Exploring deep generative modeling approaches with 2D to 3D polycrystalline microstructures generation

Master Thesis

by

Sree Sameer Kumtamukkula

A report submitted in partial fulfillment
of the requirements of Materials Science and Simulation
for obtaining the academic degree Master of Science (M.Sc.)

conducted at
Deutsches Zentrum für Luft- und Raumfahrt (DLR), Augsburg.

and submitted at
Interdisciplinary Centre for Advanced Materials Simulation,
Ruhr-Universität Bochum.

Supervisors

Prof. Dr. Markus Stricker

Dr. Yury Lysogorskiy

14th October, 2021

PREAMBLE

This work is a part of an ongoing project KILu (KI in der Luftfahrtforschung) that belongs to the institute of Test and Simulation for Gasturbines in Augsburg, an institute of Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR). The thesis was conceptualized and designed by Dr. Helal Chowdhury from DLR. The work conducted at DLR as part of KILu activities, and co-supervised by Dr. Helal Chowdhury from DLR, Prof. Dr. Markus Stricker and Dr. Yury Lysogorskiy from the Ruhr-University of Bochum.

For the completeness of the report, a number of ideas, algorithm, figures, pipelines are taken from KILu works. For example, sampling the data-space, data generation strategy and related DREAM.3D pipelines were developed by Dr. Chowdhury (DLR). The author has automated the workflow in python and generated necessary dataset for the training and validation of the machine learning models. Similarly, the algorithm for introducing lamellar-grains into polycrystalline globular microstructure is an early work of Dr. Chowdhury. The author of this thesis implemented and validated (using Python) a specific version of the algorithm.

For the selection of generative models, the idea of Unet and pix2pixGAN for the generation of microstructures comes from DLR-side. The initial model architectures including code-base were supplied by PI GmbH as a third-party partner of the project KILu. In this context, the author implemented conditional variational autoencoders, conditional generative adversarial networks using custom loss functions and Wasserstein loss. The idea of DREAM.3D postprocessing pipeline was suggested by the university supervisors, Prof. Dr. Markus Stricker and Dr. Yury Lysogorskiy as part of thesis work, and agreed by DLR. The initial structure of the report was also proposed by DLR.

All relevant figures/pipelines that comes from KILu works are marked with *.

Statutory Declaration

I declare that I completed this work on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this, nor a similar work, has been published or presented to an examination committee.

Sree Sameer Kumtamukkula

Restricted handling

This work is a part of Deutsches Zentrum für Luft- und Raumfahrt (DLR) internal project KILu (KI in Luftforschung). Since similar activities will be continued within DLR till the end of the project (October 2023), it is recommended to handle this report confidentiality in case of distribution, till the end of KILu.

Acknowledgment

Firstly, I would like to thank Dr. Helal Chowdhury for giving me this opportunity to work in Deutsches Zentrum für Luft- und Raumfahrt, Institute for Test and Simulation of gas turbines, Augsburg and providing necessary support.

I would also like to thank Prof. Dr. Markus Stricker for his guidance, support and valuable suggestions regarding the work. It is my pleasure to express my gratitude to Dr. Yury Lysogorskiy for his insights, which helped to shape this work. Their interactions provided me a scope for learning and motivated me to think in a different perspective.

My gratitude goes to ICAMS, Mrs. Jutta Kellermann and everyone involved with the masters program *Materials Science and Simulation*, who contributed immensely during my masters.

I would like to mention colleagues from PI GmbH, Dr. Kevin Cremanns and Dr. Can Bogoclu for their assistance in developing machine learning models.

Additionally, I extend my thanks to Pavan, Krishna, Praneeth, Harsha, Agni Raja, Murali, Dhanunjay and Priyatham for their engaging discussions and suggestions related to my research.

Besides, I am also grateful to Sahitya, Mouryarag, Abhilash, Ragadeep, Indu and Kaushik for supporting me throughout my master studies.

Finally, I would like to thank my parents and family for their unconditional love and support.

Contents

List of Figures

List of Tables

Abbreviations

cGANs conditional generative adversarial networks

CNNs convolutional neural networks

cVAE conditional variational autoencoder

DBM deep belief networks

DCGANs deep convolutional generative adversarial networks

EA0 first component of Euler angles

EA1 second component of Euler angles

EA2 third component of Euler angles

EBSD electron backscatter diffraction

ESD equivalent sphere diameter

FEA finite element analysis

GANs generative adversarial networks

JSD Jensen-Shannon divergence

KDE kernel density estimation

KLD Kullback-Leibler divergence

MAE mean absolute error

MLP multilayer perceptron

ODF orientation distribution function

PDF probability density function

PSP process-structure-property

RBM restricted Boltzmann machines

RSA random sequential addition

RVEs representative volume elements

SERVEs statistically equivalent representative volume element

VAEs variational autoencoders

WCGAN Wasserstein conditional generative adversarial networks

WGAN Wasserstein generative adversarial networks

Abstract

Microstructure reconstruction has been a significant area of research due to its direct usage in multiscale modeling and micro-mechanical simulations. However generation of 3D microstructures using experimental methods such as FIB-SEM or X-Ray tomography is a tedious process, so alternative approaches have been developed for reconstructing 3D microstructures. In the present work, we attempt to generate high-fidelity 3D microstructures from 2D images using deep generative modeling approaches such as variational autoencoders (VAEs) and generative adversarial networks (GANs). Since obtaining experimental images for training generative models is difficult and expensive, synthetic microstructures generated by DREAM.3D have been employed for this purpose. The generative model predicted 3D microstructures, are statistically compared with their real synthetic counterparts through microstructure-descriptors used previously for characterizing them. The generative models have been trained on the given data to efficiently generate 3D microstructures by employing Unet, conditional variational autoencoders (cVAEs), pix2pixGAN, and Wasserstein conditional generative adversarial networks (WCGANs) frameworks. The trained Unet model resulted in overfitting, while the rest were successful in the generalization of data. The trained pix2pixGAN model has been able to learn the grain structure but is not capable of capturing grain boundaries and the underlying statistics entirely. Postprocessing techniques are necessitated for optimization and the formation of a perfect grain-like structure. Besides an algorithm has been developed as a postprocessing tool to increase the complexity of synthetic microstructures by introducing lamellae.

Chapter 1

Introduction

1.1 Motivation

Currently, we are in the fourth paradigm of science which is data-driven, after empirical, theoretical, and computational paradigms [1, 2]. In the computational paradigm, modeling of experiments and simulations were developed in materials science to expedite material discovery and improve the existing high-performance materials. A new field known as materials informatics [3, 4] has been coined for integrating the techniques from data science with materials science for representation, parsing, storing and management of material knowledge systems [5]. In the present data-driven paradigm, quantifying the microstructure of the complex material has been a primary component for discovering process-structure-property (PSP) linkages for interlinking different length scales, and understand the influence of microstructure on its properties and performance [6, 7, 8, 9, 10, 11, 12]. Data-driven approaches involve the utilization of tools for a better perception of the information obtained from the previous three paradigms to develop novel techniques for improved performance.

The microstructural space of a materials system is interdependent on its manufacturing routes and plays a prominent role in governing its properties and performance. Microstructure lies at the core of both process-structure and structure-property relationship, where various 2D and 3D characterization techniques provide a better understanding of the whole PSP process-chain. Numerous experimental and computational methods have been developed for this purpose and a vast amount of research is focused on computational methods for obtaining high fidelity of experimental data [13]. Modeling of microstructures, particularly 3D ones has been trending as they are employed in multiscale simulations by serving as representative volume elements (RVEs) in finite element analysis (FEA) thereby bridging the gap between microscale and macroscale. Acquirement of high-quality 3D microstructures involves the utilization of various tomography techniques which are tedious and expensive [14, 15]. For this reason, reconstruction algorithms for the generation of synthetic microstructures are an active area of research. Advanced algorithms for synthetic microstructure generation usually comprise probabilistic methods such as

spatial tessellation algorithms and random sequential addition (RSA) [16, 17, 18]. Initially, a microstructure is quantified in form of descriptors for characterization, which are further employed for 3D reconstruction purposes. Due to the advent of the data-driven paradigm, deep generative models are also employed for reconstruction purposes. Generative modeling has been trending after the invention of generative adversarial networks [19] and the development of variational autoencoders [20] for generating artificial images. Advancements in this field opened a wide range of possibilities in various domains which were earlier believed to be impossible [21, 22, 23].

This thesis was conducted at Deutsches Zentrum für Luft- und Raumfahrt, Institute for Test and Simulation of gas turbines, Augsburg where the main focus is on developing novel aero-engine technologies. Titanium aluminide alloys which are widely used in gas turbines are considered as the material of interest in this work. Here, we attempt to generate statistically equivalent 3D microstructures from 2D images by employing deep conditional generative models. Synthetic microstructures generated by DREAM.3D are employed for gathering data for training. This is advantageous in terms of encompassing all the variations possible in practical microstructures. Microstructure-descriptors such as grain size, volume fraction, and crystallographic orientation of grains are utilized for producing synthetic 3D and 2D microstructures. The main objective of this work is to develop a deep generative model framework for generating statistically equivalent 3D microstructures from 2D such that the statistical information in 2D image is also retained in the 3D microstructure. Besides, the reconstructed microstructures should not be deterministic and have clearly demarcated grains.

1.2 Structure of the report

This report is organized in the following chapters. Chapter. 2 gives an overview of the materials and theory necessary for achieving the objective of this work. The next chapter, Chapter. 3 discusses the data generation strategies and workflow. Chapter. 4 showcases the configurations of each model employed in this work. Chapter. 5 presents the results obtained by training the configured models and discusses the observations. Chapter. 6 introduces the postprocessing pipeline and Chapter. 7 concludes the thesis with a summary and suggestions for future work.

Chapter 2

Materials and Methods

2.1 Materials

In this work, duplex titanium aluminide alloys are considered as the material of interest. They constitute γ-TiAl and a comparatively smaller amount of Ti_3Al in the form of lamellar and equiaxed α_2 grains [24]. These alloys are widely used in automotive and aerospace domains mainly due to their low density and high creep resistance. Intermetallic TiAl (Ti-47Al-2Cr-2Nb) has been originally used in low pressure turbine blades (700°C) in General Electric (GE), GEnx-1B for the Boeing 787 Dreamliner increasing its fuel efficiency (which is mainly owed to its low weight ($\approx$ 4g.cm^{-3}) compared to nickel-based superalloys ($\approx$ 8-9 g.cm^{-3})) [25]. Generally γ-TiAl alloys have three distinct microstructures namely nearly lamellar (contain globular γ-TiAl with small percent of α_2-Ti_3Al grains), fully lamellar (comprises γ-TiAl and α_2-Ti_3Al lamellae) and duplex (lamellae colonies of γ-TiAl and α_2-Ti_3Al globular grains) [26].

The duplex alloys were obtained as a middle ground between nearly and fully lamellar during improving performance by application of heat treatment processes for the reduction of coarse structure. The resultant duplex microstructure constitute γ (ordered tetragonal, L10) and $\gamma + \alpha_2$ (ordered hexagonal, D019) lamellar colonies.

In comparison with complete lamellar, the nearly lamellar and duplex alloys have low fracture toughness and creep resistance but better ductility and strength at room and high temperatures [27]. The family of γ-TiAl alloys is generally alloyed with Nb, Cr, B, C for improved mechanical properties in strength, creep resistance, and also oxidation resistance. Achieving duplex TiAl microstructure is a complex process as direct solidification yields structural inhomogeneities in TiAl and also directional cooling results in a strong texture which is hard to remove via further heat treatments [28, 29, 30, 31]. These mechanical properties are influenced by the microstructure parameters such as mean grain diameter, the volume fraction of lamellar grains, and inter-lamellar spacing which are further dependant on the heat treatment methods employed during its preparation [32]. The phase diagram of the TiAl alloy system is depicted in Fig. 2.1 and in the present work, the composition of Al in the TiAl alloy system is chosen to be 43-48 %. It can be inferred from Fig. 2.1, that at

this selected composition, the TiAl alloy microstructure consists of α_2 and γ phases.

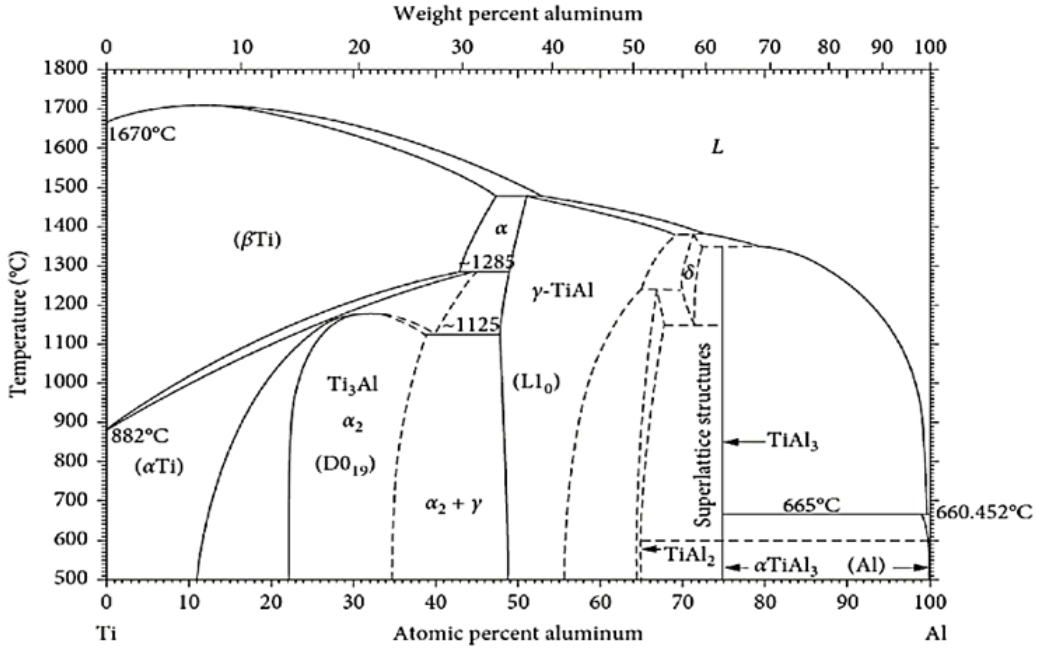


Figure 2.1: Phase diagram of TiAl alloy system [33].

2.2 3D Microstructures in multiscale modeling

Multiscale modeling constitutes hierarchical modeling at different length scales for the accurate prediction of its properties. The 3D volumes of microstructures are quintessential in multiscale modeling as they are employed as RVEs in crystal plasticity finite element analysis and homogenization methods which plays a major role in continuum mechanics. This idea of utilizing a small part of a material as an average representation of complete material was first introduced in 1963 by Hill [34] and further developed by Hashin and Shtrikman [35, 36] who proposed considering the morphology of RVEs as a reference cube. RVEs should be a miniature of the complete structure and exhibit properties (i.e., volume fraction, stresses, and strains) identical to the whole. The primary assumption in RVEs is that the internal spatial alignment is periodic and a single unit of RVEs should carry all the required information, representing the whole geometry [37]. In cases where there are severe structural inhomogeneities or gradient microstructures, the underlying assumptions in RVEs couldn't accurately model the relationship between failure predictions and experiment results.

Furthermore, Willis [38] initiated a novel strategy of using a two-point probability function as statistically equivalent representative volume element (SERVEs) to incorporate the arbitrary deviations in composite microstructures. Since all arbitrary RVEs are not

able to accommodate the microstructures with irregularities or complexities, SERVEs can be applied to determine precise homogeneous attributes and can be further employed in micro-macro modeling [39]. As the name signifies SERVEs comprise essential statistics representing the complete microstructure i.e., local distribution functions of descriptors of internal structure (such as nearest neighbor distances, volume fraction, radial distributions) should be similar to the whole. The ergodic hypothesis states that; supervising a stochastic system for a long time results in the same statistical measures when a single independent instance of that system is considered over that time [40, 8]. This permits scrutinizing a single randomly chosen instance from an ensemble, for calculating the average function over the complete space by relating it to the volumetric average. Fig. 2.2 illustrates where a point in the macrostructure materializes into a microstructure with subtle details and ergodic hypothesis is invoked to solve the problem of RVEs with the local boundary conditions.

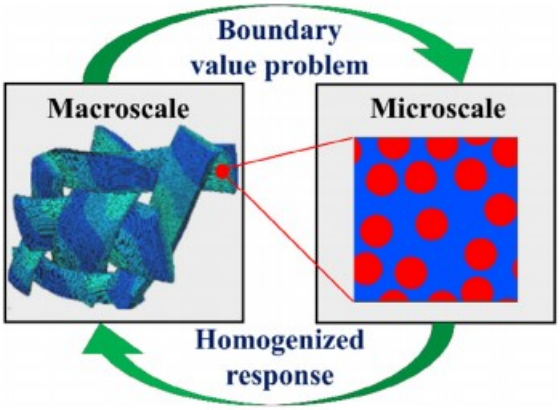


Figure 2.2: Multiscale finite element method (FEM) for interlinking microscale and macroscale using top-down (homogenization techniques) and bottom-up (Boundary value problems) approaches [40].

2.3 Methods for microstructure characterization and reconstruction

Microstructure characterization and reconstruction comprise quantifying microstructure by exploiting accessible parameters and further employing them for reconstructing a 3D microstructure. It has several applications in materials modeling such as determining a constitutive law for characterizing a homogenized response of heterogeneous material, develop material systems with desirable properties, and so on [40]. The randomness in a microstructure gives rise to probabilistic material properties and performance which provides a need for preserving their statistical behavior. As such, an ensemble of microstructures is necessary for capturing the randomness and predicting their properties (by finite element method) and further construct PSP linkages.

2.3.1 Microstructure characterisation using descriptors

Microstructure characterization involves determining parameters from a microstructure that provide a physical significance and can be used for quantitative comparison. Quantifying the internal structure of material using the descriptors is a cumbersome process as they should efficiently represent an ensemble of 3D cross-sections extracted from the specimen. So, they should inevitably be statistical as they must capture the distributions of properties from the internal structure of the material [8, 41, 42, 43, 44, 45]. Microstructure-descriptors are the parameters defined for quantifying microstructure in a consistent manner and comprehend the relationship between material internal structure and its properties and also effective for quantitative comparison [46]. In simple terms, these are the functions that represent a microstructure and are selected in such a manner that a minimum number of descriptors encompasses a majority of information of microstructure. The microstructure-descriptors are classified into statistical and physical. The statistical descriptors capture the spatial correlations between different points in a microstructure while the physical descriptors utilize the physically comprehensive functions in one, two, and higher dimensional spaces. The selection of descriptors is a heuristic process, mainly dependant on the type of material and specified application [40]. In general, the aggregate values of descriptors are determined for quantifying and designating a microstructure [47]. The information from these descriptors is conveyed to formulate a reconstruction algorithm capable of generating required statistically equivalent microstructures. The fidelity of the generated microstructures is then calibrated with mechanical simulations to determine inconsistencies.

2.3.1.1 Statistical descriptors

Statistical descriptors comprise two-point correlation functions, lineal path functions, cluster correlation functions, and so on. Reconstruction using statistical descriptors is principally a stochastic optimization problem by modifying an initial image to minimize a cost function. Yeong and Torquato (YT) method is one of the prominent optimization methods for reconstruction using statistical descriptors. Intuitively, correlation functions define correlations between random values. For instance, N random points are dispersed in a microstructure for an adequate number of times, and the probability of these points falling into a phase P_i have been calculated and the statistical probability function is termed as one-point correlation function.

$$P_i^1 = \frac{N_i}{N}\Big|_{N\to\infty} = v_i \tag{2.1}$$

This equation is analogous to calculating volume fraction of phase i. Volume fraction of phase i is calculated by the formula

$$v_\mathrm{i} = V_\mathrm{i}/V_\mathrm{tot}$$

where V_i denotes the volume of phase i and sum of all volume fractions should be unity.

In a similar manner, instead of N random points, N random line vectors with length r are thrown simultaneously in a microstructure, then the probability that both the ends of line vector fall on a particular phase x and y respectively is denoted as Two-point correlation functions [48].

$$P^2_{xy}(\bar{r}) = \frac{N_{xy}}{N}\bigg|_{N\to\infty}, \quad \bar{r} = |r_y - r_x|, r_x \in x \text{ and } r_y \in y \tag{2.2}$$

here r_x and r_y are the edges of the line vector and present in phase x and phase y respectively. Compared to one-point correlations, they provide further information and are normalized by two constraints. The first is that the sum of all feasible probabilities adds up to one and the second constraint is that the probability function should be symmetric [49]. Two-point correlation functions provided a major breakthrough in microstructure characterization as they provide an elegant way of describing microstructure [50, 51]. The stochastic nature of this function is rewarding in precisely characterizing an ensemble of 3D microstructures to the corresponding 2D images. The success of two-point correlations led to the development of stochastic algorithms for optimizing the difference in correlations between 2D and 3D microstructures. But this method is not completely foolproof, as the degeneracy of the two-point correlation function resulted in impractical results [52]. For more information related to statistical functions kindly refer [8, 49].

2.3.1.2 Physical descriptors

The main motivation for choosing physical descriptors is the existence of a large number of descriptors and strong correlations between them [5]. They give a better intuition, and are classified into deterministic (characterized using a single numerical value) or statistical (characterized by a distribution function). The descriptors such as volume fraction, surface area, and the number of clusters pertain to deterministic category. Whilst the latter comprises of nearest cluster center distance, orientation angles, and geometry-based descriptors such as aspect ratios, pore sizes and so on. They are further divided into high-level descriptors (such as volume fraction) which designates for the whole microstructure and low-level descriptors (such as grain size) are assigned to individual grains.

Initially, image segmentation techniques were widely employed for microstructure characterization to determine objects in the microstructure such as lines, clusters and edges. It is followed by the analysis of the aforementioned objects to retrieve the magnitude and distribution of deterministic and statistical descriptors respectively. In order to achieve high fidelity of microstructures, the selected descriptors should not be correlated and supply adequate information to eliminate uncertainty in generating PSP linkages. Stereological methods can be utilized to enable reconstruction of 3D microstructures from 2D images by making certain assumptions of geometrical features in 3D and determine various 3D objects from the corresponding 2D objects [53, 54]. In polycrystalline materials, grain-related features such as grain size, shape, and crystallographic orientation are extensively employed as descriptors and their influence on properties is thoroughly

investigated [55]. Finally, reconstruction of 3D microstructure is achieved by evolving the initial structure to predict its descriptors to that of the target ones.

