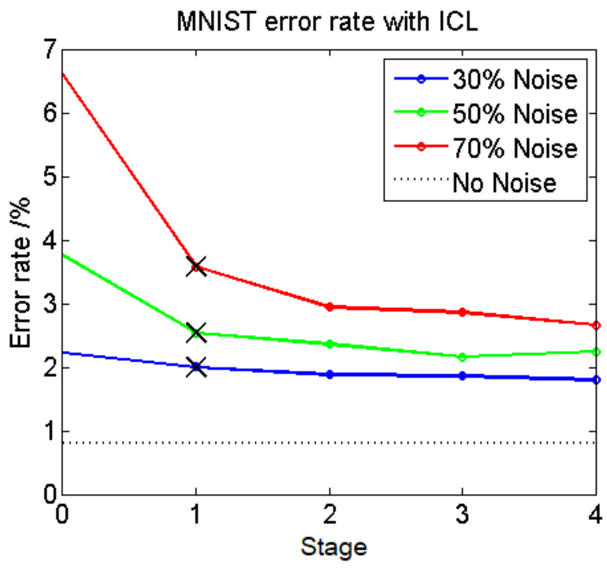
Table 7.2: *Error rates % for MNIST without clean data; "noise level" is the percent of labels randomly reassigned.*

7.4.3 CIFAR-10 with clean data for monitoring

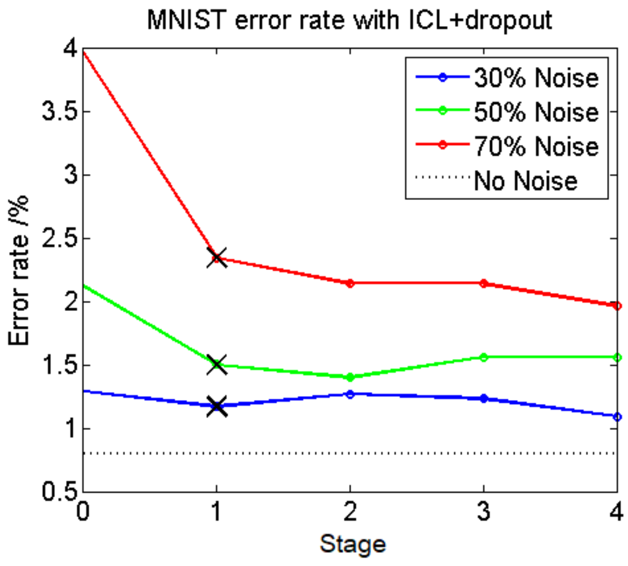
The CIFAR-10 dataset is a set of tiny color images of dimension $32 \times 32 \times 3$ [66]. We use its 50,000 training images, and randomly divided the other 10,000 images into 5,000 images for monitoring and 5,000 images for testing. Classical data augmentation techniques (random crop, random flip, random brightness and random contrast) are used for the training. We use a similar base CNN model as described above, except that the size of the max-pooling layer is 3×3 and the number of hidden units is 200.

Figure 7.6 and Table 7.3 show experimental results. Using ICL alone on the base CNN, for a 30% noise level, ICL achieves the best performance at stage 1. For a 50% noise level, it achieves the best performance at stage 3, while for a 70% noise level, it achieves the best performance at stage 4. Using ICL with the dropout regularization noise model [54], for 30% and 50% noise level, it achieves the best performance at stage 1. For a 70% noise level, it achieves the best performance at stage 2.

Again, using ICL improved the performance of both the base CNN and dropout regularization (see Table 7.3). However, unlike the results on MNIST, on CIFAR-10 the improvement at 30% noise is fairly small (the base CNN achieves a 25.1% error rate and ICL improves it to 24.9%; the dropout model achieves a 24.1% error rate and ICL improves it



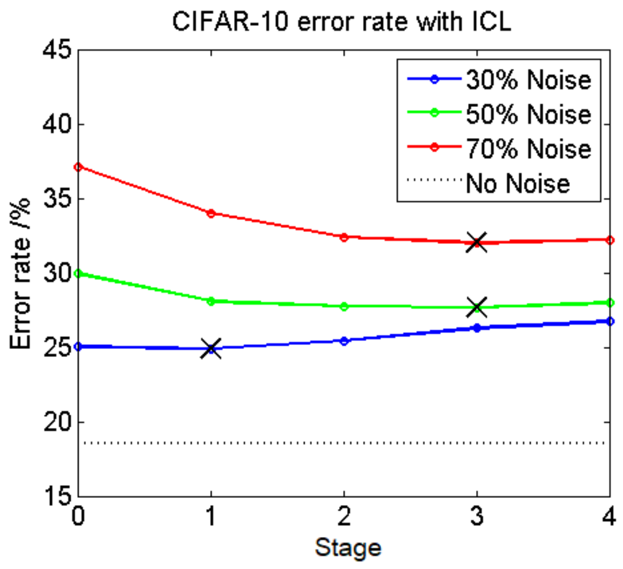
(a) ICL + base CNN



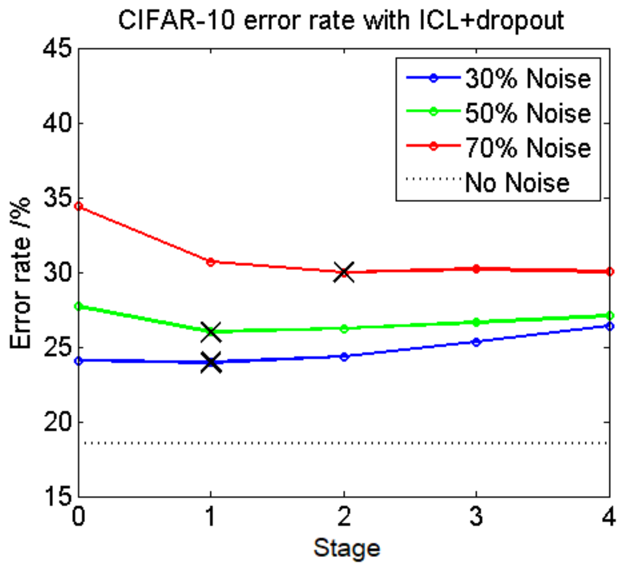
(b) ICL + dropout

Figure 7.5: *ICL on MNIST without clean data for monitoring ("X" marks the stage returned by the algorithm; it is set to be 1 if we don't have clean data for monitoring).*

to 23.9%). The original base CNN, when trained on the uncorrupted 0% noise CIFAR-10 dataset, already has a 18.5% error rate, which is much higher than the corresponding 0.8% error rate on the uncorrupted MNIST dataset. When we train the model with 30% noise, it has around a 25% error rate, which is already very close to the 30% noise level. In this case, there is not much additional useful information that ICL can use for improvement with 30% noise.



(a) ICL + base CNN



(b) ICL + dropout

Figure 7.6: *ICL on CIFAR-10 with clean data for monitoring ("X" marks the stage returned by the algorithm).*

Moreover, after the first stage of ICL, in such circumstances performance may only deteriorate in subsequent stages. Though we terminate the ICL process in practice if we find the performance does not improve after a new stage, for our experiments we ran ICL for four or more stages. We found that the performance with 30% noise actually would get worse and worse after the first stage on the CIFAR-10 dataset. If the error rate is close to the noise level, when we update the labels, we actually will bring more and more noise into the labels.

"Noise level"	30%	50%	70%
Base CNN	25.1	29.9	37.1
True noise[1]	24.8	29.6	36.2
Dropout[2]	24.1 (24.4)	27.7 (32.6)	34.4 (33.0)
Base CNN + ICL[3]	**24.9**	**27.6**	**32.0**
Dropout + ICL[3]	**23.9**	**26.0**	**30.0**

Table 7.3: *Error rates % for CIFAR-10 with clean data for monitoring; "noise level" is the percent of labels randomly reassigned.*

7.4.4 CIFAR-10 without clean data for monitoring

We also tested our algorithm without clean data on the CIFAR-10 dataset, using the same 50,000 images for training and 5,000 images for testing (Figure 7.7 and Table 7.4). The max epoch number was set to 100 and max stages to 1.

"Noise level"	30%	50%	70%
Base CNN	25.7	30.6	40.4
True noise[1]	24.8	29.6	36.2
Dropout[2]	24.6 (24.4)	28.6 (32.6)	35.3 (33.0)
Base CNN + ICL[3]	**25.5**	**28.1**	**33.8**
Dropout + ICL[3]	**24.5**	**26.8**	**30.6**

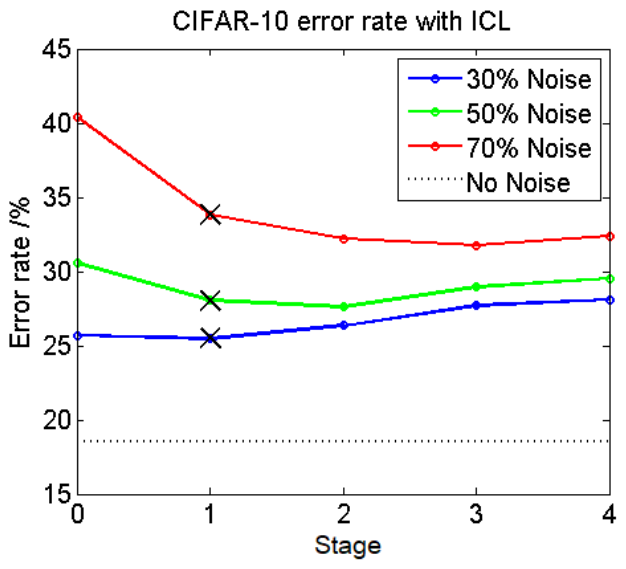
Table 7.4: *Error rates % for CIFAR-10 without clean data; "noise level" is the percent of labels randomly reassigned.*

7.5 Discussion

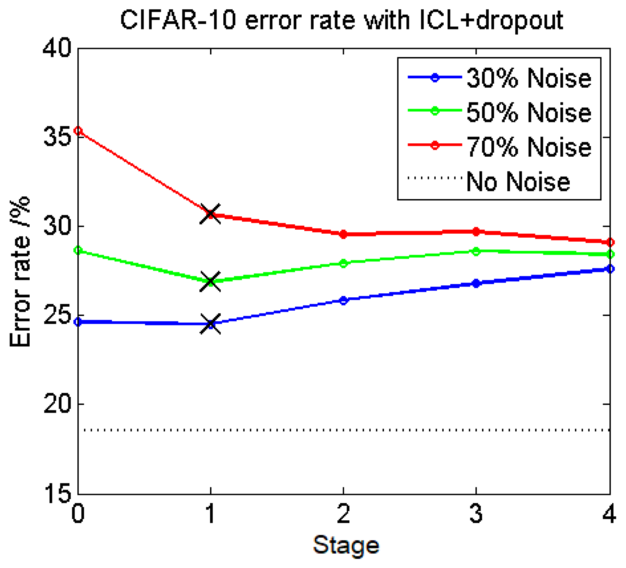
Training on the uncorrupted MNIST training set, the base model can achieve a 0.8% error rate. With 30% noise, ICL reduces the dropout regularization noise model error from 1.25% to 1.07%, substantially reducing the distance from the ideal performance achievable when trained without noise (0.8%). To compare ICL's performance with different noise levels, the absolute improvement is not as intuitive a measure as the improvement relative to the ideal of training without noise. Therefore we also measure the relative performance improvement. Define I_a, the absolute performance improvement, and I_a^*, the ideal absolute performance improvement, as

$$I_a = e - e_{ICL},\ I_a^* = e - e^*$$

where e is the model's error rate trained under noisy data, e_{ICL} is the model's error rate with using ICL, and e^* is the model's error rate trained under the original uncorrupted data.



(a) ICL + base CNN



(b) ICL + dropout

Figure 7.7: *ICL on CIFAR-10 without clean data for monitoring ("X" marks the stage returned by the algorithm).*

Then we define I_r, the relative performance improvement, as

$$I_r = I_a / I_a^* \ .$$

With this notation, the ideal performance is $e_{ICL} = e^*$, such that $I_r = 100\%$.

The relative improvement numbers (see Table 7.5) illustrate how ICL is particularly advantageous under higher noise levels. Furthermore, looking at ICL's relative improvement

"Noise level"	30%	50%	70%
MNIST with clean data, ICL improvement to:			
Base CNN	37.2%	45.0%	64.8%
Dropout	40.0%	48.0%	55.7%
MNIST without clean data, ICL improvement to:			
Base CNN	19.5%	41.2%	52.1%
Dropout	24.5%	47.0%	51.3%
CIFAR-10 with clean data, ICL improvement to:			
Base CNN	3.0%	20.2%	27.4%
Dropout	3.6%	18.5%	27.7%
CIFAR-10 without clean data, ICL improvement to:			
Base CNN	2.8%	20.1%	30.1%
Dropout	1.6%	17.8%	28.0%

Table 7.5: *ICL relative improvement I_r*

under the same noise level and the same dataset, the relative improvements are very similar from combining ICL with these two different models, base CNN and dropout regularization. This also suggests that no matter which method it is combined with, ICL will hopefully improve performance.

The underlying model still plays an important role. The final performance with ICL highly depends on the underlying model's performance. The better performance the underlying model can achieve, the better final result ICL can achieve.

7.6 Limitations

ICL has a limitation that to be effective, the underlying model, trained on uncorrupted data without ICL, must have a classification error rate lower than the noise level. If its error rate is higher than the noise level, the data will almost certainly be noisier after ICL's random flipping of labels for inputs on which the separate networks disagreed. For example, applying ICL to uncorrupted data, the error rate increases, as shown in Figure 7.8 and 7.9. ICL does not however keep going worse after each new stage (Figure 7.8 and 7.9), but seems to finally converge to a certain performance level. Note that even without a monitoring set, ICL would actually stop after stage 1 on these examples, and with a monitoring set, it should stop after stage 0, with no increase in error rate over the underlying model.

Another limitation of ICL is that it will take longer for training than training just the underlying networks it is combined with, since it needs to repeat the training for at least one stage. However, when we modify the networks' structure or tune the training hyper-parameters, we don't need to use ICL. We will only use ICL once the structure and hyper-

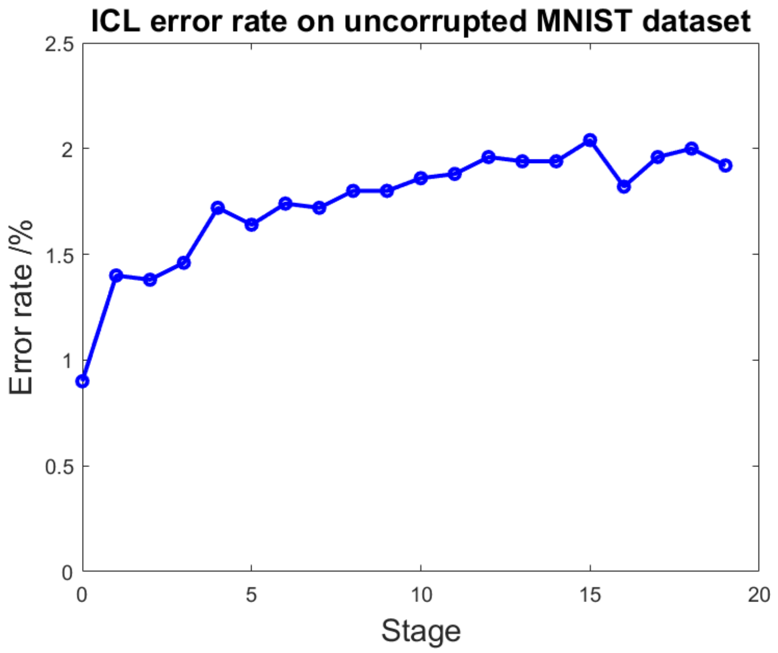


Figure 7.8: *ICL on uncorrupted MNIST data. (Stage 0's error rate is that of the underlying model.)*

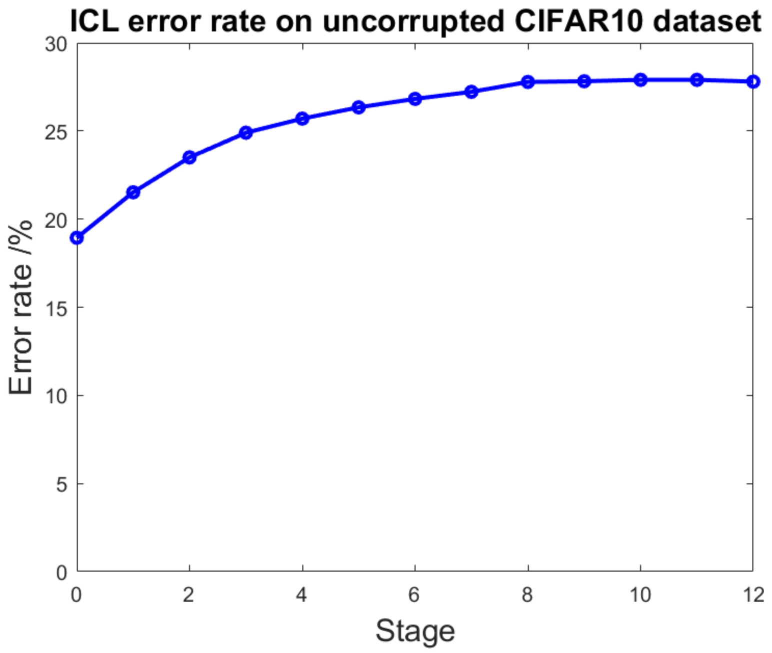


Figure 7.9: *ICL on uncorrupted CIFAR10 data. (Stage 0's error rate is that of the underlying model.)*

parameters have already been chosen and tuned using standard training. So even if ICL takes multiple stages to train, it is only at the last step of training.

7.7 Conclusions

The experimental results show that our training strategy is able to greatly improve training performance when data labels are noisy. Moreover, a great advantage of ICL is that it has virtually no parameters to choose, nor does it need to re-tune the hyper-parameters of the underlying model it is combined with. As long as the underlying model works well, ICL can be used to further improve its performance under noisy data. Furthermore, ICL could be combined with other existing or yet-to-be-invented models that address noisy labels, improving those models' performance without any cost except the training time. ICL is simple and easy to implement.

Numerous variations of the basic ICL method are envisioned. ICL doesn't rely on the structure of neural networks, so it can be used to train differently structured networks simultaneously, on different training subsets, rather than the same network structures. This could lead to different networks learning distinct features of the data, which might benefit the cross labeling process. In this chapter, we performed the experiments with convolutional neural networks as the underlying model because CNNs' performance is usually better than other approaches on image classification tasks. However, ICL isn't restricted to CNNs; it could also be combined with other types of neural networks and even other machine learning algorithms. Moreover, our training strategy isn't limited to partitioning datasets into just two separate piece with two classifiers, though fewer classifiers means that each classifier can access more training data and the label updating rules can be simpler. Different numbers of partitions and different partitioning methods are another rich area of exploration for the ICL approach.

A new algorithm inspired by ICL, Consensus Refinement Learning (CRL), will be introduced in the next Chapter.

Chapter 8

Consensus Refinement Learning from Noisy Labels

In the previous chapter, our proposed ICL algorithm demonstrates the ability to improve neural networks' performance when training on noisy datasets. However, there are still two major limitations of the ICL algorithm. One is that ICL can require lots of stages for training. The other one is that the underlying model of ICL must have a classification error rate lower than the training data's noise level.

In this chapter, to overcome ICL's limitations, we propose another easy-to-implement framework, Consensus Refinement Learning (CRL), which can also significantly improve an underlying neural network's classification performance when training on datasets with label noise and requiring only one extra hyper-parameter. The key idea of our hard-label Consensus Refinement Learning (CRL) algorithm is to retrain only on a refined, (re)labeled subset of the original data on whose classifications the diverse networks agree. Digging deeper into the problem, we realize that it is not always better to have more training data, especially when data is noisy. CRL doesn't require changing the underlying network's architecture or loss function but can significantly improve its performance, both for datasets with uniform symmetric noise as well as asymmetric noise. CRL doesn't rely on knowing the noise distribution in the training data, but can be combined with an underlying model that does so if these statistics are available. We also present a new diversity measurement to analyze the Consensus Refinement Learning algorithm and theoretically explain its effectiveness. VGGNet [122] and ResNet [48] are two of the most widely used neural network architectures. In our work, to make fair comparison with previous work, we also re-implement these two architectures in our experiments. In addition to MNIST [76] and CIFAR-10 [66], we also report experimental results on the image classification datasets CIFAR-100 [66] and Clothing1M [135] with different noise levels and different model architectures, demonstrating improvements to underlying prediction performance.

8.1 Related Work

As we stated in the previous chapter, even with significant noise in a dataset's labels, directly training a standard neural network on a dataset can still yield useful knowledge [110, 131]. Exploiting this fact, our approach independently trains multiple neural networks with identical architecture directly on the noisy labeled dataset and distills their learned consensus knowledge into a refined dataset. Our intuition is that the consensus labels are more likely to be correct. Thus, using that knowledge to train the neural network might achieve better performance than the one trained on the original noisy dataset. Unlike Hinton et al.'s [52] distillation method, which distills a cumbersome model's knowledge to a small model, in contrast we refine the original noisy labeled dataset to a smaller consensus dataset and use that dataset to retrain a single model with the same architecture as the original one. With our diversity analysis, we show that Consensus Refinement generates a dataset that tends to be significantly cleaner than the original, which is helpful for training the final model.

One reason our approach has not been tried before may be that typically in machine learning there is an assumption that more training data is better. While this is true for a clean dataset, for a noisy dataset we show that the trade-off between reducing dataset size and improving dataset label quality is handled remarkably well using consensus as the dataset refinement rule. Even though CRL will likely remove a few hard examples that would have helped to improve performance, its relabeling of a large number of mislabeled inputs plus the removal of other mislabeled inputs can have a much larger beneficial impact on performance. Furthermore, the larger the dataset, the more expensive it would be to clean it manually, but the smaller the performance hit from the reduced size of the consensus dataset.