In this work, a processed image from electron backscatter diffraction (EBSD) is employed. This image depicts contrast at each individual pixel as the intensities are based on their individual diffraction or scattering events [56]. So, in order to reconstruct a 3D microstructure from the given 2D image; volume fraction, equivalent sphere diameter (in microns), and Euler angles in orientation distribution function (ODF) are selected as microstructure-descriptors. The volume fraction is chosen due to its capability in representing a homogenized property of material while ODF and equivalent sphere diameter capture the heterogeneities across the microstructure. The ODF denotes crystallographic texture in each grain and equivalent sphere diameter represents the diameter of a sphere with equivalent volume (generally defined for an irregularly shaped object). The chosen descriptors play a significant role in determining the structure-property (S-P) linkages as they are known to be relevant for the mechanical properties of polycrystalline materials. Firstly, volume fraction determines the formation of phases in the resultant microstructure which directly controls the mechanical properties of the material. Secondly, mean equivalent sphere diameter (ESD) affects the strengthening mechanisms by grain boundaries as the strength of alloys decreases with grain size [57]. Finally, ODF gives the information regarding misorientations that controls the texture or topography of the material thereby defining the isotropic and non-isotropic behavior of the materials. The volume fraction of 3D and 2D can be compared directly by calculating the area of respective phases, but for a quantitative comparison of grain diameter and ODF, the distributions of the Euler angles in the resultant 2D and 3D needs to be analyzed. When visualizing the Euler angle components in ODF, the EA0, third component of Euler angles (EA2) exhibited a uniform distribution while EA1 displayed a Gaussian distribution. So, EA1 is particularly selected and customized accordingly for synthetic data generation.

2.3.2 Microstructure reconstruction approaches

Synthesis of 3D microstructures from experimental methods is a time consuming and tedious process as experimental techniques consist of serial sectioning techniques or utilizing X-Ray tomography [58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]. Serial sectioning techniques involve automatic progressive slicing followed by scanning them using appropriate microscopic methods. Regardless of the above mentioned disadvantages, these techniques are not employed due to their inadequacy of providing a comprehensive representation of the entire samples [71].Therefore there is a need for the development of computer-aided 3D reconstruction methods from statistical information. Since 2D microstructure images are easily obtained from the experimental image analysis techniques, they can form a basis for 3D reconstruction.

This 3D reconstruction from specific 2D microstructural information has been a classic inverse problem with numerous applications in improving material design and the solution for this complicated task is non-unique [72]. Initially, Gaussian filtering methods using correlation functions were employed for this purpose [73, 74]. Yeong et al. [75] applied

optimization algorithms for 3D reconstruction by employing two-point correlation functions as a descriptor for 2D slices. Following this, simulated annealing algorithms, support vector machines and Monte Carlo methods were employed for 3D microstructure reconstruction [76, 46, 77]. The accuracy of the simulated 3D reconstructions depends mainly on the chosen set of descriptors, and how efficiently they are able to represent the microstructure.

Quey et al. [16] presented an efficient approach for synthetic microstructure generation using spatial tessellation algorithms which is quite effective for simple microstructures. Random sequential addition (RSA) is an update to the tessellation methods which has the capability to produce microstructures with convex and non-convex grain structures but has a limitation regarding the requirement of large computational resources [17, 18]. Prasad et al. [78] developed a python package for generating complex polycrystalline microstructures by employing a collision-driven particle dynamics approach. In annealing algorithms, particle size distribution obtained from 2D image is employed for emulation of powder processing [53]. These algorithms employ statistical distributions related to grain size distributions, aspect ratios, neighbor distances, and orientation distribution functions for simulating grain growth in 3D (depicted in Fig. 3.2). This generalized method is mainly restricted to polycrystalline materials and furthermore, there is a loss of geometric data as irregular shapes are generalized as ellipsoids, and complex grain shapes are defined using mean diameters and aspect ratios. The left part in Fig. 2.3 illustrates 3D synthetic microstructure of $192 \times 192 \times 192$ obtained from DREAM.3D and the right part is the 2D slices obtained by slicing the 3D microstructure along $x = [1, 48, 96, 144, 192]$.

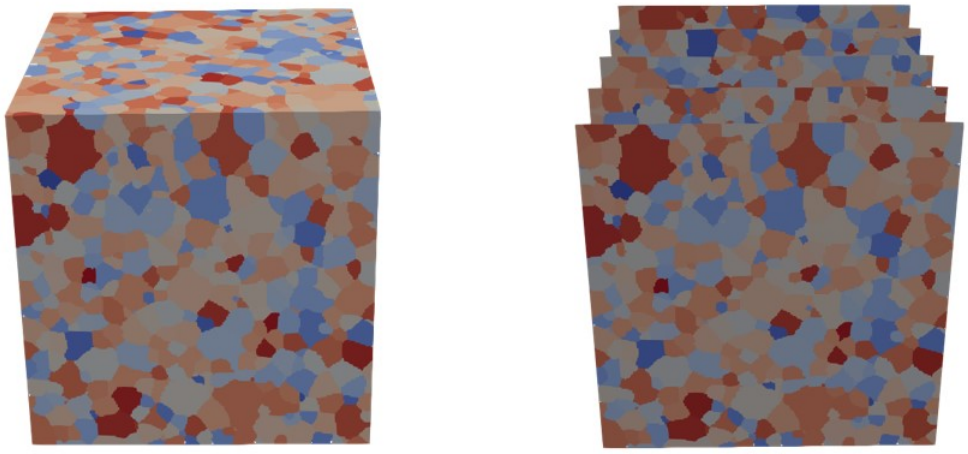


Figure 2.3: Synthetic microstructures obtained from DREAM.3D.

Lately, deep learning based generative models such as variational autoencoders [79] and

generative adversarial networks [80] have spiked the interest of material scientists exploiting their potentialities in the synthesis of artificial microstructures. These generative models are more generalized compared to the previously mentioned reconstruction methods as they learn the underlying distributions of the given dataset. Initial works in this domain include Tang et al. [81] employing GANs for microstructure material design, Cang et al. [82] utilized variational autoencoders (VAEs) for generating artificial microstructure samples of sandstones and Singh et al. [83] applied Wasserstein GANs for a two-phase heterogeneous isotropic material. Iyer et al. [84], used conditional Wasserstein GANs with gradient penalty for generating microstructures based on the given processing method, and Fokina et al. [85] employed StyleGAN architecture for the synthesis of multiphase gray-scale microstructures. Mosser et al. [86, 87] applied GANs for generating 3D representative volumetric models of limestone. Gayon-Lombardo et al. [88] generated n-phase 3D microstructures of solid oxide fuel cell by using deep convolutional generative adversarial networks (DCGANs). Recent works include microstructure generation of heterogeneous and topologically complex materials using GANs by Hsu et al. [89] and developing a new GAN framework termed as sliceGAN by Kench et al. [90]. In the present work, deep learning based generative models such as GANs and VAEs, and Unet have been employed for microstructure reconstruction.

2.4 State of the art generative modeling methods

Machine learning (ML) has reached a level where it is not only capable of learning significant information from high dimensional data and predicting outcomes, but also proficient in learning distributions for generating new data from the underlying data distribution. These models primarily involve generating new data from existing data by learning intrinsic patterns and are termed, generative models. Generative modeling constitutes a statistical model for determining a probabilistic distribution $P(X,Y)$ between observations X and targets Y and another model for computing conditional probability $P(Y|X=x)$ whether a given observation x belongs to a target label Y. The first model is called the generative model while the latter is termed the discriminative model. In simpler terms, generative models learn the probability distributions of the target data and generate data while discriminative models ensure that the generated data complements underlying distributions. There are several methods in this domain namely restricted Boltzmann machines (RBM) [91, 92], deep Boltzmann machines [93], deep belief networks (DBM) [92], Bayesian networks [94], GANs [80] and VAEs [79, 20] have been employed. A brief introduction to the methods implemented in this work is described in the subsequent sections.

2.4.1 Unet

Unet was initially invented for biomedical image segmentation by Ronnenberger et al. at Computer Science Department and BIOSS Center for Biological Signalling Studies, the

University of Freiburg [95], it mainly performs segmentation tasks in images by providing class labels to each pixel. Unet framework is similar to autoencoder with an encoder-decoder assembly but in contrast to autoencoder, the decoder cannot be stand-alone generative model due to the employment of skip connections. Its name comes from its architecture which consists of a symmetrical contracting path (downsampling) and expansive path (upsampling) which forms a U-shaped architecture. Additionally, skip connections are employed between the downsampling (encoding) and upsampling (decoding) paths to provide access to the elementary features obtained previously from the encoder. The Unet architecture is illustrated in image 2.4. In this network, the input traverses through a series of layers that gradually decreases dimensions, i.e., downsampling path, until it reaches the bottleneck layer where the process is inverted, i.e., the dimensions are increased until it reaches the required size which is termed as an upsampling path. In this whole process, it is essential that a fair amount of information from the lower levels should be passed on to higher levels directly, and skip connections are mainly employed for this reason. In this work, the Unet is modified accordingly to generate a 3D microstructure from a 2D microstructure.

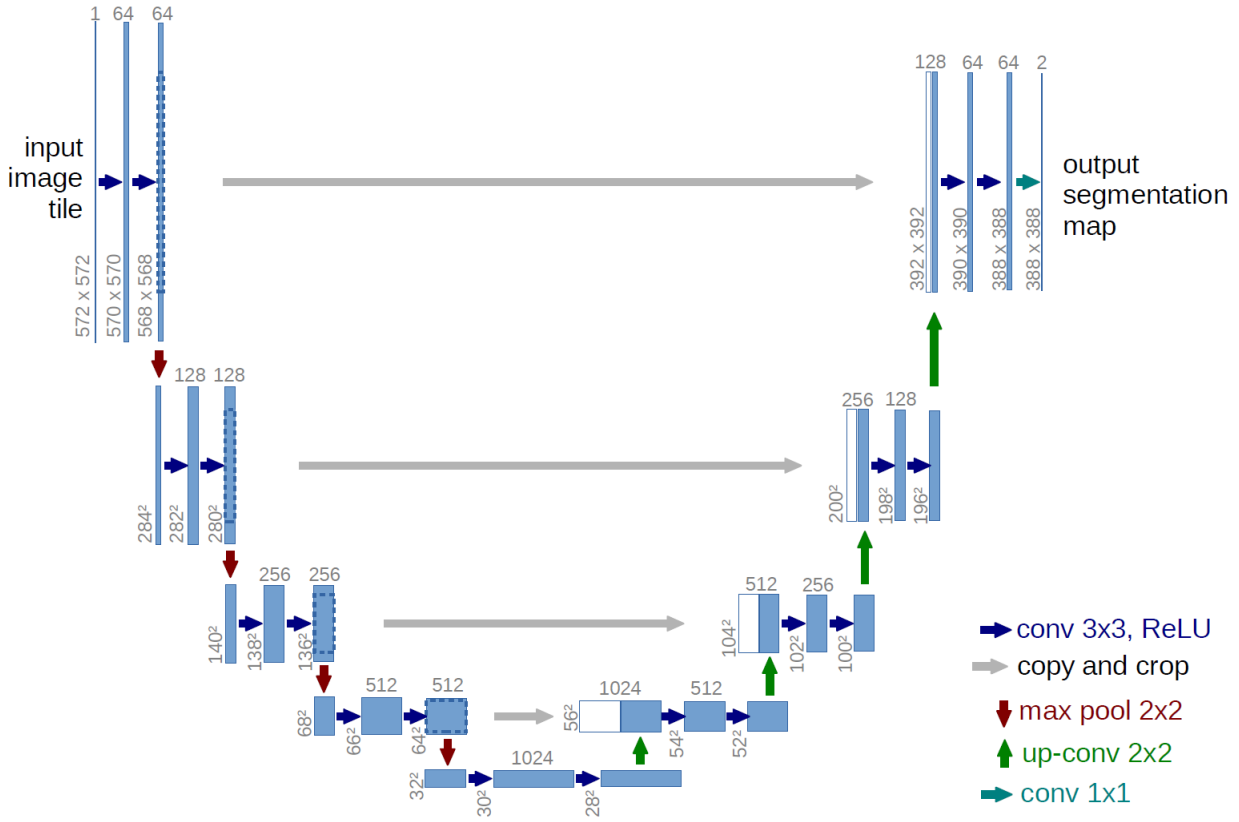


Figure 2.4: Unet architecture from original paper [95].

2.4.2 Variational autoencoders (VAEs)

In general, an autoencoder is a combination of encoder and decoder where encoder transforms the given feature space into lower dimensions while the latter reconstructs them to original dimensions [96, 97]. In deep learning regime, encoder and decoder can be represented by neural networks and this model can be functional for applications such as dimensionality reduction and denoising of images [98]. For instance, autoencoders can be used for dimensionality reduction by the creation of bottleneck for data that there is no significant loss of information and also efficient in reconstruction w.r.t original data [79, 20]. The bottleneck layer, which is also referred to as latent space is obtained after encoding the data and acts as a starting point for decoding process. It is quintessential as it constitutes the major part of the information in form of reduced representations. The main objective of autoencoders is two-fold, firstly, to develop a model with zero reconstruction loss is not strictly advised as the model is not regularized and secondly, the model should preserve the essential feature information and be efficient in pattern recognition. The main drawback with the autoencoders is that they cannot generate new data and are mainly focused on reproducing the same input as their latent space is not generalized enough to create data which is a case of severe overfitting. Variational autoencoders (VAEs) are autoencoders that are trained in a regularized manner and capable of generating data by minimizing overfitting and ensure preserving significant feature information in the latent space. In contrast to autoencoders which results a data point as output, VAEs output a distribution over a latent space, from which a random point in the distribution is considered for decoding. This technique of sampling from a distribution over a latent space generalizes the output and reduces overfitting.

For models which generate data from latent dimensions, learning complicated dependencies between dimensions is significant to accurately represent our dataset as there can be one or many settings of latent variables to generate meaningful data [20, 23, 79]. In mathematical terms, each data point X after encoding results in a latent space denoted by $\mathcal{Z}$ which represents a probability density function (PDF), $P(z)$ on latent variables z defined over $\mathcal{Z}$. These latent variables z represent a group of deterministic functions $f(z;\theta)$ in space Θ (obtained after decoding) where $f : \mathcal{Z} \times \Theta \rightarrow \mathcal{X}$. In the above equation f results in a single point for a certain z and θ, and z is random and θ is fixed. To achieve a high probability of $P(x)$ while sampling z; θ should be optimized, so the deterministic functions $f(z;\theta)$ should be considered as inputs Xs in our dataset and the equation of probability is given by

$$P(X) = \int P(X \mid z;\theta)P(z)dz \tag{2.3}$$

In the Eqn. 2.3, by utilizing probability rule $P(X \mid z;\theta)$ distribution is substituted for $f(z;\theta)$, to explicitly include the dependency of z and X. If the model is adept in the generation of training data with less reconstruction error, it is more likely to generate similar outputs with different inputs rather than less meaningful ones. This is known as maximum likelihood and generally the distribution $P(X \mid z;\theta)$ is Gaussian which is characterized by mean $f(z;\theta)$ and covariance σ^2. This assumption of Gaussian distribution is useful in using

the gradient descent method for improving probability $P(X)$ by constraining the $f(z;\theta)$ to generate plausible X for some z such that it can be used as a generative model. $P(X)$ in Eqn. 2.3 is quite complicated to calculate directly, so a new function $Q(z \mid X)$ is introduced to calculate a value in X and provide a distribution over z values that can approximate X. It is obvious that the values Q takes are comparatively smaller space than before. This can be solved easily by computing the expectation of $E_{z\sim Q}P(X \mid z)$ which can be obtained from Kullback-Leibler divergence (denoted by $\mathcal{D}$) between $P(z \mid X)$ and $Q(z)$.

$$\mathcal{D}[Q(z)\|P(z \mid X)] = E_{z\sim Q}[\log Q(z) - \log P(z \mid X)] \tag{2.4}$$

By application of Bayes rule on $P(z \mid X)$; $P(X)$ and $P(X \mid z)$ can be included in the Eqn. 2.4. The main objective function of VAEs is given by solving the Eqn. 2.4 and can be further reduced to Eqn. 2.5.

$$\log P(X) - \mathcal{D}[Q(z \mid X)\|P(z \mid X)] = E_{z\sim Q}[\log P(X \mid z)] - \mathcal{D}[Q(z \mid X)\|P(z)] \tag{2.5}$$

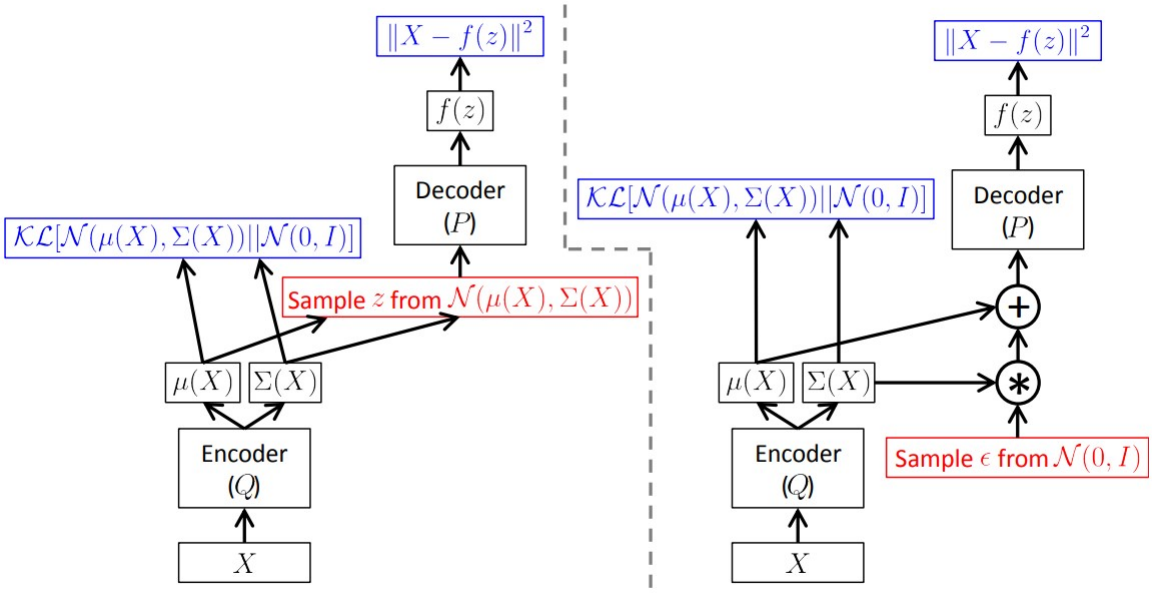


Figure 2.5: Variational autoencoder as a generative model [23].

Fig. 2.5 demonstrates VAEs with two different sampling approaches where the red color refers to non-differentiable layers and blue color corresponds to loss layers. The right one denotes sampling method in which latent variable is sampled directly from mean $\mu(X)$ and covariance $\Sigma(X)$ of $Q(z|X)$ i.e., $\mathcal{N}(\mu(X), \Sigma(X))$ and in the left part latent variable is sampled from the formula $z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$; $\mu(X)$ and $\Sigma(X)$ denote mean and covariance of output distribution of encoder respectively. This reparameterization of z

on right side of the Fig. 2.5 allows differentiation and hence they can be back-propagated. However, this is not the case in the left part of the figure as there are no random variables. The first loss, $\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, I)]$ is termed as Kullback-Leibler Divergence loss while the second loss, $||X - f(z)^2||$ is termed as reconstruction loss.

2.4.2.1 Conditional variational autoencoders

In order to impose a certain condition on the generated output from the VAEs and still satisfy one-to-many mapping, the conditional variational autoencoders (cVAE) are introduced [99]. In conditional variational autoencoder (cVAE), the label of the input is provided as a condition for both encoder and decoder. The math of VAEs is adjusted to accommodate this condition while generating output. The objective function is presented in Eqn. 2.6, where Y is the output label and the modified framework is demonstrated in Fig. 2.6. In cVAE, the output label Y is embedded with both encoder and decoder inputs, and additionally, the loss function is also modified to include the given condition.

$$\begin{aligned} &\log P(Y \mid X) - \mathcal{D}[Q(z \mid Y, X) \| P(z \mid Y, X)] = \\ &\quad E_{z \sim Q(\cdot|Y,X)}[\log P(Y \mid z, X)] - \mathcal{D}[Q(z \mid Y, X) \| P(z \mid X)] \end{aligned} \tag{2.6}$$

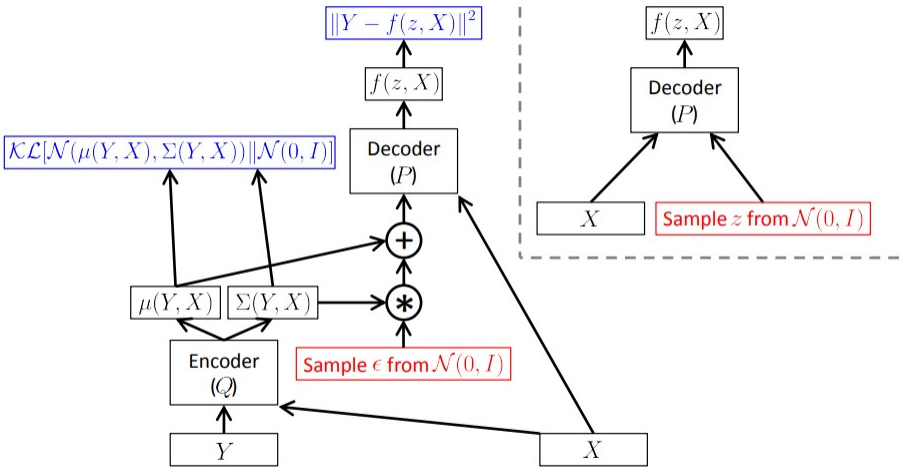


Figure 2.6: Conditional variational autoencoder [23].

2.4.3 Generative adversarial networks

Ian Goodfellow et al. developed GANs in 2014 [80], which became an instant success that they can be described as the biggest development in the field of deep learning in the last decade. The basic idea of GANs is that two neural networks namely generator and discriminator compete with each other in a zero-sum game to generate realistic data. The

generator produces data with noise as input while the latter acts as a classifier whether the generated data is plausible or not. The objective of the generator in GANs is to learn the distribution P_g over the given data points X and then define a mapping to a data space $G(z;\theta_g)$ on predefined input noise distribution $P_z(z)$, where G is a differentiable function represented by a multilayer perceptron (MLP). A discriminator $D(x,\theta)$ returns a scalar which signifies whether x is from P_g or given data. The main intention of the discriminator is to maximize this probability of classifying training data and generator outputs accurately. In simpler words, the generator tries to generate realistic data from noise $p_z(z)$ by mapping it to a data space $G(z;\theta)$ and discriminator $D(x;\theta)$ ensures that x corresponds to the training distribution. The objective function of GANs is quantified by $V(G,D)$ in Eqn. 2.7 and it is nearly non-convex. The optimization of GANs is a complex process as there is always a possibility of multiple local optima and a wide amount of research is focused on that area [21, 22, 100].

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.7}$$

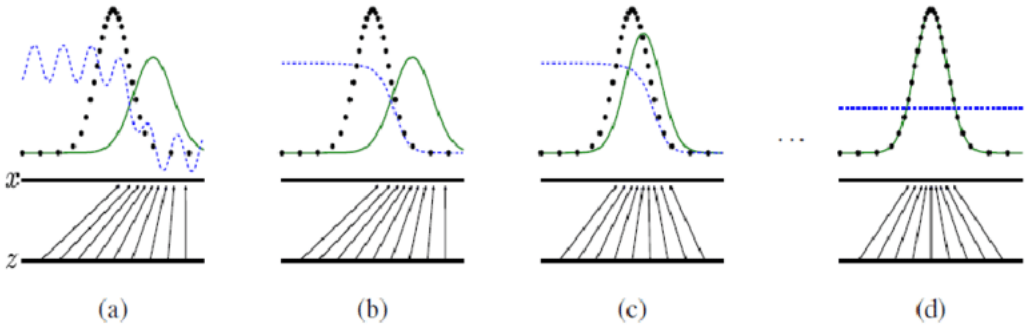


Figure 2.7: Evolution of training process of GANs: (a) The green line represents the generative distribution P_g, the blue dashed line represents discriminator distribution between the generated data and real target distribution, the black dotted normal distribution is the given data distribution and the below lines represent the orientation of latent space z domain. (b) Initially, the discriminator classifies the generated data and given data easily and the generator is updated with the discriminator feedback. (c) As training progresses, the feedback from discriminator updates generator by governing the gradients to proceed to locations in space, converging the objective function and generating plausible data. In the ideal case, after multiple training iterations, the generator learns intricate patterns in the data distribution by incorporating discriminator responses and mapping latent variables non-uniformly to resemble the given data distribution. (d) Finally, generator and discriminator converge implying that the generator is able to produce realistic distributions w.r.t training data by deceiving the discriminator. The global optima of GANs is reached when $p_g = p_{data}$ [80].