Ensembles and Diversity

Ensemble methods [98, 109] are machine learning algorithms that combine multiple classifiers together to obtain better prediction performance than a single classifier. Compared to the ensemble method, our approach has two obvious advantages. First, our trained final model shows better performance than the ensemble method. Second, since we distill multiple models' knowledge to a single model, the application of our final model is less computationally expensive.

Classifier diversity [117, 71, 19, 29] is a concept used in ensemble algorithms. Diversity can work as a metric to evaluate the correlations between ensemble classifiers. We developed Consensus Failure (CF) to evaluate the Consensus Refinement Learning's effectiveness. Similar to the Double-Fault measurement [40] and Coincident Failure Diversity [100], we also study the data points coincidently misclassified by all the classifiers, but we go deeper and focus on those points that have been misclassified into the same class.

8.2 Noisy Labels

Our goal is to address the problem of label noise and improve neural networks' prediction performance when trained with noisy labels. For a classification dataset $(\mathcal{X}, \mathcal{Y})$, where $\mathcal{X}$ is the data and $\mathcal{Y}$ is the set of corresponding mutually exclusive labels, we denote each data point $\boldsymbol{x}$, and its noisy label $\boldsymbol{y} = \{y_1, y_2, \ldots, y_C\} \in \mathcal{Y}_C$, where C is the total number of classes and $y_i \in \{0, 1\}$. If $y_i = 1$, it indicates that data point $\boldsymbol{x}$ is labeled with the i^{th} label, and otherwise $y_j = 0$ where $j \neq i$. Each data point $\boldsymbol{x}$ also has a ground-truth label $\boldsymbol{y}^{\star} = \{y_1, y_2, \ldots, y_C\} \in \mathcal{Y}_C$, which is unobservable for the (noisy) training data.

For each dataset $(\mathcal{X}, \mathcal{Y})$, we use a confusion matrix $Q \in \mathcal{R}^{C \times C}$ (some researchers also refer to it as the "noise transition matrix") to represent the noisy label distribution artificially added. Each element $Q_{ij} \in [0, 1]$ represents the percentage of members of class i being mistakenly labeled as class j.

Label noise will of course decrease neural networks' performance. However, neural networks have some robustness to label noise and can still learn useful information from noisy labels [110, 131]. An intuitive explanation is that even with label noise, the correct label still usually dominates compared to labels of any other class. Based on this insight, we train multiple neural networks directly on the noisy data and apply our proposed "Consensus Refinement" method.

8.3 Consensus Refinement

Consensus Refinement is the training algorithm we propose to address label noise. First, we train multiple instances of an underlying neural networks on the noisy data. Since all these networks are trained independently with random initialization, random order of input data, and random data augmentation, they will have diverse prediction results. Our intuition is that the correct predictions are more robust than the incorrect predictions; thus those nonrobust incorrect predictions will be removed by Consensus Refinement. Our diversity measurement, discussed in the next section, will give more insight into why our method works well.

The Consensus Refinement method requires independently training L neural networks $D_1, D_2, \ldots, D_L$ (these L neural networks have identical architectures) on the original training data $(\mathcal{X}, \mathcal{Y})$. Then we use the L classifiers' consensus to select the data on which to train the final model $\tilde{D}$. The "consensus dataset" is the subset of the original dataset, with data points $(\boldsymbol{x}_i,\ \boldsymbol{y}_i)$ such that $\forall\ j_1, j_2 \in \{1, 2, \ldots, L\}$ satisfy $D_{j_1}(\boldsymbol{x}_i) = D_{j_2}(\boldsymbol{x}_i)$, where $D_{j_1}(\boldsymbol{x}_i)$ denotes D_{j_1}'s prediction of $\boldsymbol{x}_i$. We take all the consensus data points and relabel them $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ (ignoring the original label) as a new dataset, where $\tilde{\mathcal{Y}} = D_1(\tilde{\mathcal{X}})$, and remove other points. Finally we train the final model $\tilde{D}$ on the consensus dataset $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ and output $\tilde{D}$. Algorithm 1 summarizes the Consensus Refinement algorithm.

Algorithm 4 Consensus Refinement Learning

```
1: $(\mathcal{X}, \mathcal{Y}) \leftarrow$ training set with noisy labels
2: randomly initialize weights of $L$ classifiers $D_1$, $D_2$, ..., $D_L$
3: for $j \leftarrow 1$ to $L$ do
     // randomize order, random data augmentation
4:   $D_j \leftarrow D_j$ trained independently on $(\mathcal{X}, \mathcal{Y})$
5:   for $i \leftarrow 1$ to $|\mathcal{X}|$ do
       // predict labels for $\mathcal{X}$ with $D_j$
6:     $\hat{\boldsymbol{y}}_{j,i} \leftarrow D_j(\boldsymbol{x}_i)$
7:   end for
8: end for
9: $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) \leftarrow$ new empty training set
10: for $i \leftarrow 1$ to $|\mathcal{X}|$ do
      // check if all predictions match
11:   $consensus \leftarrow true$
12:   for $j \leftarrow 2$ to $L$ do
13:     if $\hat{\boldsymbol{y}}_{1,i} \neq \hat{\boldsymbol{y}}_{j,i}$ then
        // check if all predictions match the first
14:       $consensus \leftarrow false$
15:       break
16:     end if
17:   end for
18:   if $consensus$ then
19:     add $(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_{1,i})$ into $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$
20:   end if
21: end for
22: randomly initialize weights of classifier $\tilde{D}$
23: $\tilde{D} \leftarrow \tilde{D}$ trained on $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$
24: return $\tilde{D}$
```

8.4 Neural Network Diversity

Traditionally, "diversity" is a concept used to analyze ensemble methods. Diversity measurements can help analyze the entire ensemble training system. Researchers have come up with several different ways to define and use diversity measurements. The basic diversity metrics are pairwise diversity; some of them can be extended to non-pairwise measurements [71].

We propose a new diversity measurement, the Consensus Failure (CF) fraction, which represents the fraction of incorrect labels in the consensus dataset. It helps to analyze the Consensus Refinement method. CF theoretically explains the improvement by Consensus Refinement over underlying models since the consensus dataset is proved to be a better dataset on which to train the final model.

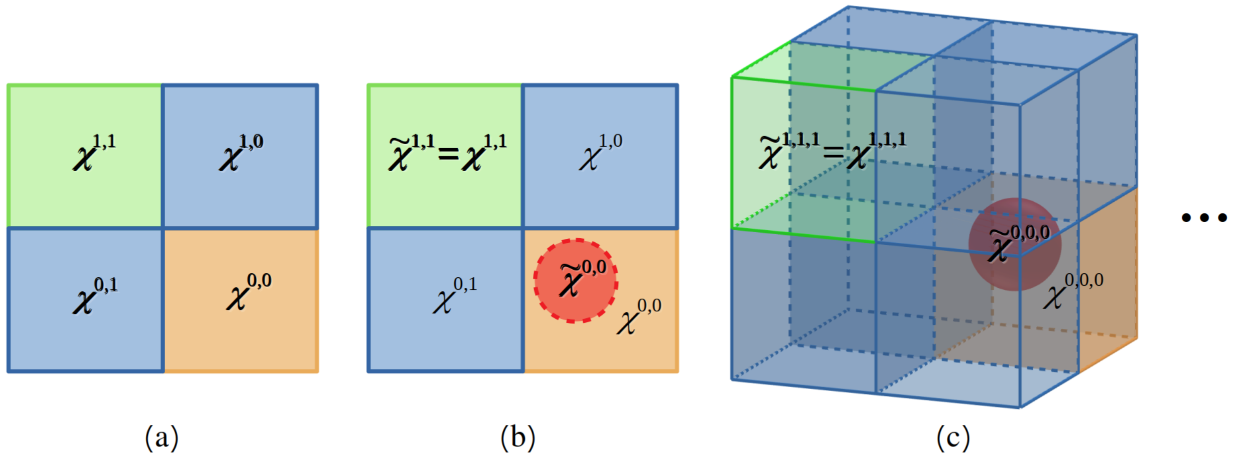


Figure 8.1: *Visualization of dataset partition for calculating* ***(a)*** *traditional pairwise diversity measurements* ***(b)*** *pairwise Consensus Failure* ***(c)*** *triplet Consensus Failure.*

8.4.1 Traditional Pairwise Diversity

Various pairwise diversity metrics have been developed to characterize the relationship between a pair of classifiers for ensemble learning. Given two classifiers, D_1 and D_2, each data point $\boldsymbol{x}_i$ will have two predicted labels, $\hat{\boldsymbol{y}}_{1,i} = D_1(\boldsymbol{x}_i)$ and $\hat{\boldsymbol{y}}_{2,i} = D_2(\boldsymbol{x}_i)$. Comparing these with the ground-truth label $\boldsymbol{y}_i^\star$, data point $\boldsymbol{x}_i$ has four possible situations, $\{d_{1,i}, d_{2,i}\}$, where $d_{j,i} = 1$ if $\hat{\boldsymbol{y}}_{j,i} = \boldsymbol{y}_i^\star$, otherwise $d_{j,i} = 0$. Accordingly, the dataset $\mathcal{X}$ can be divided into four subsets, $\mathcal{X}^{1,1}, \mathcal{X}^{1,0}, \mathcal{X}^{0,1}$ and $\mathcal{X}^{0,0}$ (Figure 8.1 a). For example, $\mathcal{X}^{1,1}$ represents the subset with data points that both D_1 and D_2 correctly recognize, $\mathcal{X}^{1,0}$ represents the subset with data points that D_1 correctly recognizes and D_2 misclassifies, and so on.

Typical diversity metrics are calculated based on the relationships between the size of these subsets, $|\mathcal{X}^{1,1}|, |\mathcal{X}^{1,0}|, |\mathcal{X}^{0,1}|$ and $|\mathcal{X}^{0,0}|$. For example, the double-fault (DF) measurement [40] is defined as the proportion of the data points that have been misclassified by both classifiers:

$$DF = \frac{|\mathcal{X}^{0,0}|}{|\mathcal{X}^{1,1}| + |\mathcal{X}^{1,0}| + |\mathcal{X}^{0,1}| + |\mathcal{X}^{0,0}|} = \frac{|\mathcal{X}^{0,0}|}{|\mathcal{X}|}.$$

Because the DF diversity measurement is designed for use with general ensemble methods, the metric takes all the data $\mathcal{X}$ into account. Furthermore it doesn't distinguish whether a double-fault data point $\boldsymbol{x}_i^{0,0} \in \mathcal{X}^{0,0}$ is misclassified into the same class by both classifiers, i.e. whether $D_1(\boldsymbol{x}_i^{0,0}) = D_2(\boldsymbol{x}_i^{0,0})$ or $D_1(\boldsymbol{x}_i^{0,0}) \neq D_2(\boldsymbol{x}_i^{0,0})$.

8.4.2 Pairwise Consensus Failure Measurement

Unlike in ensemble learning, we are only interested in the data with consensus in the predictions from different classifiers, since we will ultimately train the final model $\tilde{D}$ only on the consensus dataset $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ rather than the entire dataset $(\mathcal{X}, \mathcal{Y})$.

Thus we define a new metric, the Consensus Failure (CF) fraction. The pairwise version, denoted κ_{pair}, is defined as:

$$\kappa_{pair} = \frac{|\tilde{\mathcal{X}}^{0,0}|}{|\tilde{\mathcal{X}}^{1,1}| + |\tilde{\mathcal{X}}^{0,0}|} = \frac{|\tilde{\mathcal{X}}^{0,0}|}{|\tilde{\mathcal{X}}|}$$

where $\tilde{\mathcal{X}}^{0,0}$ is the data that both original classifiers misclassified identically: $\forall\, \tilde{\boldsymbol{x}}_i^{0,0} \in \tilde{\mathcal{X}}^{0,0}$ such that $D_1(\tilde{\boldsymbol{x}}_i^{0,0}) = D_2(\tilde{\boldsymbol{x}}_i^{0,0}) \neq \boldsymbol{y}_i^\star$; and $\tilde{\mathcal{X}}^{1,1}$ is the same as $\mathcal{X}^{1,1}$ (data points both classifiers correctly recognized), since the correct answer is always the same $D_j(\tilde{\boldsymbol{x}}_i^{1,1}) = D_j(\boldsymbol{x}_i^{1,1}) = \boldsymbol{y}_i^\star$, $j \in \{1, 2\}$ (Figure 8.1 b).

Note that, for the simplest classification problem of binary classification, $D_1(\boldsymbol{x}_{i_1}^{0,0}) \equiv D_2(\boldsymbol{x}_{i_1}^{0,0})$ and there is no case where $D_1(\boldsymbol{x}_{i_2}^{0,0}) \neq D_2(\boldsymbol{x}_{i_2}^{0,0})$. In that case, $\kappa_{pair} = DF$.

8.4.3 General Consensus Failure Measurement

CF can be extended from a pairwise to a general version. If we train three neural networks D_1, D_2, and D_3, we can similarly define the triplet Consensus Failure fraction $\kappa_{triplet}$:

$$\kappa_{triplet} = \frac{|\tilde{\mathcal{X}}^{0,0,0}|}{|\tilde{\mathcal{X}}^{1,1,1}| + |\tilde{\mathcal{X}}^{0,0,0}|} = \frac{|\tilde{\mathcal{X}}^{0,0,0}|}{|\tilde{\mathcal{X}}|}$$

where $\tilde{\mathcal{X}}^{0,0,0}$ is the consensus failure data such that $\forall\, \tilde{\boldsymbol{x}}_i^{0,0,0} \in \tilde{\mathcal{X}}^{0,0,0}$, $D_1(\tilde{\boldsymbol{x}}_i^{0,0,0}) = D_2(\tilde{\boldsymbol{x}}_i^{0,0,0}) = D_3(\tilde{\boldsymbol{x}}_i^{0,0,0}) \neq \boldsymbol{y}_i^\star$, and $\tilde{\mathcal{X}}^{1,1,1}$ is the same as $\mathcal{X}^{1,1,1}$ (Figure 8.1 c).

More generally, with L classifiers, the Consensus Failure fraction κ is:

$$\kappa = \frac{|\tilde{\mathcal{X}}^{\overbrace{0,0,\ldots,0}^{L}}|}{|\tilde{\mathcal{X}}^{\overbrace{1,1,\ldots,1}^{L}}| + |\tilde{\mathcal{X}}^{\overbrace{0,0,\ldots,0}^{L}}|} = \frac{|\tilde{\mathcal{X}}^{\overbrace{0,0,\ldots,0}^{L}}|}{|\tilde{\mathcal{X}}|}$$

Theoretically, κ can be any value in the range $[0, 1]$. However, based on experiments with corrupted versions of MNIST, CIFAR-10, and CIFAR-100, we've never observed it outside of the range $[0, 0.5]$, which means that the consensus dataset has more correct than incorrect labels. Thus, we will next analyze what happens under this assumption.

$$|\tilde{\mathcal{X}}^{\overbrace{0,0,\ldots,0}^{L}}| \leq |\tilde{\mathcal{X}}^{\overbrace{1,1,\ldots,1}^{L}}|. \tag{8.1}$$

This assumption will of course be true when the underlying model's performance is good. It also seems intuitive as long as the underlying neural network performance is not that bad. But we found that even for the CIFAR-100 dataset with high levels of label noise such that the underlying model has over 50% incorrect predictions, this assumption was never violated (as well as for all our other experiments). Under this assumption, we can perform the following analysis for the Consensus Refinement algorithm.

8.4.4 CF Analysis for Consensus Refinement

If we were to use a single underlying model's prediction as the new dataset to train a final model, the final model might overfit the new training set and therefore the best performance it could achieve would be the previous underlying model's performance. The reason Consensus Refinement works well is that the labels of the dataset $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ generated by consensus are expected to be cleaner than the underlying neural network's predicted labels would be. This makes it possible for the final neural network trained on $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ to achieve better performance, even though the consensus dataset's size will be smaller. We give the simplified proof of this for the case with a pair of classifiers; the general proof follows along similar lines.

Given two classifiers D_1 and D_2 independently trained on $(\mathcal{X}, \mathcal{Y})$, we denote their repredicted labels' error rates for $\mathcal{X}$ as e_{D_1} and e_{D_2}, respectively (the percentage of incorrect labels in the re-predicted labels), and denote the percentage of incorrect labels in the consensus dataset $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ by κ_{pair}. Since D_1 and D_2 have identical architecture, the expected error rate of the two classifiers should be the same, $E(e_{D_1}) = E(e_{D_2})$, from which it also follows that

$$E(e_{D_1}) = E(\frac{|\mathcal{X}^{0,0}| + |\mathcal{X}^{0,1}|}{|\mathcal{X}^{1,1}| + |\mathcal{X}^{1,0}| + |\mathcal{X}^{0,1}| + |\mathcal{X}^{0,0}|})$$

$$E(e_{D_2}) = E(\frac{|\mathcal{X}^{0,0}| + |\mathcal{X}^{1,0}|}{|\mathcal{X}^{1,1}| + |\mathcal{X}^{1,0}| + |\mathcal{X}^{0,1}| + |\mathcal{X}^{0,0}|}).$$

$$\iff E(|\mathcal{X}^{1,0}|) = E(|\mathcal{X}^{0,1}|).$$

$$\implies E(e_{D_1}) = \frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + 2E(|\mathcal{X}^{0,1}|) + E(|\mathcal{X}^{0,0}|)}. \tag{8.2}$$

And we also have

$$E(\kappa_{pair}) = E(\frac{|\tilde{\mathcal{X}}^{0,0}|}{|\tilde{\mathcal{X}}^{1,1}| + |\tilde{\mathcal{X}}^{0,0}|}) = \frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|) + E(|\tilde{\mathcal{X}}^{0,0}|)}. \tag{8.3}$$

With equations (8.3) and (8.2), then we will prove

$$E(\kappa_{pair}) \le E(e_{D_1}).$$

Proof:

$$E(\kappa_{pair}) \le E(e_{D_1})$$

$$\iff \frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|) + E(|\tilde{\mathcal{X}}^{0,0}|)} \le \frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + 2E(|\mathcal{X}^{0,1}|) + E(|\mathcal{X}^{0,0}|)}$$

Rearranging terms, we get

$$\frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)} \leq \frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + E(|\mathcal{X}^{0,1}|)}.$$

We consider two cases.

a) If $E(|\mathcal{X}^{0,0}|) \geq E(|\mathcal{X}^{1,1}|)$, then we have

$$\frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + E(|\mathcal{X}^{0,1}|)} \geq 1,$$

and with our assumption (8.1) that $|\tilde{\mathcal{X}}^{0,0}| < |\tilde{\mathcal{X}}^{1,1}|$, and since $|\mathcal{X}^{1,1}| = |\tilde{\mathcal{X}}^{1,1}|$:

$$\frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)} = \frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\tilde{\mathcal{X}}^{1,1}|)} \leq 1;$$

thus, we have

$$\frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)} \leq \frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + E(|\mathcal{X}^{0,1}|)}.$$

b) Otherwise, if $E(|\mathcal{X}^{0,0}|) < E(|\mathcal{X}^{1,1}|)$, then

$$\frac{E(|\mathcal{X}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)} < \frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + E(|\mathcal{X}^{0,1}|)},$$

and since $\tilde{\mathcal{X}}^{0,0} \subseteq \mathcal{X}^{0,0}$,

$$\frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)} \leq \frac{E(|\mathcal{X}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)};$$

thus, we also have

$$\frac{E(|\tilde{\mathcal{X}}^{0,0}|)}{E(|\mathcal{X}^{1,1}|)} < \frac{E(|\mathcal{X}^{0,0}|) + E(|\mathcal{X}^{0,1}|)}{E(|\mathcal{X}^{1,1}|) + E(|\mathcal{X}^{0,1}|)}.$$

Therefore, we prove that the consensus dataset $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$ should be cleaner than $(\mathcal{X}, D_1(\mathcal{X}))$. This explains why the Consensus Refinement method performs well.

8.5 Experiments

Our algorithm performs well on a variety of image classification datasets: MNIST [76], CIFAR-10 [66], CIFAR-100 [66], and Clothing1M [135]. We also tested our algorithm on a variety of different model architectures. Note that all the models in our experiments are trained from scratch. For the MNIST and both CIFAR training sets, we test on both symmetric noise and asymmetric noise by artificially adding label noise. Given a desired symmetric noise level p_{sy}, we randomly flip p_{sy} portion of the training labels to random labels

Table 8.1: *MNIST LeNet experimental results (%)* [1]

MNIST (LeNet; clean data performance: 99.3)						
Noise level	$p_{sy} = 0.2$	$p_{sy} = 0.3$	$p_{sy} = 0.6$	$p_{sy} = 0.7$	$p_{asy} = 0.2$	$p_{asy} = 0.4$
Underlying	98.3	98.0	96.9	95.6	98.3	93.4
true noise [126]		98.7		96.7		
dropout [54]		98.8		96.9		
ICL [141]		98.3		97.5		
Ensemble-3	98.7	98.5	97.6	96.8	99.1	95.7
Ensemble-4	98.7	98.5	97.6	97.1	**99.2**	96.4
CRL-2	**99.1** (0.6)	**98.9** (0.7)	**98.4** (0.8)	97.7 (1.1)	**99.2** (0.7)	97.6 (2.4)
CRL-3	99.0 (0.4)	98.7 (0.5)	98.3 (0.5)	**97.8** (0.7)	99.0 (0.6)	**97.8** (1.5)
CRL-4	99.0 (0.3)	98.7 (0.3)	98.1 (0.4)	97.7 (0.4)	99.1 (0.5)	**97.8** (1.3)

for all three datasets. We follow previous research to add asymmetric noise on MNIST and CIFAR-10. On MNIST, we randomly pick p_{asy} portion of the labels and flip them following the mapping: $2 \rightarrow 7$, $3 \rightarrow 8$, $5 \rightarrow 6$, and $7 \rightarrow 1$. On CIFAR-10, we also randomly pick p_{asy} portion of the labels and flip them following the mapping: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, and CAT $\rightarrow$ DOG. On CIFAR-100, we add asymmetric noise by randomly picking p_{asy} portion of the labels and flip them to a label randomly chosen from one of the first ten classes.