.

In the initial training stages, the generator learns the mapping between the latent variables and data space slowly while the discriminator easily differentiates targets from generator outputs. The weights of the generator are not updated directly by optimizing generator loss but through the objective function of the discriminator. As training progresses, the generator optimizes its objective function by producing better plausible outputs and it becomes more challenging for the discriminator to detect generated ones from real targets i.e., in Eqn. 2.7 the second term $\log(D(G(\boldsymbol{z})))$ increases. The training of GANs is a complex process, figure 2.7 depicts training of GANs. However, this is not the case in general, as the training process is not as smooth as it seems and suffers from various limitations.

2.4.3.1 Conditional generative adversarial networks

The GANs framework discussed in Section. 2.4.3 utilizes multi-layer perceptron in both discriminator and generator and there is no restraint on the output generated [101]. This can be achieved by employing a conditional probabilistic generative model by conditioning the input in such a manner that the one-to-many mapping between the input and output is substituted by a conditional prediction. This can be achieved by providing both discriminator and generator with supplementary information of output label y, this y can be any kind of information that is sufficient for obtaining a specified output. As a result, the generator is provided noise $p_z(z)$ combined with y as an additional input layer this permits the GANs for substantial flexibility in representing the hidden distribution. In a similar manner discriminator is also provided with inputs x and y and the objective function given in 2.7 of GANs is adjusted as 2.8 and architecture of conditional generative adversarial networks (cGANs) is illustrated in 2.8. In recent times, numerous cGANs have been developed each corresponding to its own applications.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x} \mid \boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z} \mid \boldsymbol{y})))] \tag{2.8}$$

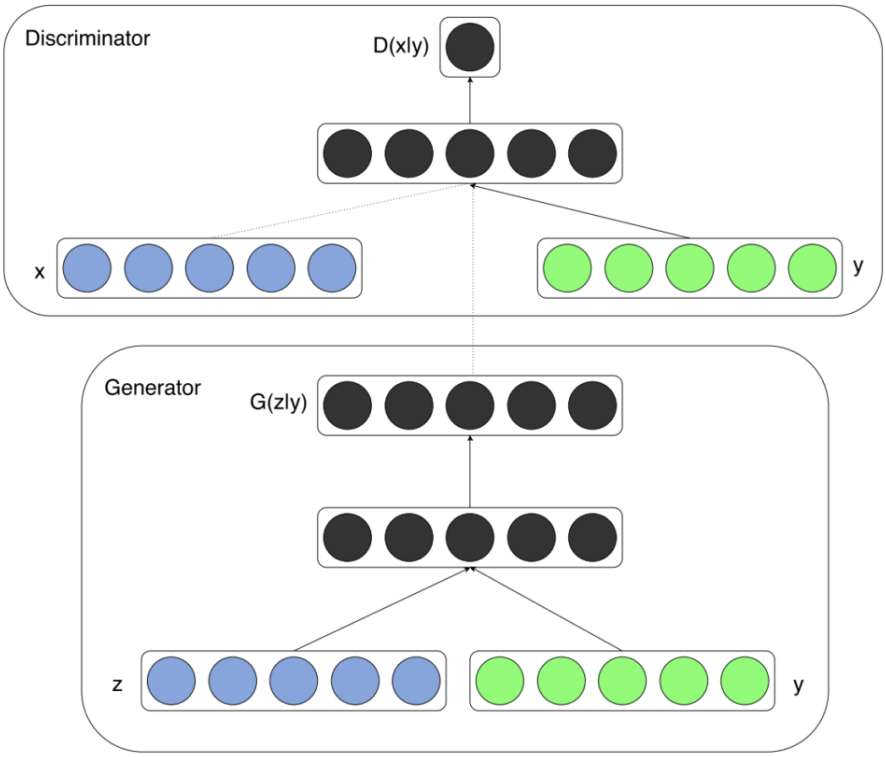


Figure 2.8: Conditional GANs [101].

2.4.4 pix2pixGAN

Isola et al. [102] developed pix2pixGAN in 2017 for image-to-image translation tasks by using cGANs. There have been other GANs developed for image-to-image translation without conditional setting and also generated remarkable results in image inpainting [103], future state prediction tasks [104], style transfer [105] and image superresolution [106]. Apart from these applications, pix2pixGAN can be used for a wider range of applications which include coloring black and white images, generate photos from mapped labels, conversion of google maps to aerial images, reconstituting images from edge maps, etc. [102]. It carries the basic idea of GANs i.e., discriminator attempting to quantify a loss that classifies whether a given image is real or fake, meanwhile synchronously training generator to reduce that loss. This cGANs assimilates mapping from input to output image by training a loss function, it is termed as pix2pix since it predicts pixels from pixels. pix2pixGAN constitutes Unet (described in Section. 2.4.1) as the generator and a PatchGAN as a classifier in discriminator. The idea of PatchGAN discriminator is motivated from Markovian GANs [105] where it is proposed first. It utilizes patches of shape $N \times N$ in the image for classification to determine whether it is real or fake, i.e., the discriminator traverses convolutionally across the patches of the image and returns aggregate of their results. This exploits the correlations between

the patches contextually and allows faster learning [102].

2.4.5 Wasserstein GANs

Wasserstein generative adversarial networks (WGAN) were developed by Arjovsky et al. [100] to overcome certain limitations pertaining to model stability, mode dropping phenomenon and vanishing gradients observed during training of GANs. In WGAN, Wasserstein distance is employed to improve stability in training. The main objective of GANs is to minimize the distance between a model distribution and target distribution, and training of GANs is dependant on the type of distance metric chosen ad hoc. The selected definition of the distance metric significantly influences the convergence of model distribution towards real distribution and for this reason that it should be continuous. For instance; VAEs minimize Kullback-Leibler divergence (KLD) and GANs use Jensen-Shannon divergence (JSD) as objective function [80]. Wasserstein distance or also termed as Kantorovich–Rubinstein metric is used as the distance metric in WGAN, this improved stability of training GANs and also led to faster convergence.

The mathematical equations of different distance metrics between two probability density distributions $P_r, P_g \in Prob(X)$ where X is the set compassing image space $[0,1]^d$ are illustrated below.

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \tag{2.9}$$

  Here Σ denotes the set of all Borel subsets of X [100]

- The *Kullback-Leibler* (KL) Divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \tag{2.10}$$

  where P_r and P_g are distributions of real and generated respectively. This divergence becomes asymmetric and discontinuous when $P_g = 0$

- The *Jensen-Shannon* (JS) Divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| P_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \tag{2.11}$$

  where $P_m = ((P_g + P_r)/2)$ and JS divergence is symmetric.

- *Earth-Mover* (EM) Distance or Wasserstein-I

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{2.12}$$

  where $\Pi(P_r, P_g)$ are a group of joint distributions and distance between those two are measured by $Y(x, y)$ where Y specifies the amount of mass required to be carried from

x to y such that the distribution P_g is equivalent to P_r. EM distance is the cost of this movement. It is better understood by visualizing the real and generated distributions as heaps of sand. The objective is achieved by transforming the generated distribution to target distribution. The Wasserstein distance is measured by the minimum cost of this transformation by calculating the amount of sand that is required to be transported and due to this analogy, it is also known as earth mover's distance.

In WGAN, the value function is defined by Kantorovich-Rubinstein duality [107]

$$W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \min_G \max_{D \in D} \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[D(x)] \tag{2.13}$$

In the above Eqn. 2.13, D corresponds to 1-Lipschitz functions and P_g denotes the generated model distribution. In WGAN discriminator is renamed as critic as its function is not to classify but minimizing the value function w.r.t the generator i.e., minimizing $W(P_r, P_g)$. In the original WGAN paper, Lipschitz constraint is enforced by weight clipping in critic by clamping weights in a fixed interval (for instance say weights $W \in [-0.01, 0.01]$) after each gradient update. But using weight clipping as a constraint resulted in optimization issues in training [22]. An alternative to weight clipping is presented by Gulrajani et al. [22] in 2017 as an improvisation to the original WGAN paper by penalizing the gradients. It constrains the norm of critic's output gradient w.r.t its input. The equation is depicted in Eqn. 2.14 and λ is a gradient penalty coefficient which can be considered as an additional hyperparameter determined by heuristic process [22].

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})]]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}\left[\left(\left\|\nabla_{\hat{\mathbf{x}}} D(\hat{\boldsymbol{x}})\right\|_2 - 1\right)^2\right]}_{\text{gradient penalty}} \tag{2.14}$$

Chapter 3

Data generation

This chapter explores the methods for data generation using DREAM.3D software and as discussed in the Chapter. 2, microstructure-descriptors such as volume fraction, grain size distribution (mean equivalent sphere diameter), and orientation distribution function are varied in their respective space to synthesize 3D and corresponding 2D microstructures. The main reason for selecting synthetic microstructures is their easy accessibility and feasibility of constructing them with specified statistics.

3.1 Sampling strategy

DREAM.3D is a software that is based on the filter-pipeline principle and focuses mainly on polycrystalline microstructures. Each filter performs a particular task and a combination of one or more filters constitutes a pipeline for addressing multiple issues together. *Stats Generator* is one of the main filters for artificial microstructure generation, by employing a series of descriptors. Based on the selected descriptors, DREAM.3D synthesizes an equivalent 3D by solving a multi-objective optimization problem, since a number of descriptors can be considered for synthetic building, arguably the software can be treated as the most accurate one for the analysis of polycrystalline microstructures. However, when a number of larger-sized synthetic 3D microstructures is expected to cover a wide range of descriptor-space, the pipeline cannot always provide globular-like grains. This issue complicates the data generation strategy, so the whole process has been divided into smaller parts by employing python scripts wherever necessary.

In this work, we are concerned with volume fraction, grain size distribution and orientation distribution function. Analyzing the statistics of a few duplex TiAl alloys, we fixed our data-space, where the range of pure γ-grains was 60-70%. On the other hand, the range of mean ESD was selected in between 6-11 μm. Regarding the orientation distribution, we observed that the first and third components of Euler angles, i.e., EA0 and EA2 are uniformly distributed, that's the main reason for not varying the space of these two components. The middle component, i.e., EA1, showed a Gaussian like distribution where we tried to generate four more sets of EA1 by varying the distribution patterns

slightly, after observing a real duplex alloy. The selected plausible values for the microstructure descriptors are illustrated in Table. 3.1. The variation of EA1 is also shown in Table. 3.1, corresponding distribution patterns are shown in Fig. 3.1.

Regarding the whole workflow of data generation, we had to apply few tricks, since 3D synthetic generation from a 2D image directly did not give high-fidelity outcomes. Initially, a single-phase 3D image has been generated then it has been converted to two phases by stochastically switching phases to achieve the expected volume fraction. Apart from grain size distribution and ODF parameters, several statistical measures are available for describing microstructure. In order to get a greater extent of information from EA1, it is exploited by altering its mean and the maximum frequency of the EA1. The ODF space of EA1 is explored and modified w.r.t its mean and frequency of EA1 values shown in Fig. 3.1. The initial ODF file has statistics of mean of EA1 0.7 and a maximum frequency of 120 between EA1 values 0.6 and 0.7. The ODF is altered in such a manner to accomplish an average EA1 component of 0.6 and 0.8 (irrespective of maximum frequency) and the maximum frequency of original ODF to 100 and 130 (with mean of EA1 as 0.7).

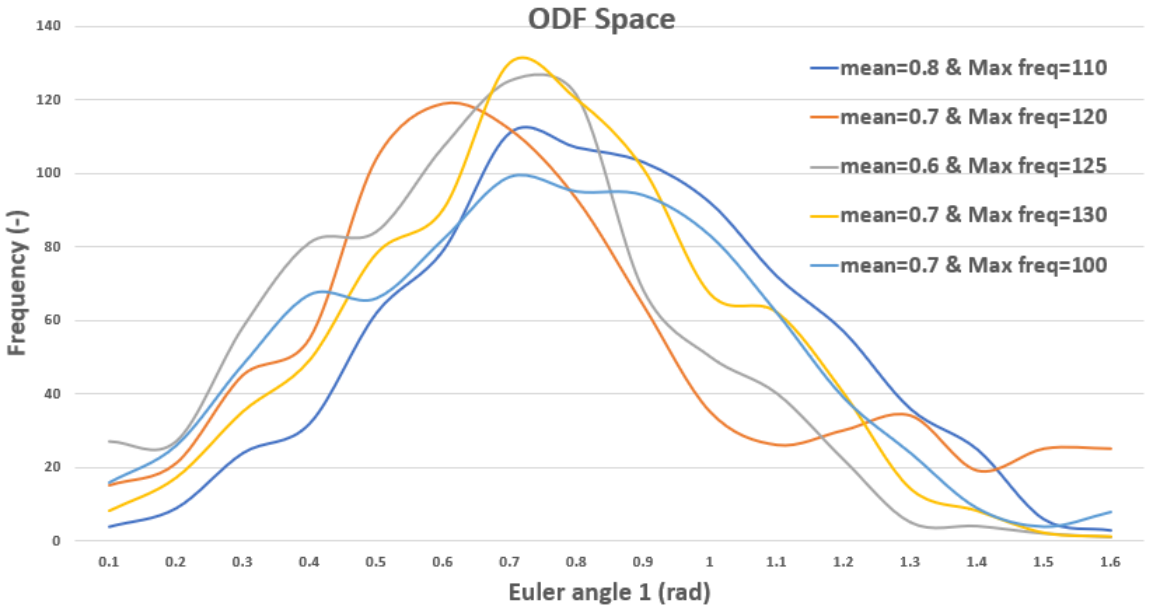


Figure 3.1: EA1 component of ODF distribution. X-axis represents the EA1 values (in rad) and Y-axis represents the frequency of EA1 values in those respective intervals*.

Table 3.1: Value range of microstructure-descriptors.

Microstructure-descriptor	Range of values
Volume fraction	0.6, 0.62, 0.64, 0.66, 0.68, 0.7
Mean ESD	6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 10.5, 11
EA1	mean = 0.6 & Max freq = 125 mean = 0.8 & Max freq = 110 mean = 0.7 & Max freq = 120 mean = 0.7 & Max freq = 130 mean = 0.7 & Max freq = 100

3.2 Synthetic microstructures using DREAM.3D

DREAM.3D [17] is an open-source software package with several utilities for processing, fragmenting, quantifying, representing, and manipulating microstructures. It is developed in such a manner that enables the user to develop pipelines constituting independent filters (each performing its own function). Its interface is user-friendly and accessing their tutorials provides a better understanding allowing users to build pipelines according to their requirements. Synthetic microstructure generation is one of the significant primary features of DREAM.3D which offers flexibility in constructing microstructures with given parameters. This work exploits the synthetic microstructure generation feature of DREAM.3D for data generation.

For synthetic microstructure generation, firstly the *Stats Generator* filter from DREAM.3D is employed to designate a set of statistics that can be utilized for constructing required synthetic microstructures. This filter allows the user to select different statistics associated with grain size and shape distributions, nearest neighbor distances and so on. It employs feature packing algorithms to instantiate features and then optimize the shapes, sizes and morphology-based on the given statistics [108]. As discussed in the previous chapter; volume fraction, grain size distribution, and grain orientation distribution functions are chosen as microstructure-descriptors. The grain size distribution tab in *Stats Generator* filter is modified accordingly by adjusting the mean and standard deviation parameters of the diameter of the sphere (which has the equivalent volume of the grain). In this work, the geometry of 3D grains is assumed to be ellipsoid and the custom ODF file is given as input. Fig. 3.2 and 3.3 illustrate the inputs in grain size distribution and ODF tabs respectively in *Stats Generator* filter for generating a synthetic microstructure with mean ESD equal to 11 (obtained by modifying the *mu* and *sigma* fields). After the generation of a synthetic single-phase 3D microstructure, the volume fraction is introduced by altering the phases of arbitrary grains by a python script.

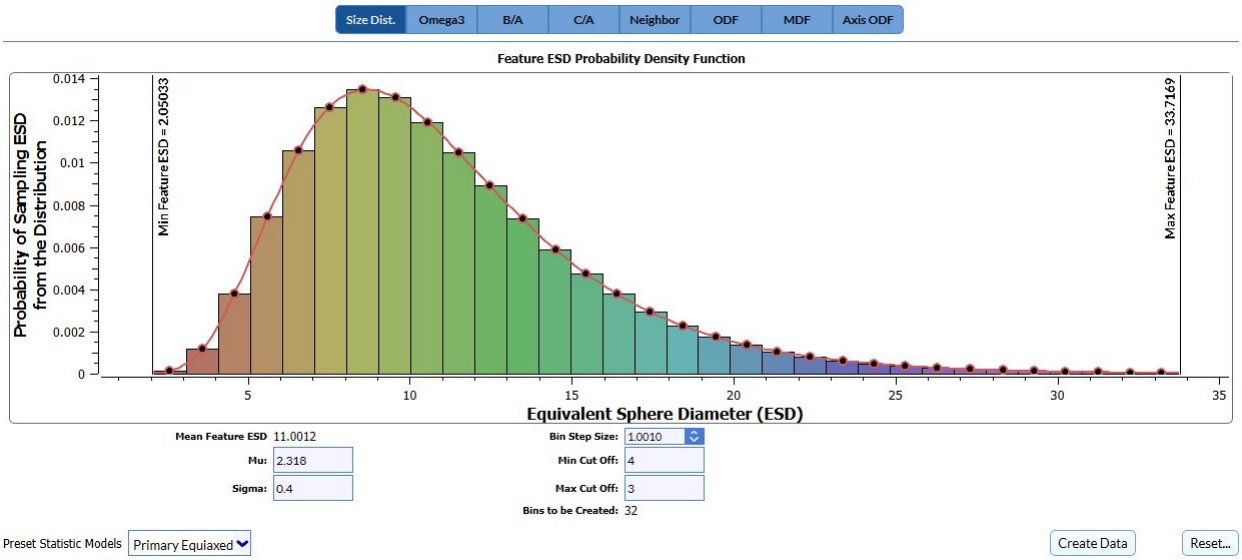


Figure 3.2: Grain size distribution tab in *Stats Generator* filter in DREAM.3D.

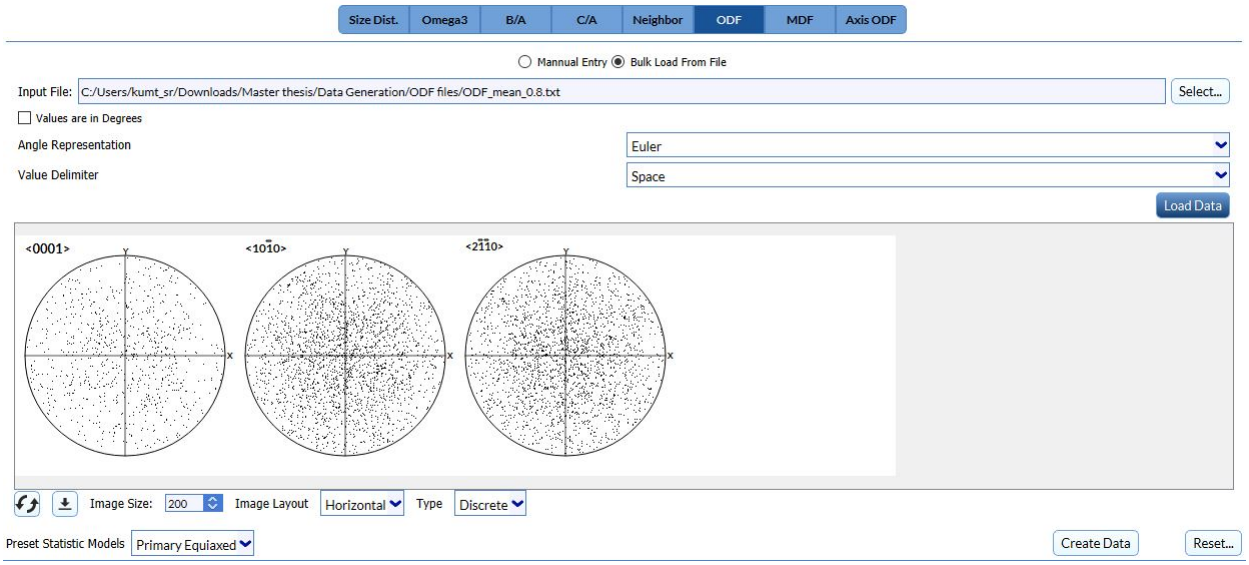


Figure 3.3: ODF distribution in *Stats Generator* filter in DREAM.3D.

3.2.1 Synthetic two-phase microstructure generation

The complete process is summarized in the steps below.

1. Initially a single-phase synthetic 3D microstructure of size 192x192x192 has been generated in DREAM.3D by using filters mentioned in the Table. 1 in Appendix A.

2. Secondly another pipeline is employed for the removal of small features by merging them with existing grains and the filters employed are shown in Table. 2 in Appendix A.

3. Phases are altered in the resultant microstructures to achieve the required volume fraction and this alteration is performed using python.

4. After the conversion of single phase to dual-phase microstructure, values of microstructure-descriptors are calculated using a third pipeline which comprises filters described in Table. 3 in Appendix A.

5. Finally a fourth pipeline is built for slicing the synthetic 3D dual phase microstructure to get a 2D synthetic 2 phase microstructure which also calculates the respective values of microstructure-descriptors for that 2D slice. The filters associated with this pipeline are described in Table. 4 in Appendix A.

6. The slicing procedure is performed on an iterative basis where the 3D volume is sliced sequentially along 3 directions. The obtained microstructure-descriptor values of 2D slice are then compared with that of corresponding 3D values and the 2D slice with the least Euclidean difference is selected.

All the above processes have been automated in python script using pipeline runner feature from DREAM.3D. The above mentioned entire process is illustrated in a flowchart demonstrated in Fig. 3.4.