Moreover, we also test on a dataset with "natural" noise, Clothing1M [135], which consists of one million training images labeled by an inaccurate automated process, and also a small clean training set of 50K images, and a separate clean test set of 10K images. All these input images are resized into 224×224 in our experiments. This dataset has 14 categories, such as coat, sweater, shirt, and so on. The noise level in the training set is around 39% and the noise is highly asymmetric [135].

Besides comparison with previous work (omitting comparison with methods that require extra information, such as the distribution of the noise), we also compare to the performance of the underlying model, since the ability to improve on the performance of the underlying model is the key contribution of the algorithm. We also compare to the results of the majority vote ensemble algorithm using the same number of underlying models. In our experiments, all the models trained on the consensus dataset use the same settings as the underlying models.

8.5.1 Experiments on MNIST

[1]Results are reported as percentages. The first row gives the underlying model's architecture and performance trained on the original clean dataset. The numbers following the word "Ensemble" or "CRL" indicate the number of underlying models. CF fraction κ values, expressed as percentages, are in parentheses after

For the MNIST dataset, we use LeNet [76] with kernel size 5 as the underlying model. A dropout layer with dropout rate of 0.5 is added to the fully connected layer. We compare with Sukhbaatar et al. [126], Jindal et al. [54], and Yuan et al. [141], since they also used similar convolutional neural networks as their underlying models. We don't list results of some other work using fully connected neural networks in Table 8.1, since their performance is surpassed by these CNN-based methods.

Because MNIST is not a very challenging dataset, all the algorithms perform well and the differences are small. Nevertheless, we can still see that CRL almost always slightly outperforms all the other algorithms.

8.5.2 Experiments on CIFAR

For both CIFAR-10 and CIFAR-100, we perform data augmentation by horizontal random flips and 32×32 random crops with 4 pixels padding. Both training batch sizes are set to 200.

For CIFAR-10, we test 14-layer ResNet [48] as our underlying model. We train the model with learning rate 1.0×10^{-1} for 100 epochs and 1.0×10^{-2} for another 50 epochs, using the SGD optimizer with 0.9 momentum. In these experiments, we see (Table 8.2) that CRL-3 and CRL-4 show better performance than CRL-2. Comparing the amount of training data in our consensus dataset, for example, when $p_{sy} = 0.2$ there are $88\%, 82\%$, and 79% training images retained for CRL-2, CRL-3, and CRL-4 respectively; when $p_{sy} = 0.6$ there are $72\%, 62\%$, and 57% training images retained for CRL-2, CRL-3, and CRL-4 respectively. Even with significantly less data, the improved quality of the consensus labels provide higher performance.

To experiment with the effect of different model architectures with the same dataset, we also test a small model architecture that is a simplified VGG network [122], removing the last two convolutional layers and the first two fully connected layers of VGG-11. We train the model for 150 epochs, using the Adam optimizer [64] and a learning rate of 1.0×10^{-4}. When trained on the clean CIFAR-10 dataset, ResNet-14 performs only slightly better, whereas on the noisy dataset, the advantages of ResNet-14 are more apparent. This suggests that ResNet is more robust to the noise.

For CIFAR-100, we use 44-layer ResNet, trained with the same learning rate and optimizer as our settings for CIFAR-10 with ResNet-14. CRL-3 instead of CRL-4 shows the best performance in all three noise levels on the CIFAR-100 dataset, especially when $p_{sy} = 0.6$. This is mainly due to the size of the training set. For example, when $p_{sy} = 0.6$ there are only $46\%, 35\%$, and 29% training images remaining for CRL-2, CRL-3, and CRL-4, respectively. There are only 50,000 images spread over 100 categories in the original training dataset. In this case, the dramatically decreased size of the dataset is still only causing a small degradation in performance, but could become a potential issue if we were to add more initial

performance numbers for our method. Bold indicates the best performance.

Table 8.2: *CIFAR-10 ResNet experimental results (%)* [1]

CIFAR-10 (ResNet-14; clean data performance: 89.2)			
Noise level	$p_{sy} = 0.2$	$p_{sy} = 0.6$	$p_{asy} = 0.2$
unhinged [130]	84.1		83.8
sigmoid [39]	66.6		71.8
bootstrap soft [107]	84.3		84.6
bootstrap hard [107]	83.6		84.7
Underlying	83.9	75.4	85.3
backward [101]	80.4		83.8
forward [101]	83.4		87.0
two-stage [30]	84.5		85.6
Ensemble-3	86.5	77.8	87.7
Ensemble-4	87.2	78.7	87.9
CRL-2	87.4 (3.2)	78.6 (11.3)	88.4 (2.4)
CRL-3	87.5 (1.7)	79.1 (6.9)	**88.6** (1.4)
CRL-4	**87.7** (1.1)	**79.9** (4.5)	88.5 (1.1)

Table 8.3: *CIFAR-10 VGG experimental results (%)* [1]

CIFAR-10 (VGG-11; clean data performance: 88.8)						
Noise level	$p_{sy} = 0.2$	$p_{sy} = 0.3$	$p_{sy} = 0.6$	$p_{sy} = 0.7$	$p_{asy} = 0.2$	$p_{asy} = 0.4$
Underlying	82.2	81.5	73.4	69.6	84.3	76.9
Ensemble-3	85.1	83.3	76.0	72.4	86.7	79.8
Ensemble-4	85.8	83.9	76.4	72.6	86.9	80.2
CRL-2	86.0 (8.2)	85.2 (9.7)	78.4 (15.0)	76.3 (18.0)	86.5 (8.1)	80.4 (14.2)
CRL-3	**86.2** (6.0)	**85.7** (6.8)	79.1 (11.9)	76.5 (13.7)	87.1 (5.9)	**81.6** (11.3)
CRL-4	**86.2** (4.7)	84.8 (5.9)	**79.5** (9.9)	**76.6** (11.7)	**87.5** (4.9)	81.2 (9.4)

underlying models. However, in practice, label noise problems are more of an issue with very large datasets. If we do have a noisy dataset with only a few data points, then it is likely better to use a smaller number of classifiers for CRL. In all cases, CRL showed a great improvement to the underlying model, as well as outperforming traditional ensemble methods.

8.5.3 Experiments on Clothing1M

We use 50-layer ResNet as the underlying model for the Clothing1M dataset. We trained the model with the SGD optimizer with 0.9 momentum, and batch size 100 using the 1M images with noisy labels. The learning rate is set to 1.0×10^{-1}, 1.0×10^{-2}, and 1.0×10^{-3}

Table 8.4: *CIFAR-100 ResNet experimental results (%)* [1]

CIFAR-100 (ResNet-44; clean data performance: 68.5)			
Noise level	$p_{sy} = 0.2$	$p_{sy} = 0.6$	$p_{asy} = 0.2$
unhinged [130]	47.5		48.0
sigmoid [39]	47.6		55.6
bootstrap soft [107]	57.8		63.8
bootstrap hard [107]	57.3		63.9
Underlying	57.9	44.0	63.5
backward [101]	55.7		63.8
forward [101]	58.6		64.2
Ensemble-3	66.9	48.7	65.0
Ensemble-4	67.1	51.3	66.3
CRL-2	67.2 (5.5)	52.0 (18.1)	66.2 (6.3)
CRL-3	**67.9** (2.9)	**52.9** (9.2)	**67.0** (3.2)
CRL-4	67.8 (1.9)	52.4 (5.8)	66.8 (2.0)

for 4, 4, and 2 epochs, respectively. As with previous work [101], we then fine-tune the model with the 50K clean training data. The learning rate for fine-tuning is set to 1.0×10^{-3} and 1.0×10^{-4} for another 10 epochs each. We perform data augmentation similar to for the CIFAR dataset, by horizontal random flips and 224×224 random crops with 10 pixels padding.

Results for the Clothing1M dataset are given in Table 8.5. (Since we don't have ground-truth labels for the training data, we are not able calculate κ values.) We can see that CRL-4 performs better than CRL-3, and CRL-3 performs better than CRL-2. (There are $81\%, 73\%$, and 69% training images remaining for CRL-2, CRL-3, and CRL-4, respectively.) Since there is enough original training data and only 14 categories, it might not make much difference to use 1M or 500K training images. The reduction of the dataset's size is not the major factor in the model's performance. Thus, CRL with a higher number of classifiers builds a better consensus dataset and achieves better performance.

Some other approaches (backward [101], EM [135]) show even better performance than CRL using ResNet-50. In fact, those approaches use the noise distribution on the Clothing1M training set (the confusion matrix was provided in [135])). This extra information will of course improve the model's performance. However, the correct noise distribution of a dataset might be difficult or impossible to get in practice on the one hand. On the other hand, if we do know the actual noise distribution, CRL can also use these models as underlying models to achieve even better performance. (Unfortunately, code for these models is not publicly available, so we were unable to show the combined performance.)

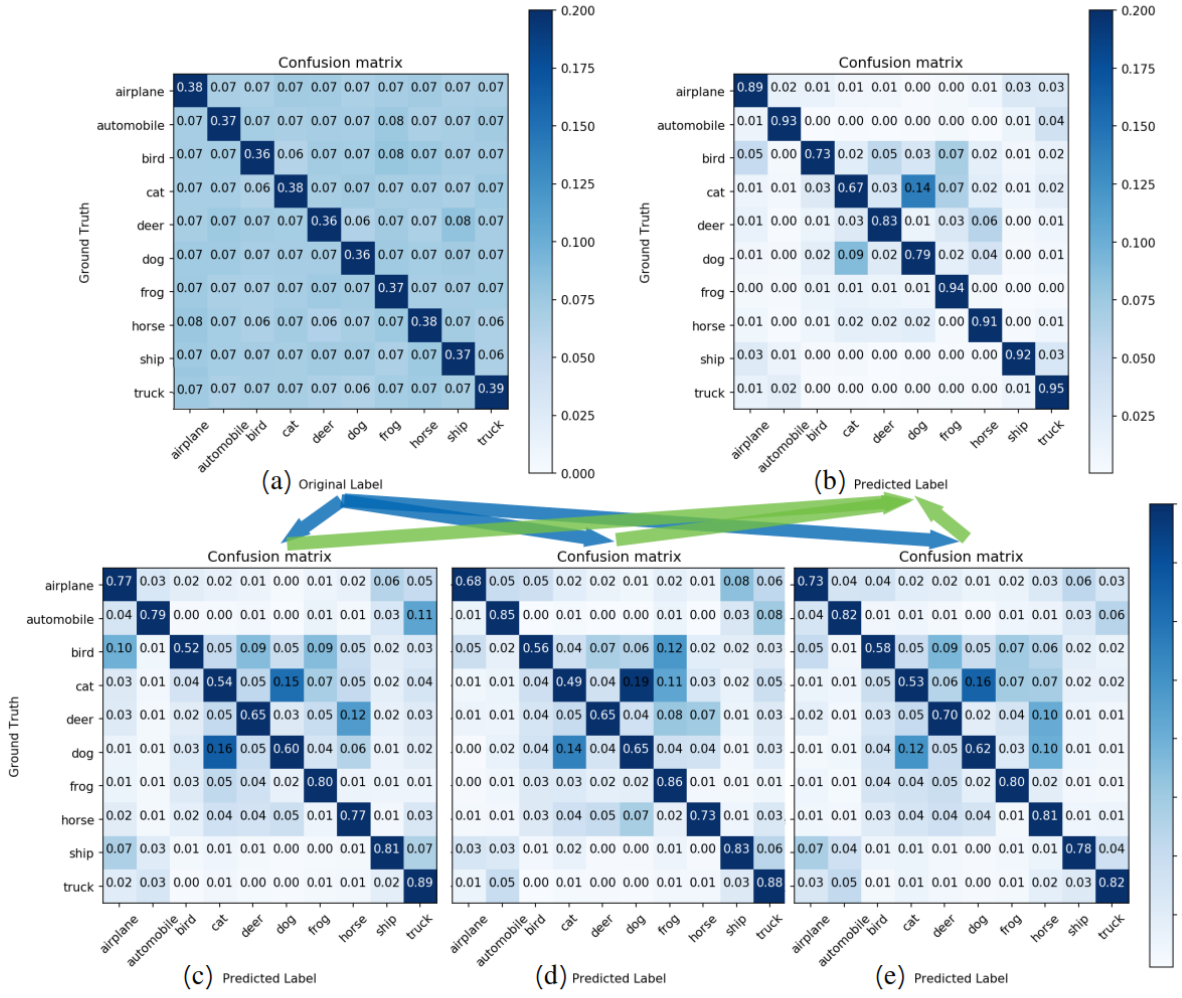


Figure 8.2: *Visualization of confusion matrices for experiments on CIFAR-10 dataset with* $p_{sy} = 0.7$ *noise:* ***(a)*** *original noisy dataset* $(\mathcal{X}, \mathcal{Y})$*, independently* ***(b)(c)(d)*** *trained models' predictions* $(\mathcal{X}, D_1(\mathcal{X}))$*,* $(\mathcal{X}, D_2(\mathcal{X}))$*,* $(\mathcal{X}, D_3(\mathcal{X}))$*, and* ***(e)*** *consensus dataset* $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$*. (Note that because the noise is randomly assigned, there is slight asymmetry even in the original confusion matrix.)*

Table 8.5: *Clothing1M ResNet experimental results (%)*

Clothing1M (ResNet-50)	
Pseudo-Label [80]	73.0
true noise [126]	76.2
EM [135]	78.2
Underlying	74.5
backward [101]	**80.4**
Ensemble-3	76.7
Ensemble-4	77.5
CRL-2	76.9
CRL-3	77.7
CRL-4	78.2

8.6 Discussion

The experimental results suggest that our proposed Consensus Refinement Learning consistently improves the underlying model's performance. Figure 8.2 gives visualization examples of confusion matrices for experiments on the CIFAR-10 dataset. It corresponds to the results when $p_{sy} = 0.7$ trained with VGG-11 in Table 8.3 CRL-3. Figure 8.2 (a) is the confusion matrix of the original training data; (b), (c) and (d) are confusion matrices of each independently trained model's prediction; and (e) shows the consensus dataset's confusion matrix, which is obviously cleaner than (b), (c) or (d). Even though the size of the consensus dataset is only 60% of the original training dataset's size, the final model's prediction accuracy is improved from the underlying model's 69.6% accuracy to 76.5%.

In fact, since many of the original data points are not in the consensus dataset, our CRL results can be further improved with semi-supervised learning treating these data points as unlabeled data. We tested this idea on the CIFAR-100 dataset with 60% symmetric noise. The consensus dataset was obtained as before with CRL-2, and then we trained our model in a semi-supervised learning fashion using the consensus dataset as the label data and the remainder of the original CIFAR-100 dataset as unlabeled data. The size of the unlabeled dataset was only around twice the size of the labeled dataset, even for this case with a relatively small consensus data set, so the improvement from semi-supervised learning would not be expected to be large. We obtain an extra 1% prediction accuracy over just CRL-2 from 52% to 53%. Even though the improvement from semi-supervised learning was not very large, semi-supervised learning does improve performance, and when the relative size of the consensus dataset is even smaller, the combination could be even more advantageous.

Comparing to other approaches, our method matches or exceeds the best performance in all cases tested except for the Clothing1M dataset. Moreover, the CF κ value is always lower than 0.5 in all cases we tested, supporting our assumption. Comparing to the ensemble approach, CRL shows better performance with training the same number of underlying models; when the noise level is high, the difference between performance of CRL and ensembles is even larger. Moreover, even taking similar training effort, CRL only uses a single model during inference, for a much lower prediction cost than for ensembles.

Due to the relatively long training time of both, we haven't combined CRL with other existing methods that use noise distribution as input. In practice, even though we can't have perfect information about the distribution of noise, we can sample some training data to estimate the approximate noise distribution. If we also use the noise distribution information with our approach, we believe we can achieve even better performance.

We have proposed a training algorithm to improve the performance of neural networks trained on datasets with noisy labels. Our CRL has no restriction on the underlying neural network architecture, and can significantly improve its performance. Analysis based on the proposed Consensus Failure diversity measure provides a theoretical explanation for the effectiveness of CRL. The proposed algorithm is a general method that can be applied to tasks beyond image classification, in any machine learning application with label noise.

Chapter 9

Summary and Future Work

In summary, we introduced our entire SLM in-situ monitoring framework and applied it to over two thousand in-situ videos. We use ex-situ measurement to get height maps for the laser scanned tracks. With our image processing algorithm, we can get accurate pixel-level classification results of height maps, and then generate labels for each corresponding in-situ video from those height maps. Then we train CNNs on these labeled videos to recognize desired metrics, such as width and continuity of tracks. Our trained models achieve excellent performance on prediction via in-situ videos with correlation coefficient of $R^2 = 0.93$ for track width, $R^2 = 0.70$ for standard deviation of track width, and prediction accuracy of 93.1% for track continuity. Our experimental results demonstrate deep learning's powerful ability and also flexibility solving AM problems. Since our proposed framework is not limited to SLM monitoring, it can also be applied to other AM monitoring systems.

In practice, there are two major issues with deep learning, insufficient labeled data and label noise. We use semi-supervised machine learning to address the problem of insufficient labeled data. This approach improves the training label efficiency and alleviates the problem of requiring a huge amount of labeled data. Our experimental results also demonstrate the semi-supervised approach's excellent performance on SLM in-situ monitoring with limited training labels.

As for the label noise problem, we proposed two original training approaches, ICL and CRL, to improve neural networks' performance on noisy data. Our proposed approaches have only a few extra hyper-parameters and no restriction on the underlying network architecture, making them easy to combine with other existing models. Since it is difficult to verify our approaches on the SLM video data because of the ambiguity of the "ground truth," we test our approaches on public datasets. Both ICL and CRL show excellent performance and improve the underlying models' performance when training on noisy datasets. Since lots of other manufacturing areas face the same problems of insufficient labeled data and label noise, we believe semi-supervised learning and also our proposed ICL and CRL algorithms are promising when addressing such issues in machine learning for monitoring in other areas of manufacturing as well.

The following topics are considered as future research:

- On the one hand, all the current experiments were done using instruments at LLNL. Our proposed in-situ monitoring approach can be tested on data collected by other institutions or researchers in the future, in order to further prove that our approach can be applied broadly; on the other hand, we labeled and organized a great amount of SLM data. The existing dataset that provides other researchers great opportunities for future research.

- Width regression and continuity classification are just the first step toward in-situ error detection. Besides these metrics, our ex-situ algorithm can also provide other track quality metrics, such as the tracks' smoothness, average height of the track, and so on. Proceeding these other qualities can be addressed using similar or even the same CNN architecture, making our in-situ monitoring method more informative and useful in practice.

- Currently, we only predict visible SLM track metrics, using only in-situ videos. However, some other sensor data, such as in-situ pyrometer data or acoustic data, or ex-situ measurements such as X-ray, could be also taken into account. With sensor fusion, we might get more information and more accurate predictions. With other data, other metrics could be predicted; for example, with X-ray measurements, we can determine whether pores (voids) exist in the final product.

- Using other sensor data for labeling, we might hope to train deep learning models for in-situ monitoring of the scanning process, ultimately using them to provide feedback to adjust the SLM system parameters in real time, and prevent defects such as pore and break generation in the final product.

In fact, both semi-supervised learning and training on noisy labels are trying to regularize the neural networks to not overfit the training data. Semi-supervised learning reduces overfitting to limited training labels; training on noisy labels is trying to reduce overfitting to inaccurate labels. We do see some previous work using semi-supervised learning methods to address label noise problems, as described in Chapter 2. However, to our knowledge, there is no existing approach that trains semi-supervised learning models first and then treats models' predictions as noisy labels for further training. In addition, it is even more difficult to simultaneously overcome the issues of limited and inaccurately labeled training data. Determining how both the label noise problem and insufficient labeled data problem can be addressed well at the same time is a rich area for future research.

Bibliography

[1] Mostafa Abdelrahman et al. “Flaw detection in powder bed fusion using optical imaging”. In: *Additive Manufacturing* 15 (2017), pp. 1–11.

[2] Nesma T Aboulkhair et al. “Reducing porosity in AlSi10Mg parts processed by selective laser melting”. In: *Additive Manufacturing* 1 (2014), pp. 77–86.

[3] Masoumeh Aminzadeh. “A machine vision system for in-situ quality inspection in metal powder-bed additive manufacturing”. PhD thesis. Georgia Institute of Technology, 2016.

[4] Masoumeh Aminzadeh and Thomas Kurfess. “Vision-based inspection system for dimensional accuracy in powder-bed additive manufacturing”. In: *ASME 2016 11th International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers. 2016, V002T04A042–V002T04A042.

[5] Masoumeh Aminzadeh and Thomas R Kurfess. “Online quality inspection using Bayesian classification in powder-bed additive manufacturing from high-resolution visual camera images”. In: *Journal of Intelligent Manufacturing* (2018), pp. 1–19.

[6] Samaneh Azadi et al. “Auxiliary image regularization for deep cnns with noisy labels”. In: *arXiv preprint arXiv:1511.07069* (2015).