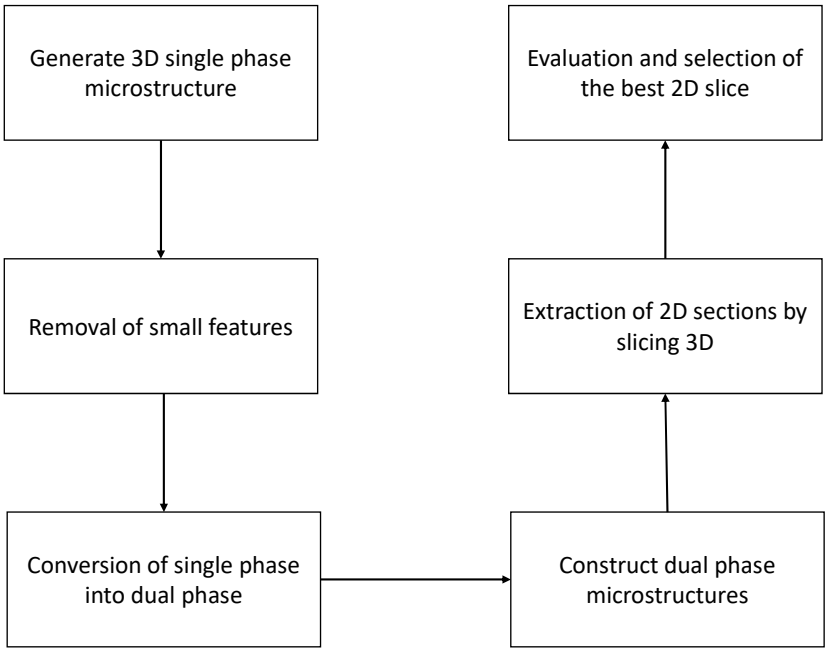


Figure 3.4: Flowchart demonstrating the data generation process*.

3.2.2 2D vs 3D comparison

After generating statistically equivalent 2D slices from the corresponding 3D microstructures, the microstructure-descriptors are computed and visualized. Euclidean distance between 2D slices and their respective 3D volumes has been calculated by summing the differences between microstructure-descriptors in 2D and 3D, i.e., $||d_{2d} - d_{3d}||_2$, where $\mathbf{d} = [\varnothing, \rho, \varphi]^T$ where $\varnothing$, ρ, φ denote volume fraction, mean ESD and mean EA1 respectively. Fig. 3.5, 3.6 and 3.7 represent the values of microstructure-descriptors calculated from 3D and 2D synthetic microstructures. Fig. 3.8 shows the Euclidean distance between 3D volumes and their 2D counterparts. It can be observed that Fig. 3.7 there is no ideal 2D slice which completely matches the microstructure-descriptors of synthetic 3D microstructures.

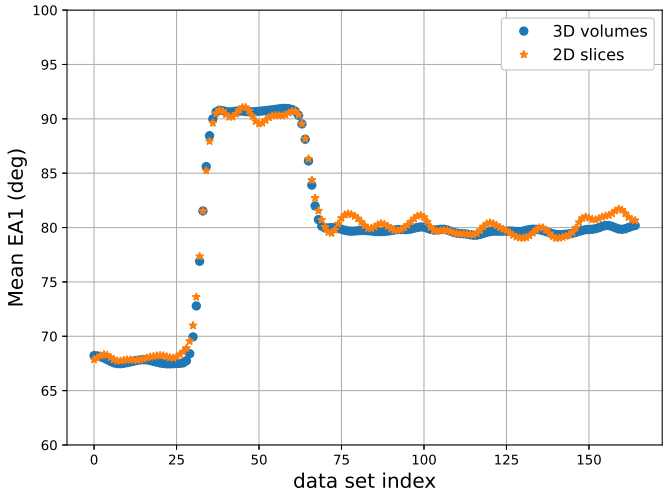


Figure 3.5: Comparison of mean EA1 between 2D and 3D synthetic microstructures*.

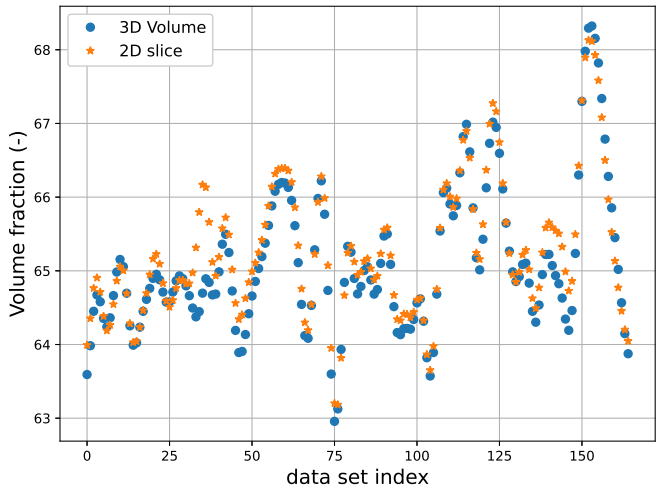


Figure 3.6: Volume fraction comparison between 2D and 3D synthetic microstructures*.

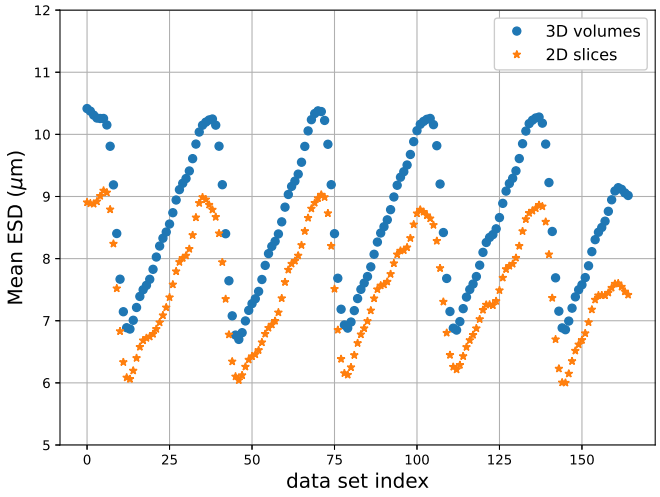


Figure 3.7: Comparison of mean equivalent sphere diameter between 2D and 3D synthetic microstructures*.

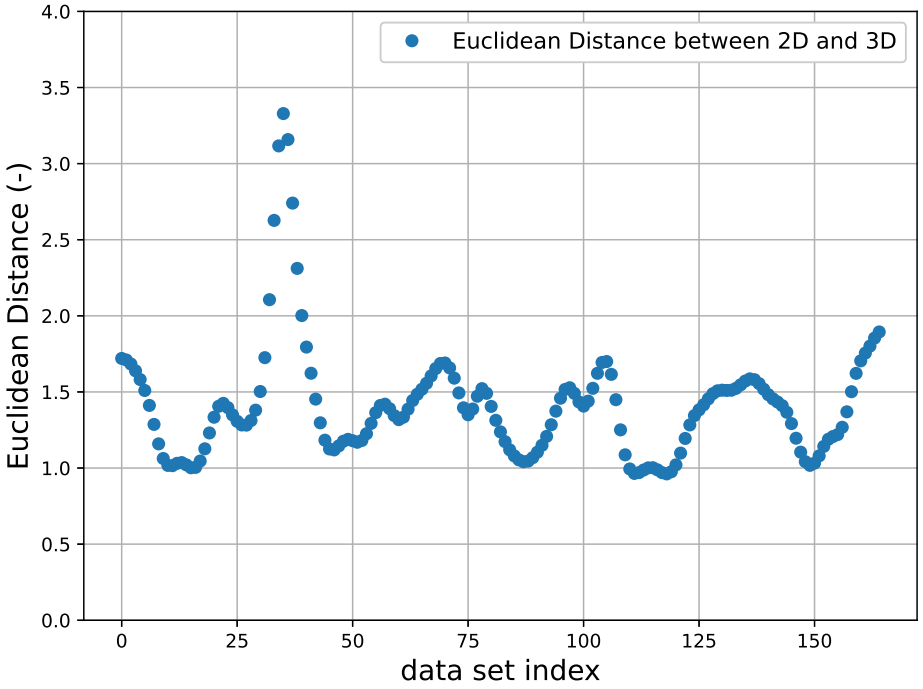


Figure 3.8: Euclidean distance between 2D slices and 3D volumes*.

3.3 Lamellar generation algorithm

Lamellae structure in a microstructure consists of ultra-fine thread-like structures which are present in alternative layers with another material. Most common examples of lamellar microstructures are duplex stainless steels and titanium aluminide alloys [109, 24]. Lamellar microstructures in TiAl are formed by heating the alloys (with 43-48% Al) into α region (>1250°C) and then cooling through $\alpha_2+\gamma$ phase region resulting in formation of alternating layers of γ and α_2 lamellae [110]. These alloys show high fracture toughness as deformations mobility which can be explained by inhibition of deformation mobility by the lamellae. Since TiAl alloy system is the material of interest in this work, they can be comprehended in a superior manner by employing synthetic lamellar microstructures. Inclusion of complexities such as lamellar in synthetic microstructures is not yet feasible in DREAM.3D software. So, an algorithm is formulated to inject lamellar with uniform thickness but with varied orientation into synthetic microstructures. Although, the algorithm is initially tested in python on 2D synthetic microstructures generated from DREAM.3D and later extended to 3D also.

3.3.1 Description of algorithm

Since the main objective is the inclusion of lamellae in a synthetic microstructure, it is important to understand the hierarchy of their data structure. Generally, pixel and voxel-based information are used for visualization of 2D and 3D microstructures respectively, and subsequently, grain-based information can be extracted from them. After the segregation of individual grains, every single grain is considered a separate entity. This grain level information is beneficial for modifying the contents of the microstructure and in this case, it is utilized for the insertion of lamellae into the grains. The introduction of lamellae in a 2D synthetic microstructure is briefly explained here.

Let us consider a particular grain G_i in a 2D synthetic microstructure in which lamellae of thickness t need to be inserted at an orientation *theta* w.r.t X axis and separated by a distance d. Firstly, these required parameters are properly defined. Secondly, a reference pixel is defined with a minimum y coordinate nearest to the X axis and the lamellae insertion process is initiated. Based on the orientation *theta*, the intercept w.r.t its predefined axis is calculated by considering the orientation as slope. A function is developed to determine whether a particular pixel in the given grain should constitute lamellae or not by employing the information related to reference pixel and lamellae parameters). Finally, every pixel in the grain G_i is mapped to this function. Based on the output of this function, the pixel data is modified for transforming a regular grain to a grain consisting of lamellae. This process is repeated for all the grains with lamellae. A simple description of the algorithm for introducing lamellar into a 2D synthetic microstructure is illustrated in Fig. 3.9. Pandas library in python is used for implementing this algorithm. In a similar manner, lamellae are also introduced into synthetic 3D microstructures by considering a reference voxel. The microstructures with lamellae in 2D and 3D are illustrated in Fig 3.10 and 3.11 respectively.

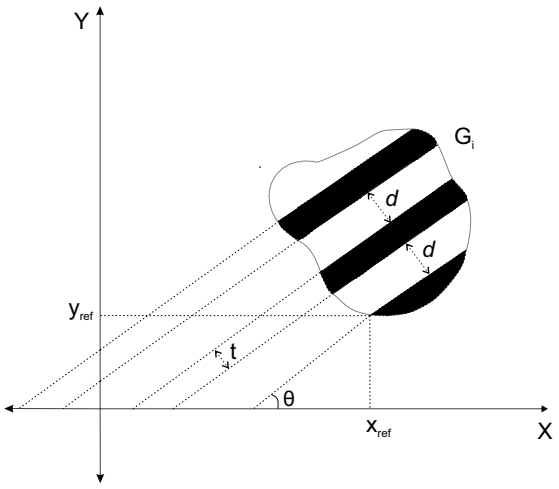


Figure 3.9: Lamellar of thickness t in a random grain G_i on a XY plane with an angle of θ with distance d between each lamellae.

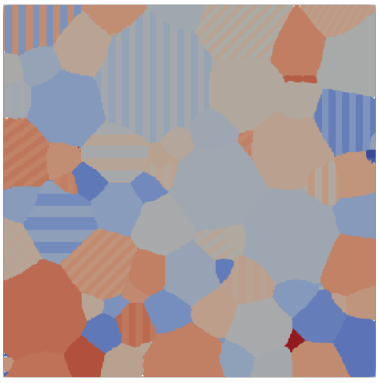


Figure 3.10: 2D synthetic lamellar microstructure visualized in paraview.

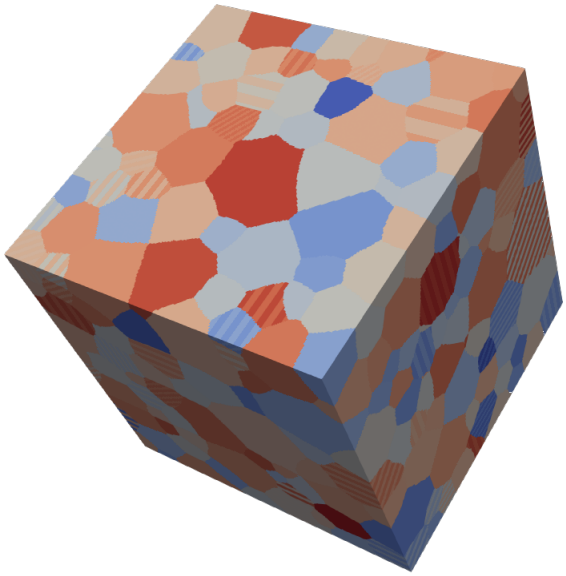


Figure 3.11: 3D Lamellar microstructure visualized in paraview.

3.3.2 Future scope

The present algorithm has immense potential to achieve high fidelity complex microstructures artificially. The developed algorithm needs optimization and some ideas for further development are discussed in this section. Currently, it can generate lamellae with the given parameters uniformly across a grain by treating it as an individual entity. The grain information can be exploited to produce lamellae with non-uniform thickness in a grain. Besides, the distance between lamellae need not be necessarily consistent across the single grain. Furthermore, the normal and orientation of lamellae are presently defined only w.r.t a single axis i.e., either X, Y or Z axes and this can be extended to planes. This algorithm can be integrated as an additional utility for generating complex synthetic microstructures in DREAM.3D.

Chapter 4

Model training

4.1 Configuring generative models and training

In this section, architectures of generative models which are employed in this work are presented. Microstructure reconstruction of statistically equivalent 3D from 2D microstructures falls under semi-supervised learning problem as it consists of learning the statistical distributions of microstructure-descriptors (unsupervised part) and qualitative comparison of labels i.e., visual representations of real and generated microstructures (supervised part). Generative modeling using convolutional neural networks (CNNs) architectures such as cVAE and DCGANs (pix2pixGAN) are applied and the results are presented and discussed in subsequent sections. The generative models are trained on the CARA cluster from DLR (Deutsches Zentrum für Luft- und Raumfahrt) endowed with Quadro P5000 GPU. Initially, the models are trained on a single channel (EA0) to determine optimal hyperparameters and then the model training is extended to all the channels. Models are saved at intermediate steps during training to supervise training process and prevent the model from overfitting. The input channels (phases and three components of Euler angles) are standardized before training. Taking into consideration, the given computational resources and thesis time-frame, for efficient training and result analysis, the dimensions of the synthetic microstructures have been reduced from $192 \times 192 \times 192$ to $64 \times 64 \times 64$.

4.1.1 Unet

Unet framework consists of a combination of encoding and decoding layers. Firstly, the given 2D microstructure is extrapolated in Z-dimension to reconstruct the required dimensions in 3D to convert it as an input to the model architecture and standard error metrics such as MAE are employed. Each encoding layer consists of Conv3D layers followed by Batch Normalization and then with leaky ReLU activation function. Decoding layers are built in a similar manner but instead of Conv3D layers, Conv3DTranspose layers are employed to increase the dimensions. Since the Unet framework employs skip

connections between encoding and decoding layers, concatenation layers are employed in decoding layers to substantiate the decoding output with the encoding output at that corresponding layer.

Unet architecture consists of five encoding layers followed by a bottleneck layer and five decoding layers which is described in Table. 4.1. The objective function is constructed to accommodate predicted image, volume fraction and distributions of Euler angles. The model is trained using Adam optimizer with a learning rate of 0.00001.

4.1.1.1 Customized loss function

In the initial stages, standard error metrics such as *Mean Absolute Error*, *binary crossentropy*, *Kullback-Leibler Divergence* were employed in loss function in the Unet and pix2pixGAN models. To improve the accuracy of predictions, a custom loss function is developed to quantitatively compare the real and generated microstructures by including volume fraction and distributions of Euler angle components. Volume fraction V_f can be computed by considering the phases channel and for comparison of Euler angles distributions, kernel density estimation (KDE) is utilized. The main reason for utilizing KDE is converting the Euler angles distributions into continuous differentiable histograms using a Gaussian filter.

$$MAE = \frac{\Sigma_{i=1}^{n} |y_{i\,(\text{true})} - y_{i\,(\text{predicted})})|}{n} \tag{4.1}$$

So, the custom loss function in addition to Eqn. 4.1, consists of additional terms such as Eqn. 4.2 and 4.3 for calculating difference in volume fractions and Euler angle distributions using KDE respectively for the true and predicted microstructures.

$$\Delta V_f = \frac{\Sigma_{i=1}^{n} |(V_{f(i,\text{true})} - V_{f(i,\text{predicted})})|}{n} \tag{4.2}$$

$$\Delta KDE = \frac{\Sigma_{i=1}^{n} \Sigma_{\phi,\theta,\psi} |(KDE_{(i,\text{true})} - KDE_{(i,\text{predicted})})|}{n} \tag{4.3}$$

where ϕ, θ, ψ in Eqn. 4.3 refer to the three components of Euler angles respectively.

Table 4.1: Unet architecture with skip connections.

Layer	**Operation**	**Output shape**	**kernel size**	**Activation**
Unet input = 2D input image (64, 64, 4)				
encoding block 1	Conv3D	(64, 64, 64, 64)	(3,3,3)	ReLU
	MaxPooling3D	(32, 32, 32, 64)		
encoding block 2	Conv3D	(32, 32, 32, 128)	(3,3,3)	ReLU
	MaxPooling3D	(16, 16, 16, 128)		
encoding block 3	Conv3D	(16, 16, 16, 256)	(3,3,3)	ReLU
	MaxPooling3D	(8, 8, 8, 256)		
encoding block 4	Conv3D	(8, 8, 8, 512)	(3,3,3)	ReLU
	MaxPooling3D	(4, 4, 4, 512)		
encoding block 5	Conv3D	(4, 4, 4, 512)	(3,3,3)	ReLU
	MaxPooling3D	(2, 2, 2, 512)		
bottleneck layer	Conv3D	(2, 2, 2, 1024)	(3,3,3)	ReLU
decoding block 1	Conv3DTranspose	(4, 4, 4, 512)	(3,3,3)	ReLU
	Concatenate	(4, 4, 4, 1024)		
	Conv3D	(4, 4, 4, 512)	(3,3,3)	ReLU
decoding block 2	Conv3DTranspose	(8, 8, 8, 512)	(3,3,3)	
	Concatenate	(8, 8, 8, 1024)		
	Conv3D	(8, 8, 8, 512)	(3,3,3)	ReLU
decoding block 3	Conv3DTranspose	(16, 16, 16, 256)	(3,3,3)	
	Concatenate	(16, 16, 16, 512)		
	Conv3D	(16, 16, 16, 256)	(3,3,3)	ReLU
decoding block 4	Conv3DTranspose	(32, 32, 32, 128)	(3,3,3)	
	Concatenate	(32, 32, 32, 256)		
	Conv3D	(32, 32, 32, 128)	(3,3,3)	ReLU
decoding block 5	Conv3DTranspose	(64, 64, 64, 64)	(3,3,3)	
	Concatenate	(64, 64, 64, 128)		
	Conv3D	(64, 64, 64, 64)	(3,3,3)	ReLU
Output layer	Conv3D	(64, 64, 64, 4)		tanh

4.1.2 Conditional variational autoencoders

The cVAE model is constructed with an encoder and decoders as described in Section. 2.4.2.1 and trained on phases and EA1 channels. The model framework is demonstrated in Table. 4.2 and the 2D microstructure is provided as an input to both encoding and decoding parts of cVAE. The encoding part gives two outputs namely μ and Σ, from which latent variable are constructed using the equation $z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$. The decoder

employs a series of Conv3D and Conv3DTranspose layers to reconstruct the given two inputs (latent variables and 2D image) to 3D volume. The decoder is defined on the decoding layers of the cVAE and can be utilized as a generative model. The cVAE model is trained for 3000 epochs at a learning rate of 0.00001.

Table 4.2: Conditional variational autoencoder.

Layer	**Operation**	**Output shape**	**kernel size**	**strides**	**Activation**	**Batch Normalization**
encoder input = 2D input image (64, 64, 2) & 3D image (64, 64, 64, 2)						
E_1	Conv3D	(32, 32, 32, 32)	(4,4,4)	(2,2,3)	ReLU	Yes
E_2	Conv3D	(16, 16, 16, 64)	(4,4,4)	(2,2,2)	ReLU	Yes
E_3	Conv3D	(8, 8,8, 128)	(4,4,4)	(2,2,2)	ReLU	Yes
E_4	Conv3D	(4, 4, 3, 256)	(4,4,4)	(2,2,2)	ReLU	Yes
E_5	Conv3D	(2, 2, 2, 512)	(4,4,4)	(2,2,2)	ReLU	Yes
E_6	Dense	(2, 2, 2, 1024)			ReLU	No
E_7	Flatten	(8192)				
μ	Dense	(50)			linear	No
Σ	Dense	(50)			linear	No
decoder input = 2D input image (64, 64, 2) & $z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$						
D_1	Conv3D	(32, 32, 2, 64)	(4,4,1)	(2,2,1)	ReLU	No
D_2	Conv3D	(16, 16, 2, 128)	(4,4,1)	(2,2,1)	ReLU	No
D_3	Conv3D	(8, 8,2, 256)	(4,4,1)	(2,2,1)	ReLU	No
D_4	Conv3D	(4, 4, 2, 512)	(4,4,1)	(2,2,1)	ReLU	No
D_5	Conv3D	(2, 2, 2, 512)	(4,4,1)	(2,2,1)	ReLU	No
D_6	Conv3DTranspose	(4, 4, 4, 1024)	(4,4,4)	(2,2,2)	ReLU	No
D_7	Conv3DTranspose	(8, 8, 8, 512)	(4,4,4)	(2,2,2)	ReLU	No
D_8	Conv3DTranspose	(16, 16, 16, 256)	(4,4,4)	(2,2,2)	ReLU	No
D_9	Conv3DTranspose	(32, 32, 32, 128)	(4,4,4)	(2,2,2)	ReLU	No
D_{out}	Conv3DTranspose	(64, 64, 64,2)	(4,4,4)	(2,2,2)	tanh	No

4.1.3 Conditional generative adversarial networks: pix2pixGAN

As discussed in Section. 2.4.4, pix2pixGAN constitutes generator with Unet framework and discriminator having a patchGAN architecture. The generator architecture is identical to that of Unet architecture shown in Table. 4.1. Generally, in vanilla GANs, the input is random noise, however, in order to condition the output as necessary, the noise is concatenated with an input label. So, in this case, Gaussian noise is concatenated with 2D

microstructure, as this Gaussian noise is significant in generating non-deterministic outputs and the 2D microstructure acts as a label for obtaining a specified generator output. The noise is concatenated with 2D input in such a way that it attains original 3D dimensions. The discriminator framework from the original pix2pixGAN paper has a patchGAN architecture (described in Section. 2.4.4), however, in this work, it is modified to act as a classifier type model with a single output determining whether its input is real or generated one. The discriminator is also provided with 2D and 3D images as input. In order to improve the accuracy of discriminator classification, volume fraction and Euler angle distributions are supplied as inputs. The discriminator and generator are trained iteratively until the convergence of the objective function is reached. The architectures of discriminator and generator are described in Table. 4.3 and 4.4 respectively.