[7] Michele Banko and Eric Brill. “Scaling to very very large corpora for natural language disambiguation”. In: *Proceedings of the 39th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2001, pp. 26–33.

[8] Ricardo Barandela and Eduardo Gasca. “Decontamination of Training Samples for Supervised Pattern Recognition Methods”. In: *Advances in Pattern Recognition: Joint IAPR International Workshops SSPR 2000 and SPR 2000 Alicante, Spain, August 30 – September 1, 2000 Proceedings*. Ed. by Francesc J. Ferri et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 621–630. ISBN: 978-3-540-44522-7. DOI: 10.1007/3-540-44522-6_64. URL: http://dx.doi.org/10.1007/3-540-44522-6_64.

[9] A Bauereiß, T Scharowsky, and C Körner. “Defect generation and propagation mechanism during additive manufacturing by selective beam melting”. In: *Journal of Materials Processing Technology* 214.11 (2014), pp. 2522–2528.

[10] Fabien Bayle and Maria Doubenskaia. “Selective Laser Melting process monitoring with high speed infra-red camera and pyrometer - art. no. 698505”. In: 6985 (Jan. 2008).

[11] Alan Joseph Bekker and Jacob Goldberger. “Training deep neural-networks based on unreliable labels”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 2682–2686.

[12] Sebastian Berumen et al. “Quality control of laser-and powder bed-based Additive Manufacturing (AM) technologies”. In: *Physics procedia* 5 (2010), pp. 617–622.

[13] Valmik Bhavar et al. “A review on powder bed fusion technology of metal additive manufacturing”. In:

[14] Avrim Blum and Tom Mitchell. “Combining labeled and unlabeled data with co-training”. In: *Proceedings of the eleventh annual conference on computational learning theory*. ACM. 1998, pp. 92–100.

[15] CD Boley et al. “Metal powder absorptivity: modeling and experiment”. In: *Applied optics* 55.23 (2016), pp. 6496–6500.

[16] Léon Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[17] Carla E. Brodley and Mark A. Friedl. “Identifying Mislabeled Training Data”. In: *J. Artif. Int. Res.* 11.1 (July 1999), pp. 131–167. ISSN: 1076-9757. URL: `http://dl.acm.org/citation.cfm?id=3013545.3013548`.

[18] Carla E Brodley and Mark A Friedl. “Identifying mislabeled training data”. In: *Journal of artificial intelligence research* 11 (1999), pp. 131–167.

[19] Gavin Brown, Jeremy L Wyatt, and Peter Tiňo. “Managing diversity in regression ensembles”. In: *Journal of machine learning research* 6.Sep (2005), pp. 1621–1650.

[20] Nicholas P Calta et al. “An instrument for in situ time-resolved X-ray imaging and diffraction of laser powder bed fusion additive manufacturing processes”. In: *Review of Scientific Instruments* 89.5 (2018), p. 055101.

[21] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. “Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]”. In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[22] Joon-Phil Choi et al. “Evaluation of powder layer density for the selective laser melting (SLM) process”. In: *Materials transactions* 58.2 (2017), pp. 294–297.

[23] Stijn Clijsters et al. “In situ quality control of the selective laser melting process using a high-speed, real-time melt pool monitoring system”. In: *The International Journal of Advanced Manufacturing Technology* 75.5-8 (2014), pp. 1089–1101.

[24] Tom Craeghs et al. “Detection of process failures in layerwise laser melting with optical process monitoring”. In: *Physics Procedia* 39 (2012), pp. 753–759.

[25] Pieter-Tjerk De Boer et al. “A tutorial on the cross-entropy method”. In: *Annals of operations research* 134.1 (2005), pp. 19–67.

[26] Tarasankar DebRoy et al. “Additive manufacturing of metallic components–process, structure and properties”. In: *Progress in Materials Science* 92 (2018), pp. 112–224.

[27] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[28] Philip J. DePond et al. “In situ measurements of layer roughness during laser powder bed fusion additive manufacturing using low coherence scanning interferometry”. In: *Materials & Design* 154 (2018), pp. 347–359. ISSN: 0264-1275. DOI: https://doi.org/10.1016/j.matdes.2018.05.050. URL: http://www.sciencedirect.com/science/article/pii/S0264127518304350.

[29] Luca Didaci, Giorgio Fumera, and Fabio Roli. “Diversity in classifier ensembles: Fertile concept or dead end?” In: *International Workshop on Multiple Classifier Systems*. Springer. 2013, pp. 37–48.

[30] Yifan Ding et al. “A Semi-Supervised Two-Stage Approach to Learning from Noisy Labels”. In: *arXiv preprint arXiv:1802.02679* (2018).

[31] M Erler et al. “Novel machine and measurement concept for micro machining by selective laser sintering”. In:

[32] Sarah K Everton et al. “Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing”. In: *Materials & Design* 95 (2016), pp. 431–445.

[33] Rob Fergus, Yair Weiss, and Antonio Torralba. “Semi-supervised learning in gigantic image collections”. In: *Advances in neural information processing systems*. 2009, pp. 522–530.

[34] BK Foster et al. “Optical, layerwise monitoring of powder bed fusion”. In: *Solid Free. Fabr. Symp. Proc.* 2015, pp. 295–307.

[35] Jason C Fox, Brandon M Lane, and Ho Yeung. “Measurement of process dynamics through coaxially aligned high speed near-infrared imaging in laser powder bed fusion additive manufacturing”. In: *Thermosense: Thermal Infrared Applications XXXIX*. Vol. 10214. International Society for Optics and Photonics. 2017, p. 1021407.

[36] Marianne M Francois et al. “Modeling of additive manufacturing processes for metals: Challenges and opportunities”. In: *Current Opinion in Solid State and Materials Science* 21.LA-UR-16-24513 (2017).

[37] William E Frazier. “Metal additive manufacturing: a review”. In: *Journal of Materials Engineering and Performance* 23.6 (2014), pp. 1917–1928.

[38] Andreas Gebhardt et al. “Additive manufacturing by selective laser melting the realizer desktop machine and its application for the dental industry”. In: *Physics Procedia* 5 (2010), pp. 543–549.

[39] Aritra Ghosh, Naresh Manwani, and PS Sastry. “Making risk minimization tolerant to label noise”. In: *Neurocomputing* 160 (2015), pp. 93–107.

[40] Giorgio Giacinto and Fabio Roli. “Design of effective neural network ensembles for image classification purposes”. In: *Image and Vision Computing* 19.9-10 (2001), pp. 699–707.

[41] Haijun Gong et al. “Influence of defects on mechanical properties of Ti–6Al–4 V components produced by selective laser melting and electron beam melting”. In: *Materials & Design* 86 (2015), pp. 545–554.

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[43] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[44] Marco Grasso and Bianca Maria Colosimo. “Process defects and in situ monitoring methods in metal powder bed fusion: a review”. In: *Measurement Science and Technology* 28.4 (2017), p. 044005.

[45] Marco Grasso et al. “In-process monitoring of selective laser melting: Spatial detection of defects via image data analysis”. In: *Journal of Manufacturing Science and Engineering* 139.5 (2017), p. 051001.

[46] M Grasso et al. “In situ monitoring of selective laser melting of zinc powder via infrared imaging of the process plume”. In: *Robotics and Computer-Integrated Manufacturing* 49 (2018), pp. 229–239.

[47] Alon Halevy, Peter Norvig, and Fernando Pereira. “The unreasonable effectiveness of data”. In: (2009).

[48] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[49] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[50] B. Heisele. “Visual object recognition with supervised learning”. In: *IEEE Intelligent Systems* 18.3 (May 2003), pp. 38–42. ISSN: 1541-1672. DOI: `10.1109/MIS.2003.1200726`.

[51] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (2006), pp. 504–507.

[52] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[53] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. “Spatial transformer networks”. In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[54] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. “Learning Deep Networks from Noisy Labels with Dropout Regularization”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE. 2016, pp. 967–972.

[55] Chandrika Kamath. “Data mining and statistical inference in selective laser melting”. In: *The International Journal of Advanced Manufacturing Technology* 86.5-8 (2016), pp. 1659–1677.

[56] Chandrika Kamath et al. “Density of additively-manufactured, 316L SS parts using laser powder-bed fusion at powers up to 400 W”. In: *The International Journal of Advanced Manufacturing Technology* 74.1-4 (2014), pp. 65–78.

[57] Jordan A Kanko, Allison P Sibley, and James M Fraser. “In situ morphology-based defect detection of selective laser melting through inline coherent imaging”. In: *Journal of Materials Processing Technology* 231 (2016), pp. 488–500.

[58] Saad A. Khairallah et al. “Laser powder-bed fusion additive manufacturing: Physics of complex melt flow and formation mechanisms of pores, spatter, and denudation zones”. In: *Acta Materialia* 108 (2016), pp. 36–45. ISSN: 1359-6454. DOI: `https://doi.org/10.1016/j.actamat.2016.02.014`. URL: `http://www.sciencedirect.com/science/article/pii/S135964541630088X`.

[59] Saad A Khairallah et al. “Laser powder-bed fusion additive manufacturing: Physics of complex melt flow and formation mechanisms of pores, spatter, and denudation zones”. In: *Acta Materialia* 108 (2016), pp. 36–45.

[60] Hoejin Kim, Yirong Lin, and Tzu-Liang Bill Tseng. “A review on quality control in additive manufacturing”. In: *Rapid Prototyping Journal* 24.3 (2018), pp. 645–669.

[61] Wayne E King et al. “Laser powder bed fusion additive manufacturing of metals; physics, computational, and materials challenges”. In: *Applied Physics Reviews* 2.4 (2015), p. 041304.

[62] Wayne King et al. “Overview of modelling and simulation of metal powder bed fusion process at Lawrence Livermore National Laboratory”. In: *Materials Science and Technology* 31.8 (2015), pp. 957–968.

[63] Diederik P Kingma et al. “Semi-supervised learning with deep generative models”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 3581–3589.

[64] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[65] Stefan Kleszczynski et al. “Error detection in laser beam melting systems by high resolution imaging”. In: *Proceedings of the Twenty Third Annual International Solid Freeform Fabrication Symposium*. Vol. 2012. 2012.

[66] Alex Krizhevsky. “Learning multiple layers of features from tiny images”. In: (2009).

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[68] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[69] Jean-Pierre Kruth et al. “On-line monitoring and process control in selective laser melting and laser cutting”. In:

[70] Abhishek Kumar and Hal Daumé. “A co-training approach for multi-view spectral clustering”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 393–400.

[71] Ludmila I Kuncheva and Christopher J Whitaker. “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy”. In: *Machine learning* 51.2 (2003), pp. 181–207.

[72] Chandrakanth Kusuma. “The effect of laser power and scan speed on melt pool characteristics of pure titanium and Ti-6Al-4V alloy for selective laser melting”. In: (2016).

[73] Chandrakanth Kusuma et al. “Effect of Laser Power and Scan Speed on Melt Pool Characteristics of Commercially Pure Titanium (CP-Ti)”. In: *Journal of Materials Engineering and Performance* 26.7 (2017), pp. 3560–3568.