Table 4.3: Discriminator architecture in pix2pixGAN.

Layer	**Operation**	**Output shape**	**kernel size**	**strides**	**Activation**	**Batch Normalization**
discriminator input = 3D (64, 64, 64, 4) and 2D (64, 64, 4) & $x \in \mathbb{R}^{64*64*64*4}$						
D_1	Conv3D	(32, 32, 22, 64)	(4,4,4)	(2,2,3)	leaky ReLU	No
D_2	Conv3D	(16, 16, 11, 128)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_3	Conv3D	(8, 8, 6, 256)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_4	Conv3D	(4, 4, 3, 512)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_5	Conv3D	(2, 2, 2, 512)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_6	Conv3D	(1, 1, 1, 1)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_7	Flatten	(1)				
D_8	Concatenate	(3)				
D_9	Dense	(10)			leaky ReLU	No
D_{out}	Dense	(1)			sigmoid	No

Table 4.4: Generator architecture in pix2pixGAN.

Layer	**Operation**	**Output shape**	**kernel size**	**Activation**
generator input = 2D input image (64, 64, 4) & $z \in \mathbb{R}^{100} \sim \mathcal{N}(0, I)$				
encoding block 1	Conv3D	(64, 64, 64, 64)	(3,3,3)	ReLU
	MaxPooling3D	(32, 32, 32, 64)		
encoding block 2	Conv3D	(32, 32, 32, 128)	(3,3,3)	ReLU
	MaxPooling3D	(16, 16, 16, 128)		
encoding block 3	Conv3D	(16, 16, 16, 256)	(3,3,3)	ReLU
	MaxPooling3D	(8, 8, 8, 256)		
encoding block 4	Conv3D	(8, 8, 8, 512)	(3,3,3)	ReLU
	MaxPooling3D	(4, 4, 4, 512)		
encoding block 5	Conv3D	(4, 4, 4, 512)	(3,3,3)	ReLU
	MaxPooling3D	(2, 2, 2, 512)		
bottleneck layer	Conv3D	(2, 2, 2, 1024)	(3,3,3)	ReLU
decoding block 1	Conv3DTranspose	(4, 4, 4, 512)	(3,3,3)	ReLU
	Concatenate	(4, 4, 4, 1024)		
	Conv3D	(4, 4, 4, 512)	(3,3,3)	ReLU
decoding block 2	Conv3DTranspose	(8, 8, 8, 512)	(3,3,3)	
	Concatenate	(8, 8, 8, 1024)		
	Conv3D	(8, 8, 8, 512)	(3,3,3)	ReLU
decoding block 3	Conv3DTranspose	(16, 16, 16, 256)	(3,3,3)	
	Concatenate	(16, 16, 16, 512)		
	Conv3D	(16, 16, 16, 256)	(3,3,3)	ReLU
decoding block 4	Conv3DTranspose	(32, 32, 32, 128)	(3,3,3)	
	Concatenate	(32, 32, 32, 256)		
	Conv3D	(32, 32, 32, 128)	(3,3,3)	ReLU
decoding block 5	Conv3DTranspose	(64, 64, 64, 64)	(3,3,3)	
	Concatenate	(64, 64, 64, 128)		
	Conv3D	(64, 64, 64, 64)	(3,3,3)	ReLU
Output layer	Conv3D	(64, 64, 64, 4)		tanh

Table 4.5: pix2pixGAN hyperparameters.

Symbol	**Hyperparameter**	**Value**
α_{gen}	learning rate for Generator	0.00001
α_{disc}	learning rate for Discriminator	0.00001
β_1	first decay rate of Adam optimizer	0.5
β_2	second decay rate of Adam optimizer	0.9

4.1.4 WCGAN with gradient penalty

WCGAN framework is similar to the typical GANs framework except for the discriminator part, which is termed as critic in WCGAN and Wasserstein distance is employed as the objective function. Unlike pix2pixGAN, in WCGAN it is not necessary to train generator and critic equal number of iterations and this ratio of updates to the critic for a single update of the generator can be treated as a hyperparameter. The architectures of generator and critic of WCGAN are shown in Table. 4.7 and 4.6 and their respective hyperparameters are depicted in 4.8. Training of WCGAN is carried out on EA1 only.

4.1.4.1 Grain boundary mapping

Besides reconstruction of complete 3D microstructures, an attempt to reconstruct mapping of 3D grain boundaries from 2D microstructures is also carried out using WCGAN architecture described in Table.4.6 and 4.7. The corresponding data has been generated by using scikit-image library [111] for 2D and 3D microstructures.

Table 4.6: Generator architecture in WCGAN.

Layer	**Operation**	**Output shape**	**kernel size**	**strides**	**Activation**	**Batch Normalization**
generator input = 2D input image (64, 64, 4) & $z \in \mathbb{R}^{100} \sim \mathcal{N}(0, I)$						
G_1	Conv3D	(32, 32, 32, 64)	(4,4,4)	(2,2,3)	ReLU	Yes
G_2	Conv3D	(16, 16, 16, 128)	(4,4,4)	(2,2,2)	ReLU	Yes
G_3	Conv3D	(8, 8,8, 256)	(4,4,4)	(2,2,2)	ReLU	Yes
G_4	Conv3D	(4, 4, 3, 512)	(4,4,4)	(2,2,2)	ReLU	Yes
G_5	Conv3D	(2, 2, 2, 512)	(4,4,4)	(2,2,2)	ReLU	Yes
G_6	Conv3DTranspose	(4, 4, 4, 512)	(4,4,4)	(2,2,2)	ReLU	Yes
G_7	Conv3DTranspose	(8, 8, 8, 512)	(4,4,4)	(2,2,2)	ReLU	Yes
G_8	Conv3DTranspose	(16, 16, 16, 256)	(4,4,4)	(2,2,2)	ReLU	Yes
G_9	Conv3DTranspose	(32, 32, 32, 128)	(4,4,4)	(2,2,2)	ReLU	Yes
G_{out}	Conv3DTranspose	(64, 64, 64, 1)	(4,4,4)	(2,2,2)	tanh	No

Table 4.7: Critic architecture in WCGAN.

Layer	**Operation**	**Output shape**	**kernel size**	**strides**	**activation**	**Batch Normalization**
Critic input = 3D (64, 64, 64, 4) and 2D (64, 64, 4) & $x \in \mathbb{R}^{64*64*64*4}$						
D_1	Conv3D	(32, 32, 22, 64)	(4,4,4)	(2,2,3)	leaky ReLU	Yes
D_2	Conv3D	(16, 16, 11, 128)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_3	Conv3D	(8, 8, 6, 256)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_4	Conv3D	(4, 4, 3, 512)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_5	Conv3D	(2, 2, 2, 1024)	(4,4,4)	(2,2,2)	leaky ReLU	Yes
D_7	Flatten	(8192)				
D_8	Dense	(10)			leaky ReLU	No
D_9	Dense	(10)			leaky ReLU	No
D_out	Dense	(1)			linear	No

Table 4.8: WCGAN-GP hyperparameters.

Symbol	**Hyperparameter**	**Value**
α_{gen}	learning rate for Generator	0.00009
α_{critic}	learning rate for Critic	0.00009
β_1	first decay rate of Adam optimizer	0
β_2	second decay rate of Adam optimizer	0.9
n_{critic}	number of critic updates w.r.t generator	5
λ	gradient penalty	10

Chapter 5

Results and discussions

5.1 Training results

The results obtained from training the models (discussed in Chapter. 4) are presented and analyzed in this chapter. It should be noted that the predictions shown were randomly chosen from their respective datasets (train/test). The synthetic microstructures obtained from DREAM.3D are termed as real images to avoid confusion and the generated microstructures are synonymous with predicted ones. For visual comparison, one of the Euler angle components (either EA0, EA1 or EA2) are plotted against the respective targets. Each component of Euler angles has various ranges. The values of EA0 and EA2 range from 0 to 2π while values in EA1 ranges between 0 and π.

5.1.1 Unet

Firstly, Unet is implemented with *Mean Absolute Error* as loss metric and constituting architecture described in Table. 4.1 for generating 3D microstructures using 2D slices. The model is trained for 4000 epochs and the evolution of loss is plotted as a function of epochs. It can be inferred from Fig. 5.1, that there is a smooth convergence of loss function as training advances. The trained model is then employed for predictions on the train and test datasets, which were later visually compared with real synthetic microstructures and depicted in Fig. 5.2 and 5.3 respectively. Fig. 5.2 illustrate the first component of Euler angles, EA0 in true and predicted microstructures. The cubes in Fig. 5.2 and 5.3 demonstrate the microstructures of true and predicted targets in different orientations and the values of phases and Euler angles in a single grain in microstructure are similar. From visual inspection, the predictions of Unet on the training dataset were identical to their 3D counterparts but mismatching on the test dataset. This is clearly evident from the Fig. 5.3, where there is no clear differentiation between individual grains. From a statistical perspective, the volume fraction and EA1 distribution of the predictions (from Fig. 5.4a) obtained from the training dataset were similar to the synthetic microstructures from DREAM.3D. These statistics (from Fig. 5.4b), however when compared on test dataset showed significant disparities.

5.1.1.1 With standard loss metric

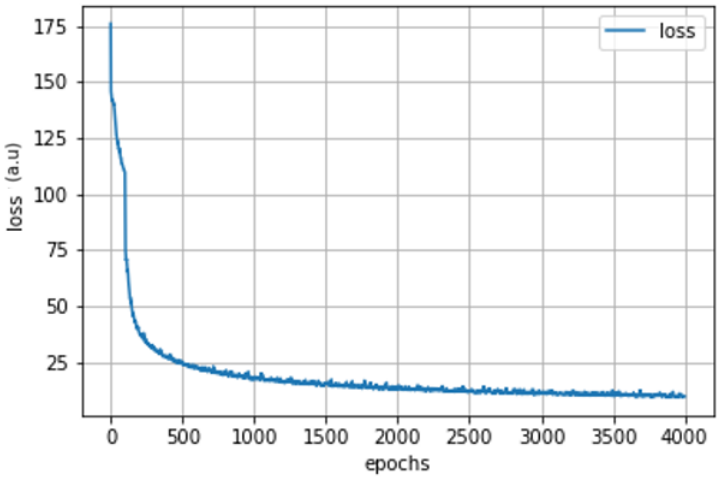


Figure 5.1: Loss curve obtained for Unet model trained with standard MAE for 4000 epochs.

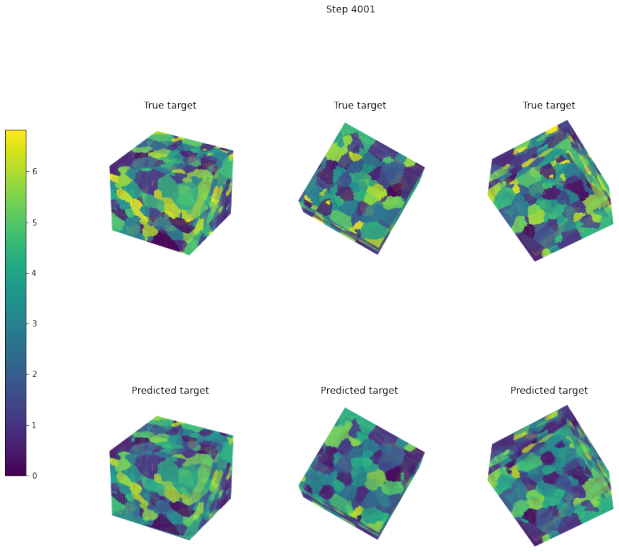


Figure 5.2: True and predicted EA0 for Unet with MAE on train data for 4000 epochs.

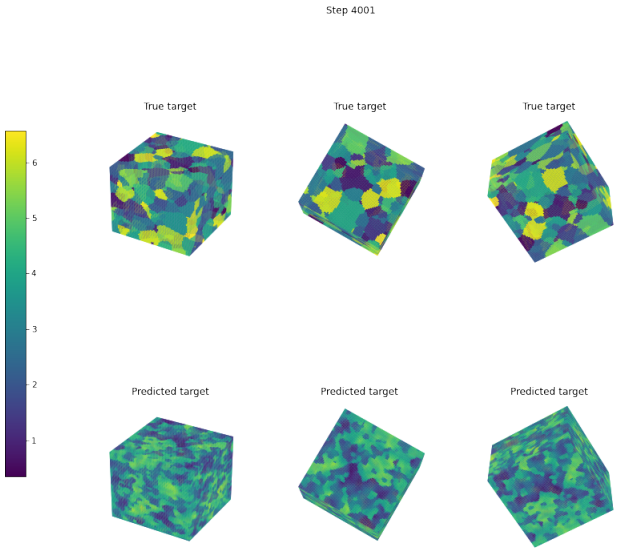
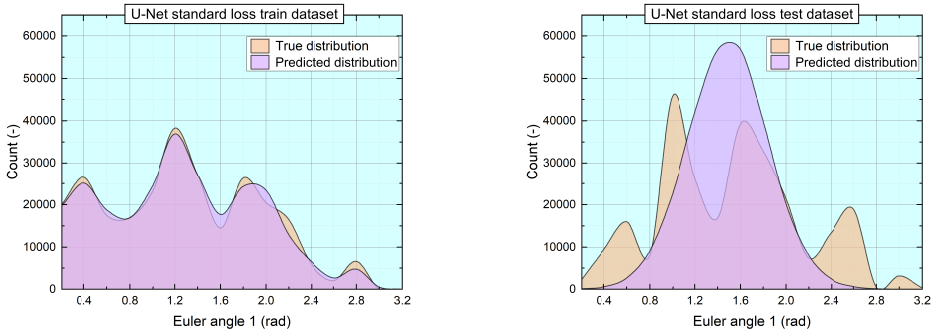


Figure 5.3: True and predicted EA0 for Unet trained with MAE on test data for 4000 epochs.



(a) (b)

Figure 5.4: Comparison of true and predicted distributions of EA1 from Unet model with MAE as loss metric trained for 4000 epochs on train (a) and test (b) datasets.

From analyzing the results of Unet with *Mean Absolute Error* as the loss metric, the model was incapable of predicting plausible outputs on the test dataset. So from this inference, if the loss function is provided with information related to volume fraction and EA1 distribution, the model can efficiently predict samples with identical statistics.

5.1.1.2 With custom loss function

From analyzing the results of Unet with *Mean Absolute Error* as the loss metric, the model was incapable of predicting plausible microstructures on the test dataset. The loss function is customized to include information of volume fraction and Euler angle distributions (obtained after application of Gaussian kernels on distributions of Euler angles). By employing this custom loss function (described in Section. 4.1.1.1), the model attempts to minimize the difference of volume fraction and Euler angle distributions between the predicted and target samples. The model was trained for 2000 epochs and the loss obtained after each epoch was plotted and given in Fig. 5.5. The loss curve in Fig. showed a smooth convergence.Fig. 5.6 and 5.7 depict the EA0 channel in true and predicted microstructures on train and test datasets respectively. On visual inspection, the true and predicted microstructures in the training dataset were identical, however, they showed severe disparities in the test dataset as the grain structure is not clearly evident. Fig. 5.8a and 5.8b illustrate the statistical comparison of a random sample in the corresponding datasets. The model predictions of volume fractions and Euler angle distributions on the test dataset have shown minor improvement compared to the Unet model trained with standard loss metrics.

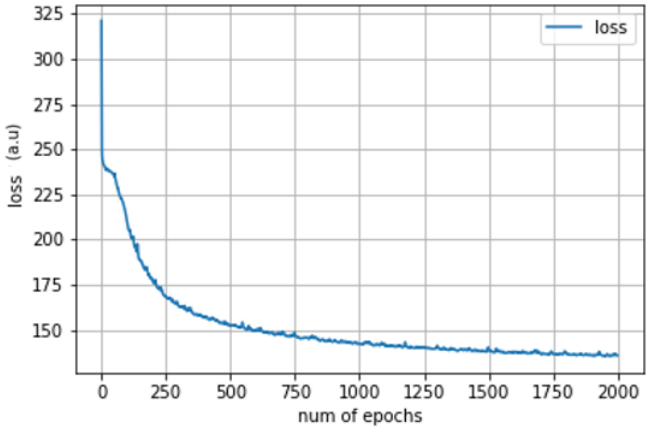


Figure 5.5: Loss curve of Unet trained with customized loss function for 2000 epochs.

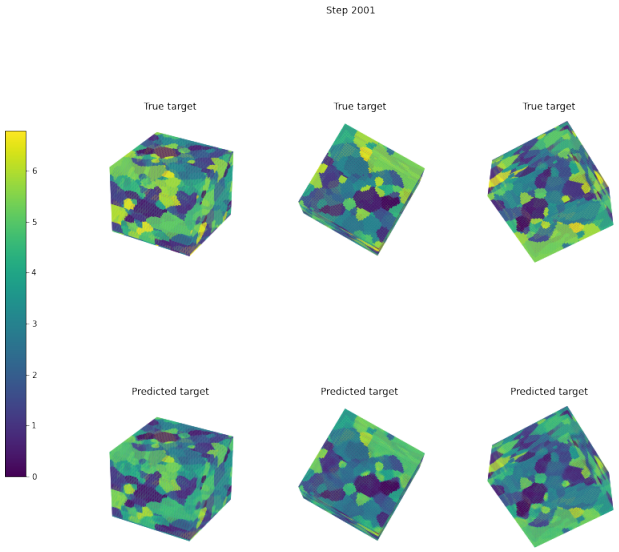


Figure 5.6: True and predicted EA0 for Unet trained with custom loss function on train data for 2000 epochs.

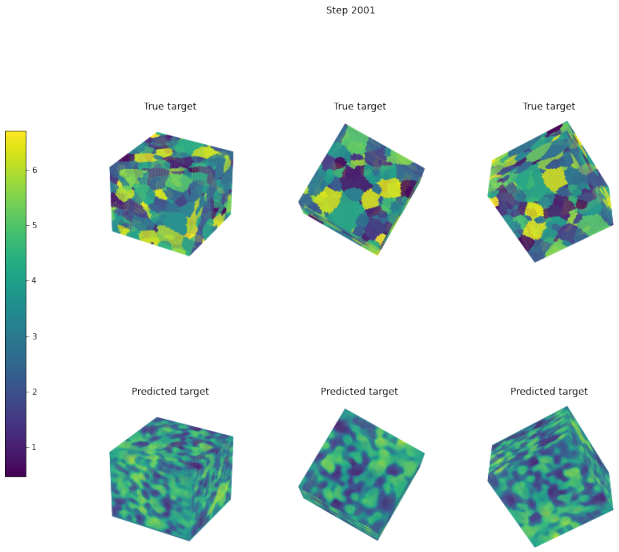
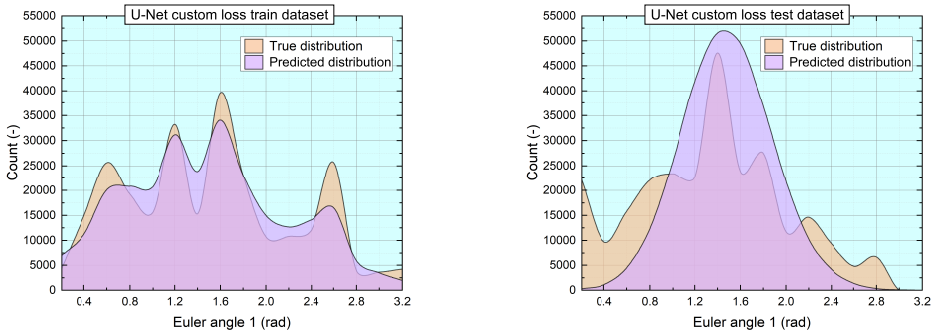


Figure 5.7: True and predicted EA0 for Unet trained with custom loss function on test data for 2000 epochs.



(a) (b)

Figure 5.8: Comparison of true and predicted distributions of EA1 from Unet model trained with customized loss for 2000 epochs on train (a) and test (b) datasets.

5.1.2 Conditional variational autoencoder

The cVAE model constitutes the framework described in Table. 4.2. This model is trained on all channels for 3000 epochs and the generated samples are compared with targets visually and statistically. The loss curve plotted between loss and number of epochs is depicted in Fig. 5.9 and it can be observed that the loss converged smoothly. The EA1 in true and predicted microstructures on train and test datasets were illustrated in Fig. 5.10 and 5.11 respectively.The predictions obtained were blurry and the grain boundaries are not clearly visible on visual examination. From the statistical point of view, the volume fraction and EA1 distributions (depicted in Fig. 5.12) in both train and test datasets were mostly similar to the target microstructures. The blurriness in the generated samples is a common problem of VAEs [112].

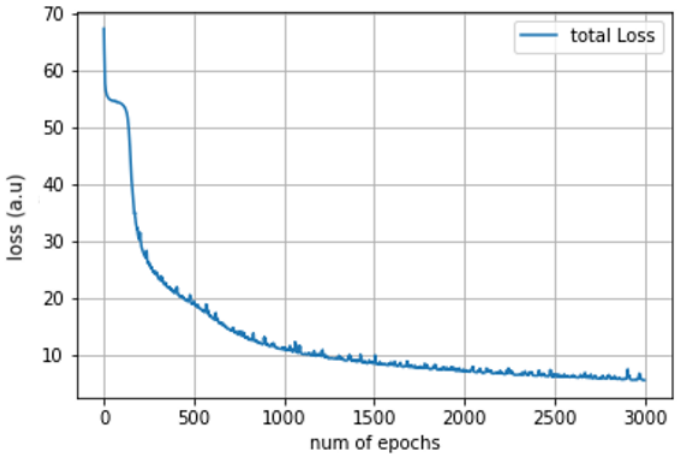


Figure 5.9: Loss curve obtained after training cVAE for 3000 epochs.

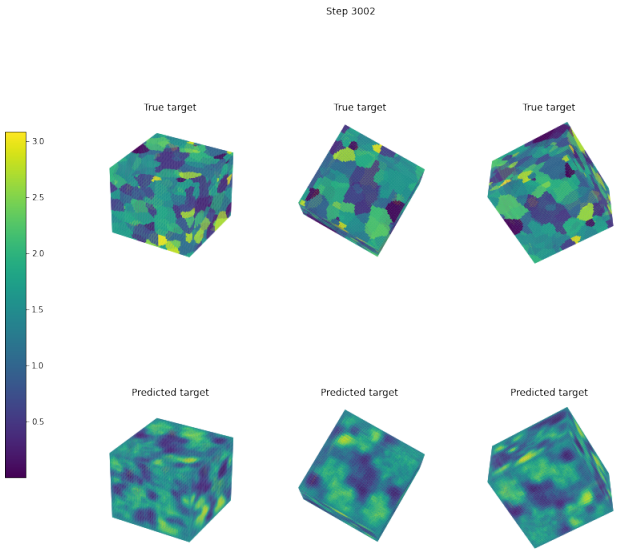


Figure 5.10: True and predicted EA1 for cVAE on train data after 3000 epochs.

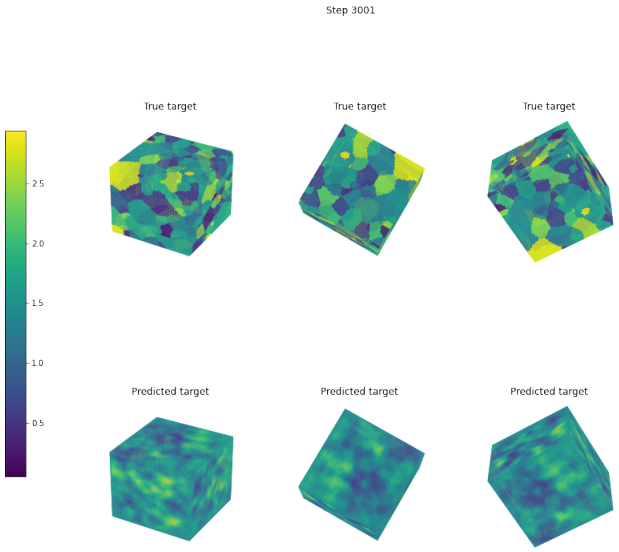
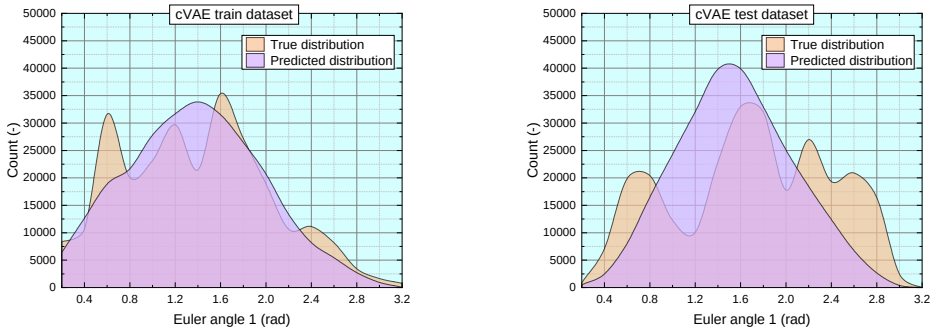


Figure 5.11: True and predicted EA1 for cVAE on test data after 3000 epochs.



(a) (b)

Figure 5.12: Comparison of true and predicted distributions of EA1 from cVAE model after 3000 epochs on train (a) and test (b) datasets.

5.1.3 Conditional generative adversarial networks

5.1.3.1 pix2pixGAN

As stated earlier in Section. 4.1.3, the model is trained on custom loss function for 3000 epochs. The loss curve is plotted for each step of the training batch i.e., for every batch, the

loss is calculated, and since there are 140 samples in the training dataset, 14 training steps complete an epoch. The generator loss is computed after each step and plotted against the number of steps and the resultant loss curve is shown in Fig.5.13. The true and generated EA0 channel in train and test datasets is visualised in Fig. 5.14 and 5.15 respectively.

The pix2pixGAN generated microstructures were similar to the true targets but the boundaries of the grains are not clearly defined. The grain structure is comparatively more evident than cVAE, however the grains cannot be segmented accurately as their boundaries are not distinct. The EA1 distributions in the true and generated microstructures are given in Fig. 5.16, it can be observed that the predictions were not completely identical with the true target distributions. The volume fractions of the generated microstructures were in proximity with the targets.

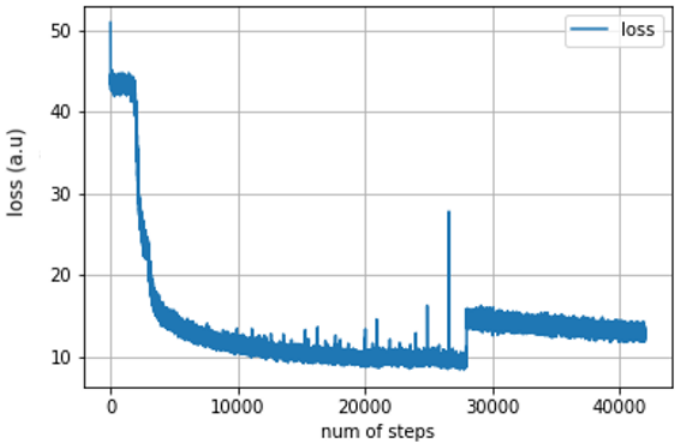


Figure 5.13: Loss curve obtained after training generator in pix2pixGAN for 3000 epochs

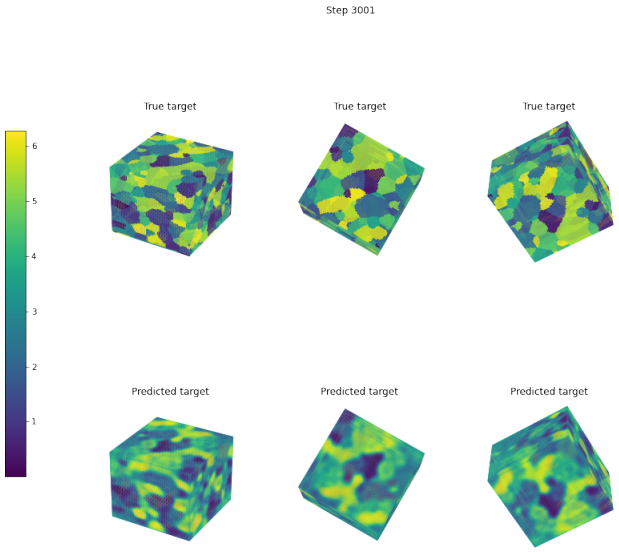


Figure 5.14: True and predicted EA0 for pix2pixGAN on train data after 3000 epochs.

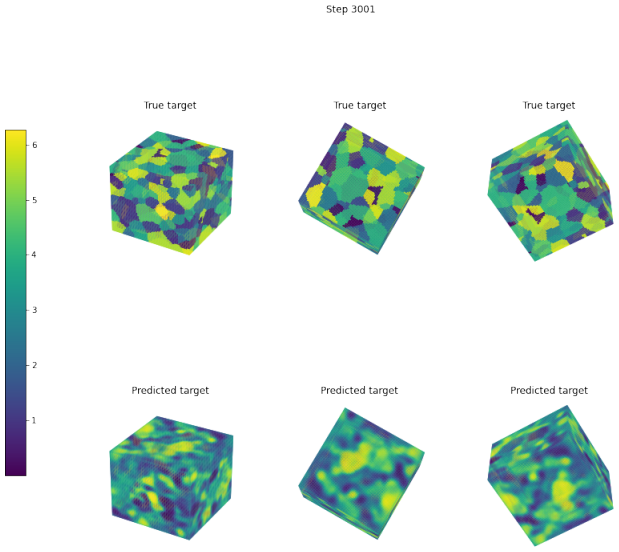
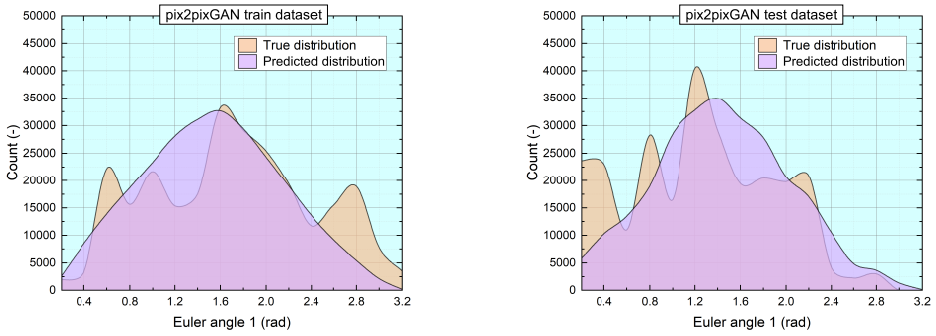


Figure 5.15: True and predicted EA0 for pix2pixGAN on test data after 3000 epochs.



(a) (b)

Figure 5.16: Comparison of true and predicted distributions of EA1 from pix2pixGAN model after 3000 epochs on train (a) and test (b) datasets.

5.1.3.2 Using Wasserstein loss with gradient penalty

Wasserstein GANs are employed with conditional setting with gradient penalty as Lipschitz constraint for microstructure reconstruction. The architecture and hyperparmeters of WCGAN is described in Section. 4.1.4 . The configured model is trained for 5000 epochs and the obtained loss is plotted as a function of epochs in Fig. 5.17. From the loss curve shown in Fig. 5.17, it can be inferred that there is no smooth convergence of loss and there are significant spikes in loss values during training the model. After examining the predictions visually, the predictions at 5000 epochs were very noisy on both train and test datasets which is evident from Fig 5.18 and 5.19. Interestingly, the predictions of WCGAN at 4000 epochs showed similar EA1 distributions, illustrated in 5.23a and 5.23b, albeit the visual predictions presented in from Fig. 5.21 and 5.22 were not identical.

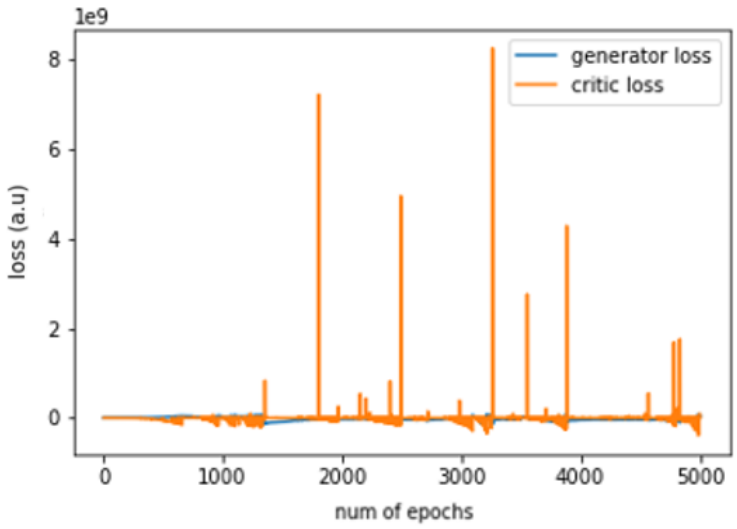


Figure 5.17: Loss curve of critic and generator obtained after training WCGAN for 5000 epochs.

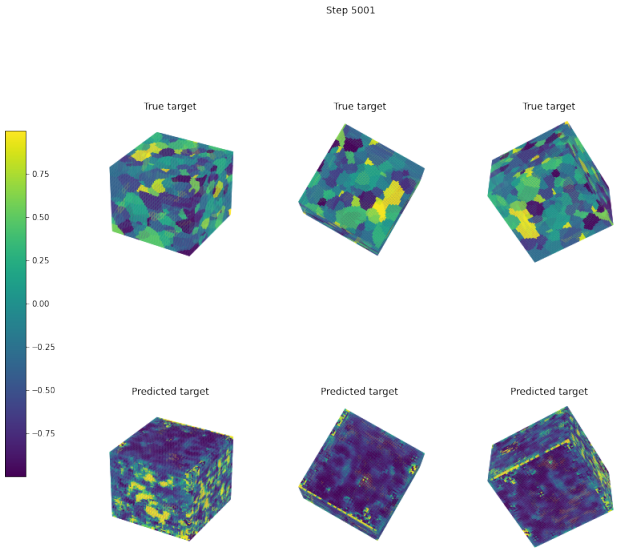


Figure 5.18: True and predicted EA1 (normalized) for WCGAN on train data after 5000 epochs.

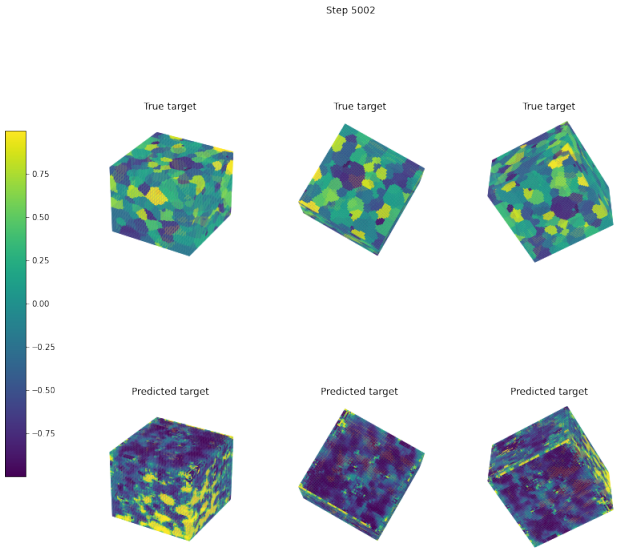
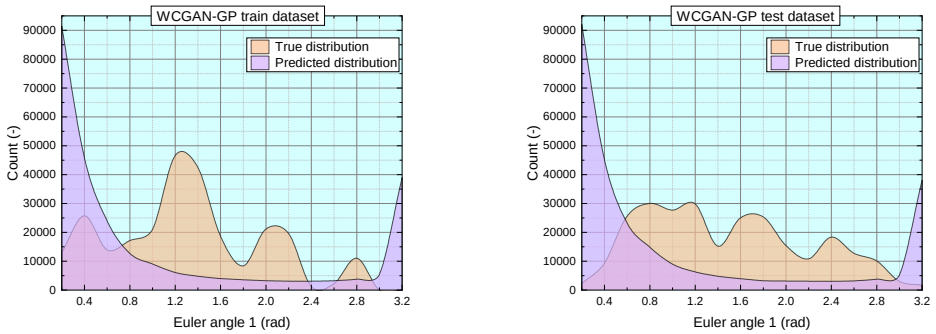


Figure 5.19: True and predicted EA1 (normalized) for WCGAN on test data after 5000 epochs.



(a) (b)

Figure 5.20: Comparison of true and predicted distributions of EA1 from WCGAN model after 5000 epochs on train (a) and test (b) datasets.

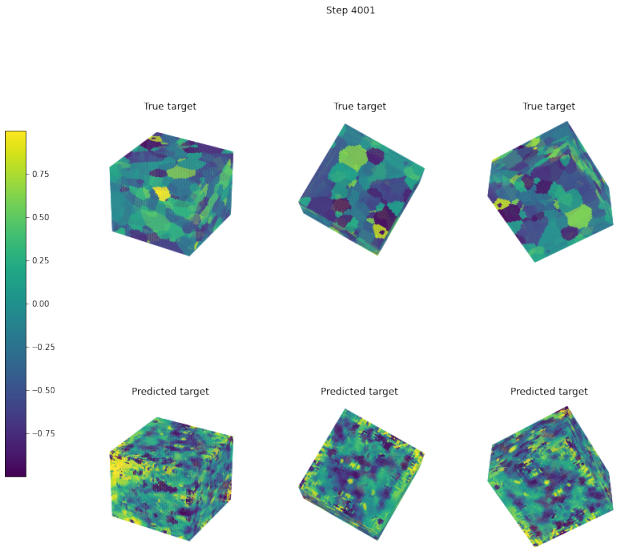


Figure 5.21: True and predicted EA1 (normalized) for WCGAN on train data after 4000 epochs.

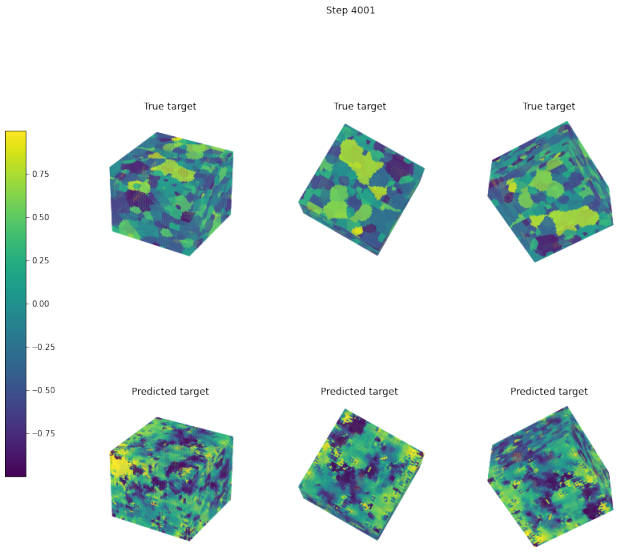
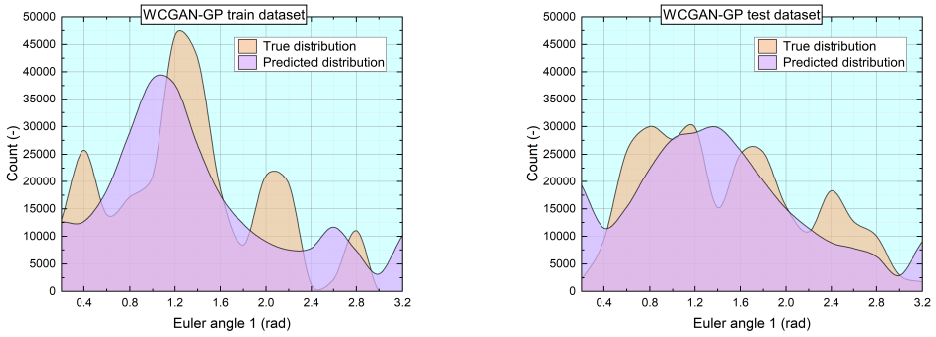


Figure 5.22: True and predicted EA1 (normalized) for WCGAN on test data after 4000 epochs.



(a) (b)

Figure 5.23: Comparison of true and predicted distributions of EA1 from WCGAN model after 4000 epochs on train (a) and test (b) datasets.

5.1.3.3 Grain boundary mapping

The predictions obtained from training WCGAN with gradient penalty on grain boundary mappings after 5000 epochs were illustrated in Fig. 5.25 and 5.26. From the loss curve in Fig. 5.24, the critic loss converged smoothly while the generator loss increased exponentially. This is not completely unusual as the generator loss computes the average of Dense layer outputs from critic while the critic loss calculates the difference between Dense layer outputs of true and generated images. In this case, the model failed in mapping the grain boundary in both training and test datasets which is apparent from Fig. 5.25 and 5.26.

Figure 5.24: Loss curve obtained after training WCGAN on grain boundary mapping for 5000 epochs.

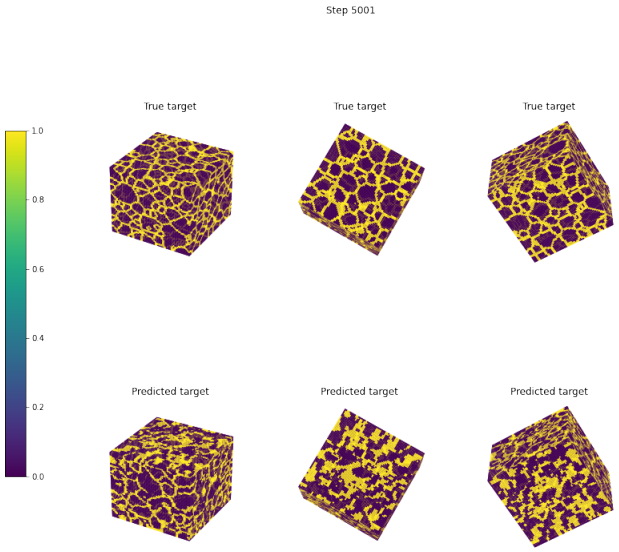


Figure 5.25: True and predicted mappings of grain boundaries from WCGAN on train data after 5000 epochs.

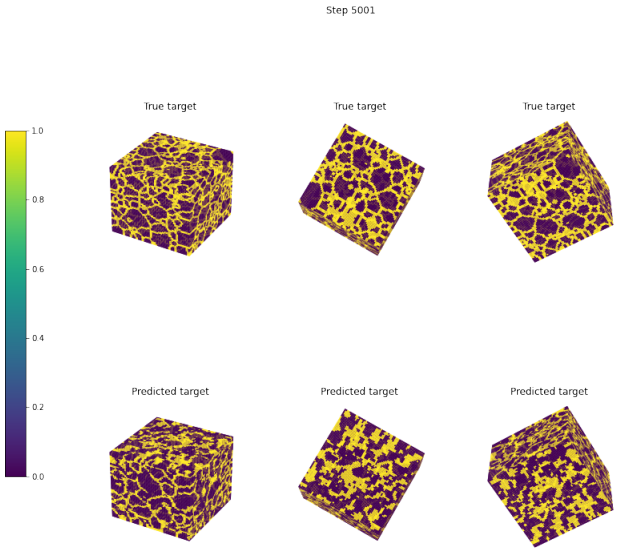


Figure 5.26: True and predicted mapping of grain boundaries from WCGAN on test data after 5000 epochs.

5.2 Overall analysis of the generated microstructures

In the above sections, the predictions after training from various generative models were presented. Initially, a single channel (generally EA0) is considered for training the model to determine the optimal values of hyperparameters and later on extended to all the channels. From analyzing their results, modifications are made in the speculated areas such as employing custom loss function or adjusting the complexity of the model for improving the fidelity of predictions. The volume fractions in generated and real microstructures are calculated on the test dataset for all the models employed in this work. The respective differences between true and predicted volume fractions of different models are plotted in a box plot and illustrated in Fig. 5.27.

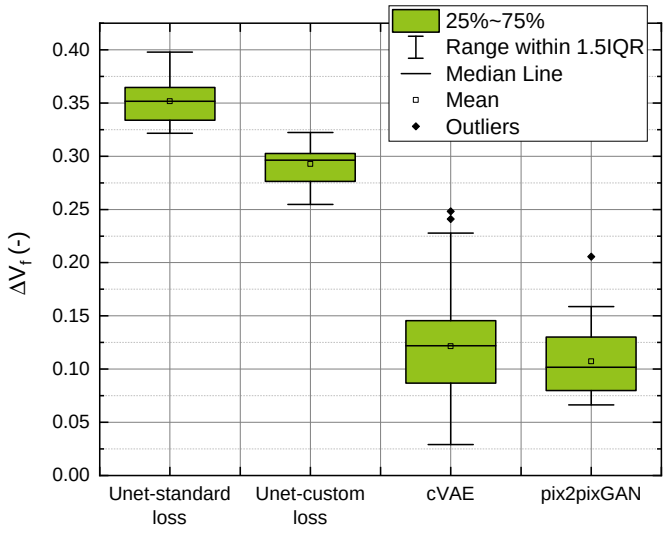


Figure 5.27: Comparison of difference in volume fraction predictions from different models on test dataset.

In the model training part, firstly, Unet with standard loss metrics is employed with some modifications to make it applicable for constructing 3D microstructures from an extruded 2D slice. After visual inspection of the generated images, the predictions on the train dataset were extremely similar to that of the real microstructures depicted in Fig. 5.2, however, the predictions on the test dataset were highly contrasting which can be seen in Fig. 5.3. Besides, the microstructure-descriptors such as volume fraction and Euler angle distributions of the generated microstructures in the train dataset were equivalent to target microstructures but showed severe discrepancies in test dataset. This is a case of severe overfitting, as the model learned the patterns by-heart from the training data and was inept in generalization on test data.

So, the loss function is modified to include volume fraction and Euler angle distributions in the Unet model and trained for 2000 epochs. When compared to previous predictions, the custom loss trained Unet showed less difference in volume fraction between real and generated microstructures which is evident from Fig. 5.27. We cannot get any concrete inferences by comparing the predictions visually from Fig. 5.3 and 5.7. Unet models accurately reproduced the EA1 distributions on the train dataset (depicted in Fig. 5.4a and 5.8a), however on test dataset, they were approximated to a Gaussian distribution (shown in Fig. 5.4b and 5.8b).