[74] John Lafferty, Andrew McCallum, and Fernando CN Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: (2001).

[75] Samuli Laine and Timo Aila. “Temporal ensembling for semi-supervised learning”. In: *arXiv preprint arXiv:1610.02242* (2016).

[76] Yann LeCun. “The MNIST database of handwritten digits”. In: *http://yann.lecun.com/exdb/mnist/* (1998).

[77] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[78] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[79] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. “Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree”. In: *Artificial Intelligence and Statistics*. 2016, pp. 464–472.

[80] Dong-Hyun Lee. “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks”. In: *Workshop on Challenges in Representation Learning, ICML*. Vol. 3. 2013, p. 2.

[81] S Leuders et al. “On the mechanical behaviour of titanium alloy TiAl6V4 manufactured by selective laser melting: Fatigue resistance and crack growth performance”. In: *International Journal of Fatigue* 48 (2013), pp. 300–307.

[82] Anat Levin, Paul A Viola, and Yoav Freund. “Unsupervised Improvement of Visual Detectors using Co-Training.” In: *ICCV*. 2003, pp. 626–633.

[83] Zhongwei Li et al. “In Situ 3D Monitoring of Geometric Signatures in the Powder-Bed-Fusion Additive Manufacturing Process via Vision Sensing Methods”. In: *Sensors* 18.4 (2018). ISSN: 1424-8220. DOI: `10.3390/s18041180`. URL: `http://www.mdpi.com/1424-8220/18/4/1180`.

[84] Tongliang Liu and Dacheng Tao. “Classification with noisy labels by importance reweighting”. In: *IEEE Transactions on pattern analysis and machine intelligence* 38.3 (2016), pp. 447–461.

[85] Philipp Lott et al. “Design of an optical system for the in situ process monitoring of selective laser melting (SLM)”. In: *Physics Procedia* 12 (2011), pp. 683–690.

[86] AO Martins Luiz, LC Pádua Flávio, and EM Almeida Paulo. “Automatic detection of surface defects on rolled steel using computer vision and artificial neural networks”. In: *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society.* IEEE. 2010, pp. 1081–1086.

[87] Sonny Ly et al. “Metal vapor micro-jet controls material redistribution in laser powder bed fusion additive manufacturing”. In: *Scientific reports* 7.1 (2017), p. 4085.

[88] Daniel Maturana and Sebastian Scherer. “Voxnet: A 3d convolutional neural network for real-time object recognition”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 2015, pp. 922–928.

[89] Qiguang Miao et al. “RBoost: label noise-robust boosting algorithm based on a non-convex loss function and the numerically stable base learners”. In: *IEEE transactions on neural networks and learning systems* 27.11 (2016), pp. 2216–2228.

[90] Rada Mihalcea. “Co-training and self-training for word sense disambiguation”. In: *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*. 2004.

[91] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

[92] Volodymyr Mnih and Geoffrey Hinton. “Learning to Label Aerial Images from Noisy Data”. In: *Proceedings of the 29th Annual International Conference on Machine Learning (ICML 2012).* Edinburgh, Scotland, June 2012.

[93] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10).* 2010, pp. 807–814.

[94] Maryam M Najafabadi et al. “Deep learning applications and challenges in big data analytics”. In: *Journal of Big Data* 2.1 (2015), p. 1.

[95] Nagarajan Natarajan et al. “Learning with Noisy Labels”. In: *Neural Information Processing Systems (NIPS).* Dec. 2013.

[96] David F. Nettleton, Albert Orriols-Puig, and Albert Fornells. "A study of the effect of different types of noise on the precision of supervised learning techniques". In: *Artificial Intelligence Review* 33.4 (2010), pp. 275–306. ISSN: 1573-7462. DOI: `10.1007/s10462-010-9156-z`. URL: `http://dx.doi.org/10.1007/s10462-010-9156-z`.

[97] Kamal Nigam and Rayid Ghani. "Analyzing the effectiveness and applicability of co-training". In: *Proceedings of the ninth international conference on information and knowledge management.* ACM. 2000, pp. 86–93.

[98] David Opitz and Richard Maclin. "Popular ensemble methods: An empirical study". In: *Journal of artificial intelligence research* 11 (1999), pp. 169–198.

[99] Paul O'Regan et al. "Metal based additive layer manufacturing: variations, correlations and process control". In: *Procedia Computer Science* 96 (2016), pp. 216–224.

[100] Derek Partridge and Wojtek Krzanowski. "Software diversity: practical statistics for its measurement and exploitation". In: *Information and software technology* 39.10 (1997), pp. 707–717.

[101] Giorgio Patrini et al. "Making deep neural networks robust to label noise: a loss correction approach". In: *stat* 1050 (2017), p. 22.

[102] Tuomas Purtonen, Anne Kalliosaari, and Antti Salminen. "Monitoring and adaptive control of laser processes". In: *Physics Procedia* 56 (2014), pp. 1218–1231.

[103] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017, pp. 652–660.

[104] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[105] Pranav Rajpurkar et al. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (2016).

[106] Antti Rasmus et al. "Semi-supervised learning with ladder networks". In: *Advances in Neural Information Processing Systems.* 2015, pp. 3546–3554.

[107] Scott Reed et al. "Training Deep Neural Networks on Noisy Labels with Bootstrapping." In: (2014). URL: `http://search.ebscohost.com/login.aspx?direct=true&db=edsarx&AN=1412.6596&site=eds-live`.

[108] Giulia Repossini et al. "On the use of spatter signature for in-situ monitoring of Laser Powder Bed Fusion". In: *Additive Manufacturing* 16 (2017), pp. 35–48.

[109] Lior Rokach. "Ensemble-based classifiers". In: *Artificial Intelligence Review* 33.1-2 (2010), pp. 1–39.

[110] David Rolnick et al. "Deep learning is robust to massive label noise". In: *arXiv preprint arXiv:1705.10694* (2017).

[111] A Rubenchik et al. “Direct measurements of temperature-dependent laser absorptivity of metal powders”. In: *Applied optics* 54.24 (2015), pp. 7230–7233.

[112] Tim Salimans et al. “Improved techniques for training GANs”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.

[113] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[114] Luke Scime and Jack. Beuth. “A Multi-scale Convolutional Neural Network for Autonomous Anomaly Detection and Classification in a Laser Powder Bed Fusion Additive Manufacturing Process”. In: *Additive Manufacturing* 24.5786 (2018), pp. 504–507.

[115] Luke Scime and Jack Beuth. “Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm”. In: *Additive Manufacturing* 19 (2018), pp. 114–126.

[116] Pierre Sermanet et al. “Overfeat: Integrated recognition, localization and detection using convolutional networks”. In: *http://arxiv.org/abs/1312.6229* ().

[117] Amanda JC Sharkey and Noel E Sharkey. “Combining diverse neural nets”. In: *The Knowledge Engineering Review* 12.3 (1997), pp. 231–247.

[118] Sergey A Shevchik et al. “Acoustic emission for in situ quality monitoring in additive manufacturing using spectral convolutional neural networks”. In: *Additive Manufacturing* 21 (2018), pp. 598–604.

[119] John Shore and Rodney Johnson. “Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy”. In: *IEEE Transactions on information theory* 26.1 (1980), pp. 26–37.

[120] P Sibi, S Allwyn Jones, and P Siddarth. “Analysis of different activation functions using back propagation neural networks”. In: *Journal of Theoretical and Applied Information Technology* 47.3 (2013), pp. 1264–1268.

[121] Sara Silva et al. “A semi-supervised Genetic Programming method for dealing with noisy labels and hidden overfitting”. In: *Swarm and Evolutionary Computation* (2017).

[122] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[123] John A Slotwinski et al. “Characterization of metal powders used for additive manufacturing”. In: *Journal of research of the National Institute of Standards and Technology* 119 (2014), p. 460.

[124] Thomas G Spears and Scott A Gold. “In-process sensing in selective laser melting (SLM) additive manufacturing”. In: *Integrating Materials and Manufacturing Innovation* 5.1 (2016), p. 2.

[125] Nitish Srivastava et al. “Dropout: A simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[126] Sainbayar Sukhbaatar et al. “Training convolutional networks with noisy labels”. In: *arXiv preprint arXiv:1406.2080* (2014).

[127] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[128] Gustavo Tapia et al. “Gaussian process-based surrogate modeling framework for process planning in laser powder-bed fusion additive manufacturing of 316L stainless steel”. In: *The International Journal of Advanced Manufacturing Technology* 94.9-12 (2018), pp. 3591–3603.

[129] Arash Vahdat. “Toward robustness against label noise in training deep discriminative neural networks”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5601–5610.

[130] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. “Learning with symmetric label noise: The importance of being unhinged”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 10–18.

[131] Andreas Veit, Michael J Wilber, and Serge Belongie. “Residual networks behave like ensembles of relatively shallow networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 550–558.

[132] Andreas Veit et al. “Learning From Noisy Large-Scale Datasets With Minimal Supervision”. In: *CoRR* abs/1701.01619 (2017).

[133] Xiaojun Wan. “Co-training for cross-lingual sentiment classification”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-volume 1*. Association for Computational Linguistics. 2009, pp. 235–243.

[134] K Wasmer et al. “In situ and real-time monitoring of powder-bed AM by combining acoustic emission and artificial intelligence”. In: *International Conference on Additive Manufacturing in Products and Applications*. Springer. 2017, pp. 200–209.

[135] Tong Xiao et al. “Learning from massive noisy labeled data for image classification”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 2691–2699.

[136] I Yadroitsev, Ph Bertrand, and I Smurov. “Parametric analysis of the selective laser melting process”. In: *Applied surface science* 253.19 (2007), pp. 8064–8069.

[137] I Yadroitsev et al. “Single track formation in selective laser melting of metal powders”. In: *Journal of Materials Processing Technology* 210.12 (2010), pp. 1624–1631.

[138] Bing Yao et al. "Multifractal analysis of image profiles for the characterization and detection of defects in additive manufacturing". In: *Journal of Manufacturing Science and Engineering* 140.3 (2018), p. 031014.

[139] Chor Yen Yap et al. "Review of selective laser melting: Materials and applications". In: *Applied physics reviews* 2.4 (2015), p. 041101.

[140] David Yarowsky. "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods". In: *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*. ACL '95. Cambridge, Massachusetts: Association for Computational Linguistics, 1995, pp. 189–196. DOI: `10.3115/981658.981684`. URL: `https://doi.org/10.3115/981658.981684`.

[141] Bodi Yuan et al. "Iterative Cross Learning on Noisy Labels". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), pp. 757–765.

[142] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[143] Xiaojin Zhu. *Semi-Supervised Learning Literature Survey*. Tech. rep. 1530. Computer Sciences, University of Wisconsin-Madison, 2005.

[144] Xingquan Zhu and Xindong Wu. "Class noise vs. attribute noise: A quantitative study". In: *Artificial intelligence review* 22.3 (2004), pp. 177–210.