State of the art generative models such as VAEs and GANs were employed in a conditional setting by using cVAE, pix2pixGAN and WCGAN with gradient penalty. The cVAE, were trained using Kullback-Leibler divergence and reconstruction loss. The generated microstructures from cVAE showed indistinct grain boundaries (blurry edges) on both train and test datasets. Even though the cVAE model generated blurry microstructures, the volume fraction values of the generated microstructures from both train and test datasets were in the vicinity to that of the real ones. From Fig. 5.27, the mean difference in predictions of volume fraction in cVAE lies around 0.13 (from Fig. 5.27) which is much better than the Unet models (greater than 0.3). The EA1 distributions were quite identical on test and train datasets (illustrated in Fig. 5.12a and 5.12b) and have been generalized closely to a normal distribution. So, the cVAE model learnt the underlying patterns of the data instead of absolute distributions from the training dataset which is desirable.

Subsequently, pix2pixGAN is trained with a modified discriminator to include volume fraction and Euler angle distributions. After training the model for 3000 epochs, the generated microstructures illustrated distinguishable grain structures but with hazy grain boundaries. The microstructure-descriptors of the generated images also showed a satisfactory resemblance to the real microstructures. The predicted volume fraction values of microstructures from pix2pixGAN model are more equivalent to that of the real ones compared to cVAE. On visual examination, the grains in generated microstructures from pix2pixGAN are more distinguishable than the ones generated from cVAE.

Additionally, WCGAN with gradient penalty is trained using EA1 channel. WCGAN utilizes Wasserstein distance as a loss metric for training the model. Even though these are efficient in generating more realistic predictions, training them is a complicated task due to the involvement of hyperparameters such as gradient penalty parameter and ratio of critic updates w.r.t generator. After determining the hyperparameters in a heuristic process, the configured model has been trained for 5000 epochs and from visual and statistical analysis of generated microstructures, conclusions can be drawn that the model was incapable of learning grain structures and EA1 distributions which can be described in Fig. 5.18, 5.19, 5.20a and 5.20b. Since we cannot get any specific information from the loss curve, this may be attributed to the local minima problem as even the training predictions were also inaccurate. This argument is further supported by analysing the WCGAN model saved after 4000 epochs which showed better similarities in EA1 distributions and used in future analysis (illustrated in Fig. 5.21, 5.22, 5.23a and 5.23b). The statistical comparison of EA1 distributions depicted in Fig. 5.23 showed a good approximation of EA1 distributions. In fact, the estimation of

EA1 distribution is not completely exact but much better than the generated microstructures from pix2pixGAN and cVAE (can be observed from 5.16b and 5.12b). As the latter two predicted a seemingly Gaussian like distribution, the WCGAN model after 4000 epochs produced EA1 distribution similar to that of the real target microstructure. This led to an inference that employing EA1 for the initial training process resulted in better EA1 distributions which can be explained due to its Gaussian like distributions. Employing only EA1 channel for training purposes helps the model to learn the patterns of the data compared to training the model with EA0 or EA2 channels which have uniform distributions.

Even though the WCGAN generated a closer EA1 distribution, it is not further sought as the model could not reproduce microstructures with visible grain structures. So, instead of training the model to generate microstructures with phases and Euler angles, the WCGAN is trained on mapping grain boundaries. So the data is boundary cell data i.e., those cells which are part of grain boundary are explicitly marked and the model should be capable of generating them. After training the model for 5000 epochs, this model is unsuccessful in producing a contiguous and distinct grain like structure.

Table 5.1: Qualitative comparison of generative models.

Model	**Grain structure predictions**		**EA1 predictions**	
	Train	Test	Train	Test
Unet with *Mean Absolute Error*	++	- -	++	- -
Unet with custom loss metric	++	-	++	-
cVAE	++	+	+	+
pix2pixGAN	++	++	+	+
WCGAN-GP (saved after 4000 epochs)	-	-	++	++

The shortcoming of the generative models employed in this work is the generalization over the samples which are not under the training dataset. Unet is employed as a baseline model for microstructure reconstruction, was able to reconstruct the microstructures perfectly which were supplied during the training process. It failed to capture the statistics of the underlying data and reconstruct plausible microstructures on the test dataset. The blurry microstructures generated by cVAE framework is foreseen and a common problem of VAEs. The pix2pixGAN model is trained with a customized discriminator by incorporating the volume fractions and Euler angle distributions during optimization. It generated microstructures with distinguishable grain structures but indistinct grain boundaries. Although the pix2pixGAN and cVAE models were comparatively efficient in emulating microstructure-descriptors, the generated microstructures are still not adequate enough to employ it in micromechanical simulations and require further optimization. Using image postprocessing tools from DREAM.3D, the microstructures generated from pix2pixGAN are postprocessed for investigating the microstructure characteristics.

WCGAN with gradient penalty captured the underlying statistics of EA1 quite well but ineffectual in learning the grain structure. Wasserstein distance as objective function in WCGAN proved to be effective in minimizing the difference in EA1 distributions.

Chapter 6

Postprocessing using DREAM.3D

6.1 Postprocessing pipeline

From the results of trained models, it is evident that the predictions from Unet, pix2pixGAN, and WCGAN failed in the clear demarcation of grains. Although the grain structure is evident for some predictions on the test dataset, the grain boundary is not clearly defined. Even the grain size information cannot be calculated directly from the generated microstructures, so there is a need for postprocessing for better analysis of microstructure-descriptors. A postprocessing pipeline is developed in DREAM.3D which is demonstrated in Table. 1 in Appendix B and the input microstructure should contain both phases and Euler angle channels. From the below mentioned filters, only two are significant for grouping the voxels for forming grain structures namely, *Segment Features (Misorientation)* and *Minimum Size*. The *Segment Features (Misorientation)* filter groups the nearby voxels by segmenting them to form a single grain based on a user-defined variable *Misorientation tolerance*. This field variable indicates the minimum tolerance in misorientation angle required to group the neighboring voxels into a single grain. The latter filter i.e., *Minimum Size* takes the input parameter *Minimum Allowed Feature Size* which denotes the minimum number of voxels necessary to form grain. These two variables can be adjusted for tuning the microstructures in terms of grain size distributions and the number of grains in the postprocessed microstructures. Nevertheless, the parameters for *Minimum Size* and *Minimum Allowed Feature Size* should be chosen in such a manner that the average grain size and number of features of the postprocessed microstructure were matching with the real target microstructure.

6.2 Postprocessing of pix2pixGAN predictions

The postprocessing pipeline (see Table. 1 in Appendix B) has been employed on microstructures predicted from pix2pixGAN on test dataset. The postprocessed microstructures are further analyzed quantitatively by comparing the mean ESD and the number of grains, with real microstructures. Analysis of postprocessed images reflected

several concerns related to the statistics, as we are able to obtain a perfect grain like structure with clearly demarcated grain boundaries which can be observed in Fig. 6.1 and 6.3. The postprocessed microstructure and real microstructure have noticeable differences when perceived visually which is expected, As we are only trying to mimic the statistics of the input to reconstruct a statistically equivalent microstructure. The average misorientations between each voxel in the postprocessed and real microstructures are demonstrated in Fig. 6.2a and 6.4a. Furthermore, their respective grain size distributions are compared in Fig. 6.2b and 6.4b.

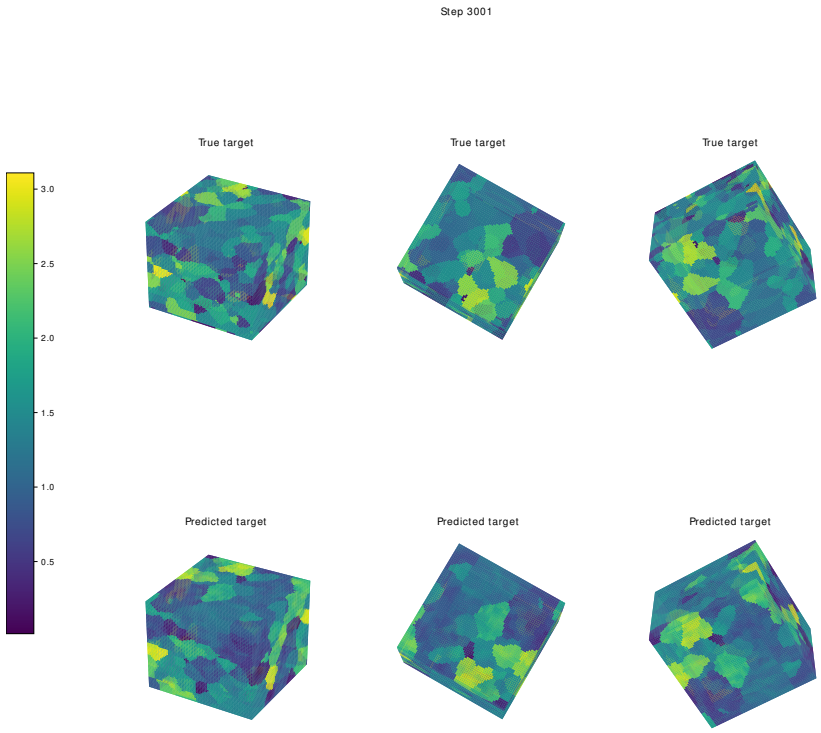
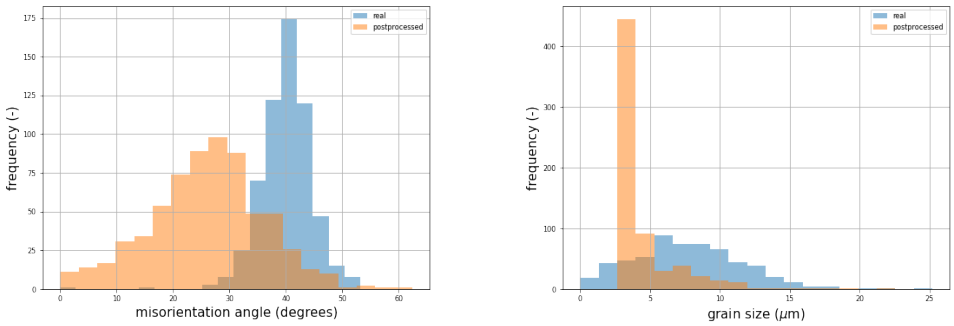


Figure 6.1: True and postprocessed predictions EA1 from pix2pixGAN on train dataset.



(a) (b)

Figure 6.2: Comparison of misorientation angles (a) and grain size distributions (b) after postprocessing predicted microstructures from pix2pixGAN on train dataset.

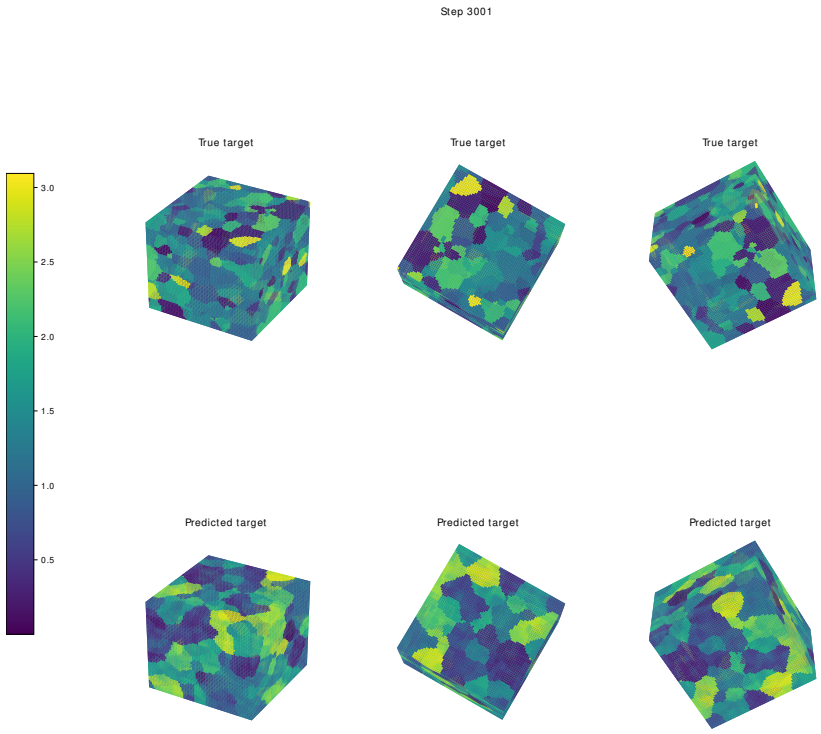
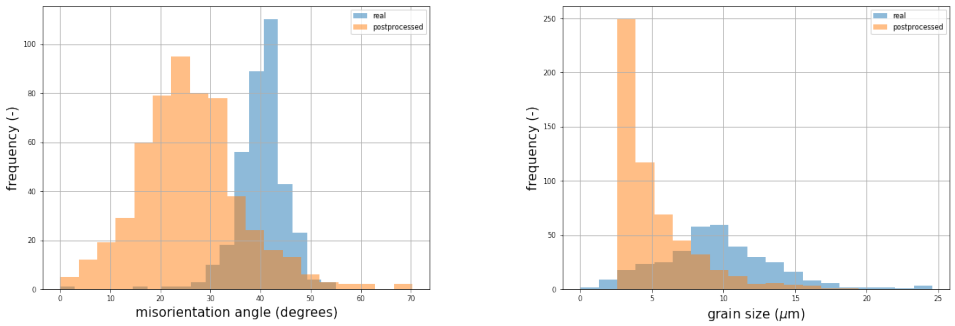


Figure 6.3: True and postprocessed predictions EA1 from pix2pixGAN on test dataset.



(a) Misorientation angles comparison (b) comparison of grain size distribution

Figure 6.4: Comparison of misorientation angles (a) and grain size distributions (b) after postprocessing predicted microstructures from pix2pixGAN on test dataset.

Besides, the postprocessing pipeline is applied on some test samples of pix2pixGAN, and the number of grains and mean ESD are computed. The two parameters *Misorientation tolerance* and *Minimum Allowed Feature Size* parameters are selected such that the average grain size and the number of features are closest to the real microstructures. It is quite interesting that the number of grains and mean ESD are not identical as shown in Fig. 6.5a and 6.5b. As one would expect the mean ESD and the number of grains are inversely proportional, which is evident from Figs. 6.5a and 6.5b. However, there are inconsistencies in the volume fraction of the samples after postprocessing which are not analyzed in this work.

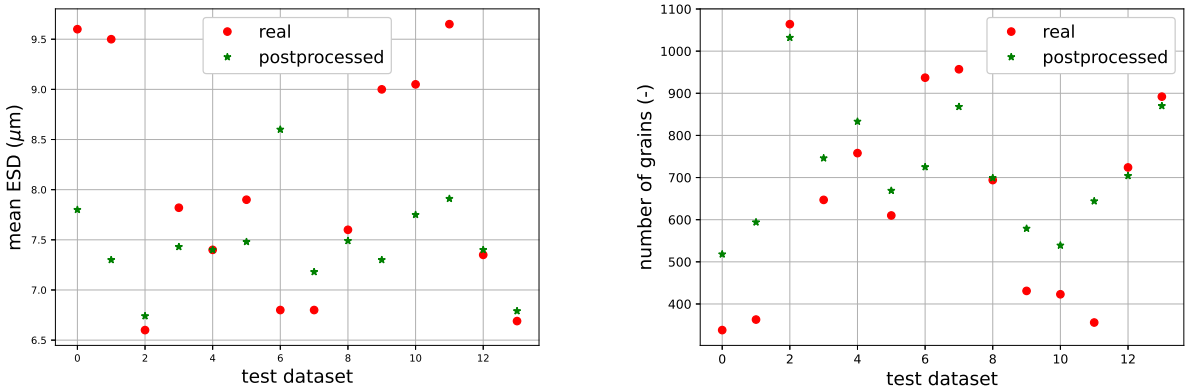


Figure 6.5: Analyzing postprocessed microstructures quantitatively in terms of mean ESD and number of grains.

Chapter 7

Conclusions

7.1 Summary

This work is an ambitious effort to reconstruct 3D statistically equivalent microstructures from 2D microstructures by employing generative models. It mainly focuses on microstructure reconstruction and does not involve property predictions. The microstructure is quantified by defining the microstructure-descriptors namely volume fraction, grain size and grain orientation information. Since the accumulation of data from experiments for the given objective is not only expensive but practically impossible due to the enormous efforts involved. So, DREAM.3D has been employed for synthetic microstructure generation by capitalizing the microstructure-descriptors.

Furthermore, an algorithm is also developed to include lamellae in the synthetic microstructures. It is a postprocessing tool applied on synthetic microstructures to augment generated polycrystals with additional features to make them more realistic. It involves modifying the existing grain information by taking the geometrical aspects of the grain into account. The required parameters such as distance between lamellae, thickness, orientation, and normal direction of the lamellae can be utilized for customizing them.

The initially chosen dimensions for a synthetic 3D microstructure constructed from DREAM.3D is $192 \times 192 \times 192$ and a relative 2D microstructure is acquired by slicing the 3D volume. Comparing the statistics of obtained 2D slices with that of the entire 3D volume, the 2D slice which accurately describes the 3D volume is selected. A dataset of 165 data points constituting 3D and corresponding 2D microstructures has been prepared. In consideration of the available computational resources, the dimensions of synthetic microstructures have been reduced to $64 \times 64 \times 64$ for faster training.

Initially, Unet framework has been employed for microstructure reconstruction using standard loss metrics and employing 2D slice as input. The Unet predictions failed in generalization of data points, resulted in overfitting of training samples, and showed severe disparities in the microstructure-descriptors which hinted at optimizing the objective function. Subsequently, the Unet is trained with a customized loss function to get more plausible outputs. The custom objective function improved the generated microstructures

slightly, however it is not sufficient for generating high-fidelity microstructures.

State of the art generative models namely VAEs and GANs have been applied for reconstructing the synthetic microstructures using 2D slice as a conditional setting. The cVAE model generated blurry images with minor deviations in microstructure-descriptors. The cGANs constituting pix2pixGAN framework has been employed for microstructure reconstruction and its predictions had shown decent similarities with the targets. It also showed competence in learning the microstructure-descriptors, and the generated microstructures showed visually distinguishable grain like structures, however with opaque grain boundaries. This emphasized the need for postprocessing for better analysis of the generated microstructures. The postprocessed microstructures showed visually distinguishable grains with distinct grain boundaries and are also effectual in quantitative comparison of the generated microstructures with real ones. The postprocessed microstructures displayed some disparities when microstructure-descriptors are compared.

Besides, WCGAN with gradient penalty has been also employed for this purpose and trained only with a single channel. The resultant microstructures did not show any visible grain like regions, however competent in reproducing EA1 distributions. WCGAN was also been applied for mapping grain boundary solely, however, it could not predict grain boundaries satisfactorily. From the above mentioned generative models, pix2pixGAN trained with customized loss function indicated potential in generating microstructures with similar microstructure-descriptors.

7.2 Recommendations for future work

The present work has the potential to reconstruct a 3D microstructure from the given 2D slice rapidly on-the-go compared to the conventional algorithms which require considerable time and computational resources. For instance, the time taken for generating a synthetic 3D microstructure by employing present algorithms can be reduced from several minutes to few minutes or seconds. However, this work demands high computational resources only during training the configured models. Even though the generative models employed in this work were not completely successful in the generation of statistically equivalent microstructures, they provided valuable insights into microstructure reconstruction. Firstly, one needs to comprehend the complexity in this task to reconstruct a 3D microstructure of dimensions $64 \times 64 \times 64$ from a 2D image of size 64×64. Another significant insight is customizing the evaluation metrics by the inclusion of descriptors improved the accuracy in generated microstructures in terms of statistics. The efficiency of the model can be enhanced further by customizing the loss metrics, and the current evaluation metrics for the comparison of two 3D images require further research.

Since postprocessing the predictions showed promising results, it can be included immediately after training the generative model on a single channel specifically EA1 (owing to its Gaussian like distribution) using pix2pixGAN architecture with more effective evaluation metrics. Subsequently filling the other Euler angle components and phases channels from the information obtained from 2D slice using segmentation algorithms.

Optimization of latent space in generative models can be performed to achieve more realistic microstructures. There is also scope for expanding the microstructure-descriptors space (such as aspect ratios and number of neighbors) to incorporate the possible variance in the dataset which is critical for the generalization of microstructures. Although the main objective of this work is to reconstruct statistically equivalent synthetic microstructures, the ultimate aim is to expand this framework for experimental and complex microstructures. Nevertheless, the present proposed approach is still evolving and has huge potential in microstructure reconstruction.

Bibliography

[1] Ankit Agrawal and Alok Choudhary. Perspective: Materials informatics and big data: Realization of the "fourth paradigm" of science in materials science. *APL Materials*, 4(5):053208, 2016.

[2] National Science and Technology Council (US). *Materials genome initiative for global competitiveness*. Executive Office of the President, National Science and Technology Council, 2011.

[3] John R Rodgers and David Cebon. Materials informatics. *MRS bulletin*, 31(12):975–980, 2006.

[4] Kim F Ferris, Loni M Peurrung, and James M Marder. Materials informatics: Fast track to new materials. *Advanced Materials & Processes, 165 (1): 50-51*, 165(PNNL-SA-52427), 2007.

[5] Hongyi Xu, Ruoqian Liu, Alok Choudhary, and Wei Chen. A machine learning-based design representation method for designing heterogeneous microstructures. *Journal of Mechanical Design*, 137, 05 2015.

[6] Ahmet Cecen, Hanjun Dai, Yuksel C. Yabansu, Surya R. Kalidindi, and Le Song. Material structure-property linkages using three-dimensional convolutional neural networks. *Acta Materialia*, 146:76–84, 2018.

[7] Gregory B Olson. Computational design of hierarchically structured materials. *Science*, 277(5330):1237–1242, 1997.

[8] David T Fullwood, Stephen R Niezgoda, Brent L Adams, and Surya R Kalidindi. Microstructure sensitive design for performance optimization. *Progress in Materials Science*, 55(6):477–562, 2010.

[9] Daniel Şopu, Celal Soyarslan, Baran Sarac, Swantje Bargmann, Mihai Stoica, and Jürgen Eckert. Structure-property relationships in nanoporous metallic glasses. *Acta materialia*, 106:199–207, 2016.

[10] Bradley S Fromm, Kunok Chang, David L McDowell, Long-Qing Chen, and Hamid Garmestani. Linking phase-field and finite-element modeling for process–structure–property relations of a ni-base superalloy. *Acta materialia*, 60(17):5984–5999, 2012.

[11] Yuksel C Yabansu and Surya R Kalidindi. Representation and calibration of elastic localization kernels for a broad class of cubic polycrystals. *Acta Materialia*, 94:26–35, 2015.

[12] Ruoqian Liu, Yuksel C Yabansu, Zijiang Yang, Alok N Choudhary, Surya R Kalidindi, and Ankit Agrawal. Context aware machine learning approaches for modeling elastic localization in three-dimensional composite microstructures. *Integrating Materials and Manufacturing Innovation*, 6(2):160–171, 2017.

[13] Anh Tran and Hoang Tran. High-fidelity 2d microstructure reconstruction via non-local patch-based image inpainting. *Acta Materialia*, 08 2019.

[14] Henning Friis Poulsen, Søren Fæster Nielsen, Erik M Lauridsen, Søren Schmidt, RM Suter, U Lienert, L Margulies, T Lorentzen, and D Juul Jensen. Three-dimensional maps of grain boundaries and the stress state of individual grains in polycrystals and powders. *Journal of applied crystallography*, 34(6):751–756, 2001.

[15] X Fu, HF Poulsen, Søren Schmidt, Søren Fæster Nielsen, EM Lauridsen, and D Juul Jensen. Non-destructive mapping of grains in three dimensions. *Scripta materialia*, 49(11):1093–1096, 2003.

[16] Romain Quey, PR Dawson, and Fabrice Barbe. Large-scale 3d random polycrystals for the finite element method: Generation, meshing and remeshing. *Computer Methods in Applied Mechanics and Engineering*, 200(17-20):1729–1745, 2011.

[17] Michael A Groeber and Michael A Jackson. Dream. 3d: a digital representation environment for the analysis of microstructure in 3d. *Integrating materials and manufacturing innovation*, 3(1):56–72, 2014.

[18] N Vajragupta, P Wechsuwanmanee, J Lian, M Sharaf, S Münstermann, A Ma, A Hartmaier, and W Bleck. The modeling scheme to evaluate the influence of microstructure features on microcrack formation of dp-steel: The artificial microstructure model and its application to predict the strain hardening behavior. *Computational materials science*, 94:198–213, 2014.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[20] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[21] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[22] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[23] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[24] Masaharu Yamaguchi. Tial alloys: Phase transformation, microstructure and mechanical properties. *Advanced Light Alloys and Composites*, pages 137–146, 1998.

[25] Fritz Appel, Jonathan David Heaton Paul, and Michael Oehring. *Gamma titanium aluminide alloys: science and technology*. John Wiley & Sons, 2011.

[26] R Gerling, H Clemens, and FP Schimansky. Powder metallurgical processing of intermetallic gamma titanium aluminides. *Advanced Engineering Materials*, 6(1-2):23–38, 2004.

[27] Young-Won Kim. Ordered intermetallic alloys, part iii: gamma titanium aluminides. *Jom*, 46(7):30–39, 1994.

[28] Marc Thomas, Thierry Malot, and Pascal Aubry. Laser metal deposition of the intermetallic tial alloy. *Metallurgical and Materials Transactions A*, 48(6):3143–3158, 2017.

[29] MH Loretto, AB Godfrey, D Hu, PA Blenkinsop, IP Jones, and TT Cheng. The influence of composition and processing on the structure and properties of tial-based alloys. *Intermetallics*, 6(7-8):663–666, 1998.

[30] L Gélébart, Michel Bornert, T Bretheau, D Caldemaison, Jérôme Crépin, and A Zaoui. Lamellar grains distribution and plastic strain heterogeneities in tiai cast samples. experiments and modelling. *Matériaux & Techniques*, 92(1-2):69–76, 2004.

[31] Xinhua Wu. Review of alloy and process development of tial alloys. *Intermetallics*, 14(10-11):1114–1122, 2006.

[32] Sung G Pyo, YW Chang, and Nack J Kim. Microstructure and mechanical properties of duplex tial alloys containing mn. *Metals and Materials*, 1(2):107–115, 1995.

[33] Thywill Dzogbewu. Additive manufacturing of tial-based alloys. *Manufacturing Review*, 7, 12 2020.

[34] Rodney Hill. Elastic properties of reinforced solids: some theoretical principles. *Journal of the Mechanics and Physics of Solids*, 11(5):357–372, 1963.

[35] Zvi Hashin and Shmuel Shtrikman. A variational approach to the theory of the elastic behaviour of multiphase materials. *Journal of the Mechanics and Physics of Solids*, 11(2):127–140, 1963.

[36] Z am Hashin and S Shtrikman. On some variational principles in anisotropic and nonhomogeneous elasticity. *Journal of the Mechanics and Physics of Solids*, 10(4):335–342, 1962.

[37] Helmut J Böhm. A short introduction to basic aspects of continuum micromechanics. *Cdl-fmd Report*, 3, 1998.

[38] JR Willis. Elasticity theory of composites. In *Mechanics of solids*, pages 653–686. Elsevier, 1982.

[39] Shriram Swaminathan, Somnath Ghosh, and NJ Pagano. Statistically equivalent representative volume elements for unidirectional composite microstructures: Part i-without damage. *Journal of Composite materials*, 40(7):583–604, 2006.

[40] Ramin Bostanabad, Yichi Zhang, Xiaolin Li, Tucker Kearney, L Catherine Brinson, Daniel W Apley, Wing Kam Liu, and Wei Chen. Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques. *Progress in Materials Science*, 95:1–41, 2018.

[41] GW Milton. The theory of composites (cambridge monographs on applied and computational mathematics) cambridge university press. *Cambridge, UK*, 2002.

[42] SA Tabei, A Sheidaei, M Baniassadi, F Pourboghrat, and H Garmestani. Microstructure reconstruction and homogenization of porous ni-ysz composites for temperature dependent properties. *Journal of Power Sources*, 235:74–80, 2013.

[43] Majid Baniassadi, Behzad Mortazavi, H Amani Hamedani, Hamid Garmestani, Said Ahzi, Madjid Fathi-Torbaghan, David Ruch, and M Khaleel. Three-dimensional reconstruction and homogenization of heterogeneous materials using statistical correlation functions and fem. *Computational Materials Science*, 51(1):372–379, 2012.

[44] David T Fullwood, Brent L Adams, and Surya R Kalidindi. A strong contrast homogenization formulation for multi-phase anisotropic materials. *Journal of the Mechanics and Physics of Solids*, 56(6):2287–2297, 2008.

[45] Brent L Adams and T Olson. The mesostructure—properties linkage in polycrystals. *Progress in Materials Science*, 43(1):1–87, 1998.

[46] Veeraraghavan Sundararaghavan and Nicholas Zabaras. Classification and reconstruction of three-dimensional microstructures using support vector machines. *Computational Materials Science*, 32(2):223–239, 2005.

[47] Mikhail Tashkinov. Statistical methods for mechanical characterization of randomly reinforced media. *Mechanics of Advanced Materials and Modern Processes*, 3(1):1–18, 2017.

[48] SR Niezgoda, DT Fullwood, and SR Kalidindi. Delineation of the space of 2-point correlations in a composite material system. *Acta Materialia*, 56(18):5285–5292, 2008.

[49] Salvatore Torquato and HW Haslach Jr. Random heterogeneous materials: microstructure and macroscopic properties. *Appl. Mech. Rev.*, 55(4):B62–B63, 2002.

[50] David T Fullwood, Stephen R Niezgoda, and Surya R Kalidindi. Microstructure reconstructions from 2-point statistics using phase-recovery algorithms. *Acta Materialia*, 56(5):942–948, 2008.

[51] Yang Jiao, FH Stillinger, and S Torquato. Modeling heterogeneous materials via two-point correlation functions. ii. algorithmic details and applications. *Physical Review E*, 77(3):031135, 2008.

[52] Cedric J Gommes, Yang Jiao, and Salvatore Torquato. Microstructural degeneracy associated with a two-point correlation function and its information content. *Physical Review E*, 85(5):051140, 2012.

[53] Hongyi Xu, Dmitriy A Dikin, Craig Burkhart, and Wei Chen. Descriptor-based methodology for statistical characterization and 3d reconstruction of microstructural materials. *Computational Materials Science*, 85:206–216, 2014.

[54] P. G. T. Howell. Stereological methods vol. 2: Theoretical foundation: By e. r. weibel academic press inc., london, 1980 340 pages, many figures and tables isbn 0-12-742202-1. *Scanning*, 4(4):208–208, 1981.

[55] S Wilson, C Hefferen, F Li, R Suter, and AD Rollett. Microstructural characterization and evolution in 3d. In *Proceedings 31st International Risoe Symposium*, 2010.

[56] Stuart I Wright, Matthew M Nowell, René de Kloe, Patrick Camus, and Travis Rampton. Electron imaging with an ebsd detector. *Ultramicroscopy*, 148:132–145, 2015.

[57] William F.. Smith, Javad Hashemi, and Francisco Presuel-Moreno. *Foundations of materials science and engineering*. Mcgraw-Hill Publishing, 2006.

[58] Jonathan E Spowart. Automated serial sectioning for 3-d analysis of microstructures. *Scripta Materialia*, 55(1):5–10, 2006.

[59] Jonathan E Spowart, Herbert E Mullens, and Brian T Puchala. Collecting and analyzing microstructures in three dimensions: a fully automated approach. *Jom*, 55(10):35–37, 2003.

[60] McLean P Echlin, Alessandro Mottura, Christopher J Torbet, and Tresa M Pollock. A new tribeam system for three-dimensional multimodal materials analysis. *Review of Scientific Instruments*, 83(2):023701, 2012.

[61] EA Wargo, AC Hanna, A Cecen, SR Kalidindi, and EC Kumbur. Selection of representative volume elements for pore-scale analysis of transport in fuel cell materials. *Journal of power sources*, 197:168–179, 2012.

[62] Paul G Kotula, Gregory S Rohrer, and Michael P Marsh. Focused ion beam and scanning electron microscopy for 3d materials characterization. *MRS Bulletin*, 39(4):361–365, 2014.

[63] Julie Villanova, Peter Cloetens, Heikki Suhonen, Jérôme Laurencin, François Usseglio-Viretta, Elisa Lay, Gérard Delette, Pierre Bleuet, David Jauffrès, Denis Roussel, et al. Multi-scale 3d imaging of absorbing porous materials for solid oxide fuel cells. *Journal of Materials Science*, 49(16):5626–5634, 2014.

[64] Martin Ebner, Felix Geldmacher, Federica Marone, Marco Stampanoni, and Vanessa Wood. X-ray tomography of porous, transition metal oxide based lithium ion battery electrodes. *Advanced Energy Materials*, 3(7):845–850, 2013.

[65] Oliver Betz, Ulrike Wegst, Daniela Weide, Michael Heethoff, Lukas Helfen, WAH-KEAT LEE, and Peter Cloetens. Imaging applications of synchrotron x-ray phase-contrast microtomography in biological morphology and biomaterials science. i. general aspects of the technique and its advantages in the analysis of millimetre-sized arthropod structure. *Journal of Microscopy*, 227(1):51–71, 2007.

[66] Alexandre Stienon, Arnaud Fazekas, J-Y Buffière, Alain Vincent, P Daguier, and F Merchi. A new methodology based on x-ray micro-tomography to estimate stress concentrations around inclusions in high strength steels. *Materials Science and Engineering: A*, 513:376–383, 2009.

[67] Henry Proudhon, Jean-Yves Buffière, and Siegfried Fouvry. Three-dimensional study of a fretting crack using synchrotron x-ray micro-tomography. *Engineering Fracture Mechanics*, 74(5):782–793, 2007.

[68] John F Bingert, Robert M Suter, Jonathan Lind, Shiu Fai Li, Reeju Pokharel, and Carl P Trujillo. High-energy diffraction microscopy characterization of spall damage. In *Dynamic Behavior of Materials, Volume 1*, pages 397–403. Springer, 2014.

[69] Leyun Wang, Meimei Li, Jonathan Almer, Thomas Bieler, and Rozaliya Barabash. Microstructural characterization of polycrystalline materials by synchrotron x-rays. *Frontiers of Materials Science*, 7(2):156–169, 2013.

[70] Reeju Pokharel, Jonathan Lind, Anand K Kanjarla, Ricardo A Lebensohn, Shiu Fai Li, Peter Kenesei, Robert M Suter, and Anthony D Rollett. Polycrystal plasticity: comparison between grain-scale observations of deformation and simulations. *Annu. Rev. Condens. Matter Phys.*, 5(1):317–346, 2014.

[71] David M Turner and Surya R Kalidindi. Statistical construction of 3-d microstructures from 2-d exemplars collected on oblique sections. *Acta Materialia*, 102:136–148, 2016.

[72] Ali Hasanabadi, Majid Baniassadi, Karen Abrinia, Masoud Safdari, and Hamid Garmestani. 3d microstructural reconstruction of heterogeneous materials from 2d cross sections: A modified phase-recovery algorithm. *Computational Materials Science*, 111:107–115, 2016.

[73] Jacques A Quiblier. A new three-dimensional modeling technique for studying porous media. *Journal of Colloid and Interface Science*, 98(1):84–102, 1984.

[74] PM Adler, Ch G Jacquin, and JA Quiblier. Flow in simulated porous media. *International Journal of Multiphase Flow*, 16(4):691–712, 1990.

[75] CLY Yeong and Salvatore Torquato. Reconstructing random media. *Physical review E*, 57(1):495, 1998.

[76] Bogdan Bochenek and Ryszard Pyrz. Reconstruction of random microstructures—-a stochastic optimization problem. *Computational Materials Science*, 31(1-2):93–112, 2004.

[77] Majid Baniassadi, Hamid Garmestani, DS Li, Said Ahzi, M Khaleel, and Xin Sun. Three-phase solid oxide fuel cell anode microstructure realization using two-point correlation functions. *Acta materialia*, 59(1):30–43, 2011.

[78] Mahesh RG Prasad, Napat Vajragupta, and Alexander Hartmaier. Kanapy: A python package for generating complex synthetic polycrystalline microstructures. *Journal of Open Source Software*, 4(43):1732, 2019.

[79] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[80] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[81] Zijiang Yang, Xiaolin Li, L Catherine Brinson, Alok N Choudhary, Wei Chen, and Ankit Agrawal. Microstructural materials design via deep adversarial learning methodology. *Journal of Mechanical Design*, 140(11), 2018.

[82] Ruijin Cang, Hechao Li, Hope Yao, Yang Jiao, and Yi Ren. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. *Computational Materials Science*, 150:212–221, 2018.

[83] Rahul Singh, Viraj Shah, Balaji Pokuri, Soumik Sarkar, Baskar Ganapathysubramanian, and Chinmay Hegde. Physics-aware deep generative models for creating synthetic microstructures. *arXiv preprint arXiv:1811.09669*, 2018.

[84] Akshay Iyer, Biswadip Dey, Arindam Dasgupta, Wei Chen, and Amit Chakraborty. A conditional generative model for predicting material microstructures from processing methods. *arXiv preprint arXiv:1910.02133*, 2019.

[85] Daria Fokina, Ekaterina Muravleva, George Ovchinnikov, and Ivan Oseledets. Microstructure synthesis using style-based generative adversarial networks. *Physical Review E*, 101(4):043308, 2020.

[86] Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E*, 96(4):043309, 2017.

[87] Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Stochastic reconstruction of an oolitic limestone by generative adversarial networks. *Transport in Porous Media*, 125(1):81–103, 2018.

[88] Andrea Gayon-Lombardo, Lukas Mosser, Nigel P Brandon, and Samuel J Cooper. Pores for thought: The use of generative adversarial networks for the stochastic reconstruction of 3d multi-phase electrode microstructures with periodic boundaries. *arXiv preprint arXiv:2003.11632*, 2020.

[89] Tim Hsu, William K Epting, Hokon Kim, Harry W Abernathy, Gregory A Hackett, Anthony D Rollett, Paul A Salvador, and Elizabeth A Holm. Microstructure generation via generative adversarial network for heterogeneous, topologically complex 3d materials. *JOM*, 73(1):90–102, 2021.

[90] Steve Kench and Samuel J Cooper. Generating 3d structures from a 2d slice with gan-based dimensionality expansion. *arXiv preprint arXiv:2102.07708*, 2021.

[91] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[92] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[93] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.

[94] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of data integration*. Elsevier, 2012.

[95] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[96] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[97] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[98] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[99] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491, 2015.

[100] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[101] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[102] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[103] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[104] Yipin Zhou and Tamara L Berg. Learning temporal transformations from time-lapse videos. In *European conference on computer vision*, pages 262–277. Springer, 2016.

[105] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016.

[106] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[107] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[108] Somnath Ghosh and Michael A Groeber. Developing virtual microstructures and statistically equivalent representative volume elements for polycrystalline materials. *Handbook of Materials Modeling: Methods: Theory and Modeling*, pages 1631–1656, 2020.

[109] Robert Gunn. *Duplex stainless steels: microstructure, properties and applications*. Woodhead Publishing, 1997.

[110] A Denquin and S Naka. Phase transformation mechanisms involved in two-phase tial-based alloys—i. lambellar structure formation. *Acta materialia*, 44(1):343–352, 1996.

[111] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[112] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Appendix A

Pipelines for data creation

A.1: Pipeline for single phase synthetic microstructure generation*.

Filter #	Filter name	Reason for Use
0	Stats Generator	constructs the arrays necessary to generate synthetic microstructures
1	Initialize Synthetic Volume	produces an empty volume as basis for inserting features in image geometry for creation of synthetic microstructures
2	Establish Shape Types	allocates shape type to each ensemble
3	Pack Primary Phases	pack precipitate features to achieve required statistics
4	Find Feature Neighbors	calculates number of neighbours of that feature
5	Match Crystallography	matches ODF to feature set
6	Find Feature Sizes	determines the size of each features
7	Delete Data	removes unwanted or unnecessary data
8	Create Element Array from Feature array	copies the data of a feature to all elements that belong to it
9	Export ASCII Data	outputs cell data array to a file in ASCII format
10	Export ASCII Data	outputs feature sizes array to a file in ASCII format
11	Export ASCII Data	outputs data containing volumes of features to a file in ASCII format
12	Write DREAM.3D Data file	Writes out all attribute matrices and pipeline to a file

A.2: Pipeline for removal of small features*.

Filter #	Filter name	Reason for Use
0	Read DREAM.3D Data File	reads all attribute matrices into a data structure
1	Threshold Objects	threshold data values based on given conditions
2	Remove Flagged Features	deletes features which are flagged by the threshold objects filter
3	Delete Data	removes unwanted or unnecessary data
4	Find Feature Sizes	determines the size of each features
5	Create Element Array from Feature Array	copies the data of a feature to all elements that belong to it
6	Export ASCII Data	outputs cell data array to a file in ASCII format
7	Export ASCII Data	outputs equivalent diameters to a file in ASCII format
8	Export ASCII Data	outputs feature volumes to a file in ASCII format
9	Write DREAM.3D Data File	writes out all the attribute matrices and pipeline to a file

A.3: Pipeline for generating two-phase 3D microstructures*.

Filter #	Filter name	Reason for Use
0	Create Data Container	generates a new empty data container
1	Create Geometry(Image)	creates geometry of image for constituting 3D rectilinear grid of voxels or pixels
2	Import ASCII Data	imports ASCII data from any text file into DREAM.3D format
3	Combine Attribute Arrays	merges specified attribute arrays into a single attribute array
4	Convert Orientation Representation	converts given representation of orientation to specified representation
5	Create Ensemble Info	stores info about the crystal structure and phase types of all the features
6	Create String Array	produces an attribute array of string type
7	Segment Features (Misorientation)	divides features by grouping neighbouring voxels such that misorientation tolerance is satisfied
8	Find Feature Neighbours	computes number of neighbours for that feature
9	Find Surface Features	governs whether a feature is present on the surface of the sample
10	Find Feature Centroids	computes centroid of each feature
11	Create Feature Array from Element Array	copies the associated element data to all the elements belonging to a feature
12	Delete Data	erases data which is not required or replica of existing data
13	Find Feature Sizes	computes the size of each feature
14	Find Feature Average Orientations	calculates the average orientation of each feature
15	Find Volume fractions of Ensembles	evaluates volume fraction of each ensemble
16	Export Pole Figure Images	outputs pole figure images for each ensemble
17	Export ASCII Data	outputs volume fractions in microstructure in ASCII format
18	Export ASCII Data	Outputs cell data in microstructure in ASCII format
19	Export ASCII Data	outputs diameters of features
20	Export ASCII Data	writes average of Euler angles to a file
21	Write DREAM.3D Data File	writes out all attribute matrices and pipeline to a file

A.4: Pipeline for generating two-phase 2D slices from 3D microstructures*.

Filter #	Filter name	Reason for Use
0	Read DREAM.3D Data File	reads all attribute matrices to a data structure
1	Crop Geometry(Image)	slice 3D microstructure into 2D based on the are
2	Create Ensemble Info	reads the information about crystal structure and phase types of all the features
3	Delete Data	deletes unwanted data
4	Threshold Objects	thresholding data values based on specified conditions
5	Segment Features (Misorientation)	divides features by grouping neighbouring voxels such that misorientation tolerance is satisfied
6	Find Feature Sizes	computes size of the features
7	Create Feature Array from Element Array	replicates the associated element data to all the elements belonging to a feature
8	Find Volume Fractions of Ensembles	evaluates volume fraction of each ensemble
9	Export ASCII Data	outputs pole figure image for each ensemble
10	Export ASCII Data	outputs volume fractions in microstructure in ASCII format
11	Export ASCII Data	outputs cell data in microstructure in ASCII format
12	Export ASCII Data	outputs diameters of features
13	Create String Array	produces an attribute array of string type
14	Export Pole Figure Images	outputs pole figure image for each ensemble
15	Write DREAM.3D Data File	writes out all attribute matrices and pipeline to a file

Appendix B

Postprocessing pipeline

A.1: Postprocessing pipeline.

Filter #	Filter name	Reason for Use
0	Create Data Container	generates a new empty data container
1	Create Geometry (Image)	creates geometry of image for constituting 3D rectilinear grid of voxels or pixels
2	Import ASCII Data	imports ASCII data from text file into DREAM. 3D format
3	Threshold Objects	permits user to define conditions for limiting Attribute arrays
4	Combine attribute Arrays	merges specified attribute arrays into a single attribute array
5	Convert Orientation Representation	converts given representation of orientation to specified representation
6	Create Ensemble Info	stores info about the crystal structure and phase types of all the features
7	Isolate Largest Feature (Identify Sample)	identifies the bad data which might be present due to overscanned volume
8	Neighbor Orientation Comparison (Bad Data)	orientations of bad cells are compared with their neighbors
9	Segment Features (Misorientation)	divides features by grouping neighbouring voxels such that misorientation tolerance is satisfied
10	Find Feature Phases	determines the ensemble of features
11	Find Feature Average Orientations	calculates the average orientation of each feature
12	Find Feature Neighbors	computes number of neighbours for that feature

13	Find Feature Sizes	computes the size of each feature
14	Delete Data	removes selected data from the prevailing data
15	Minimum Size	deletes features having a total number of cells below the given minimum threshold value
16	Find Feature Neighbors	computes number of neighbours for that feature
17	Fill Bad Data	eradicates noise in data by keeping only possible features
18	Erode/Dilate Bad Data	(with Erode) reduces the bad data by one cell for a given number of iterations
19	Erode/Dilate Bad Data	(with Dilate) expands the bad data by one cell for a given number of iterations
20	Find Feature Centroids	computes centroid of each feature
21	Delete Data	removes selected data from the prevailing data
22	Find Feature Sizes	computes the size of each feature
23	Create Element Array from Feature Array	duplicates the values of a feature to all the elements belonging to that feature
24	Find Feature Shapes	computes second-order moments of each feature to calculate principal axis lengths, principal axis directions, aspect ratios and moment invariant Omega3s
25	Export ASCII Data	outputs Feature data in microstructure in ASCII format
26	Generate IPF Colors	generate inverse pole figure (IPF) colors for crystal structures
27	Write DREAM.3D Data File	writes out all the attribute matrices and pipeline to a file
28	Export Feature Data as CSV File	writes the data associated with each feature to a CSV file