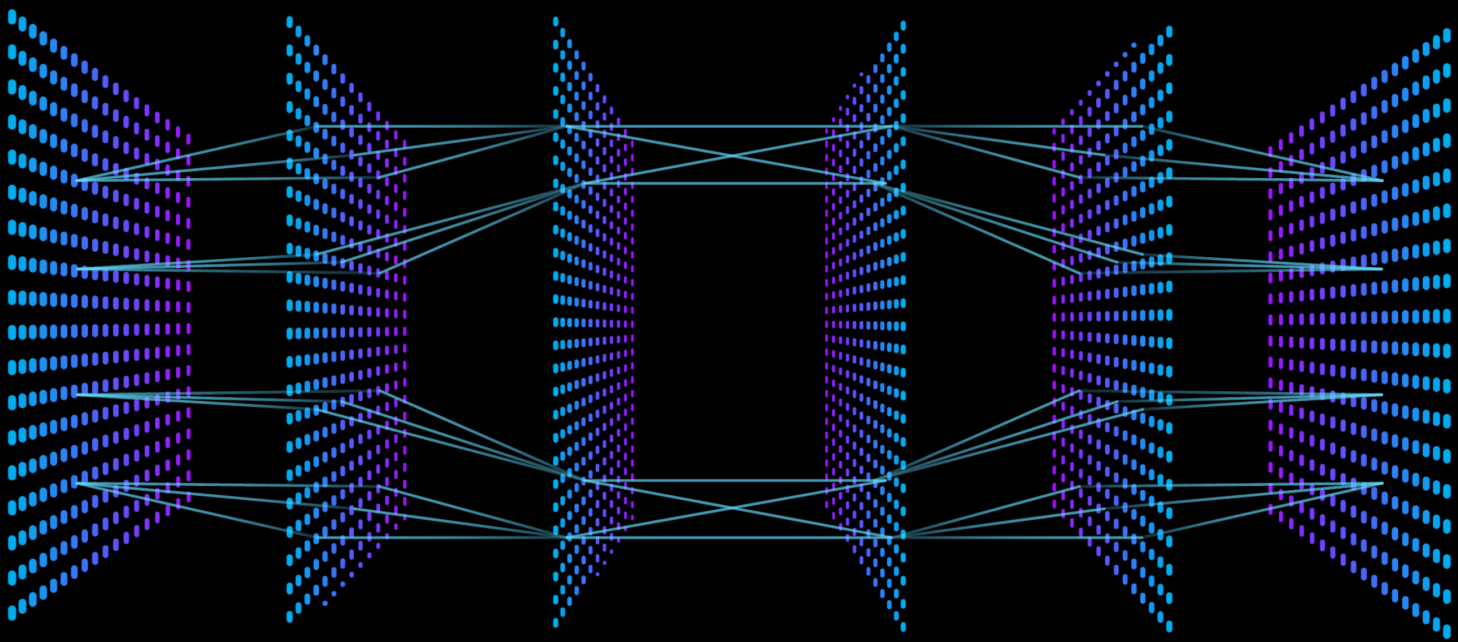
Master's Thesis

Offline stage acceleration of the self-consistent clustering analysis method: towards real-time material predictions

AE5711

Max S. Kukkola

Delft University of Technology



TU Delft

BROWN

Master's Thesis

Offline stage acceleration of the self-consistent clustering analysis method: towards real-time material predictions

by

Max S. Kukkola

4827104

in partial fullfilment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at Delft University of Technology
to be defended publicly on 31-05-2024 at 13:00

Graduation Committee:

Role	Name	Affiliation
Chair	Dr. Dimitrios Zarouchas	Delft University of Technology
Responsible supervisor	Dr. Baris Caglar	Delft University of Technology
Examiner	Dr. Bianca Giovanardi	Delft University of Technology
External supervisor	Dr. Miguel A. Bessa	Brown University

Supervisors:

Role	Name	Affiliation
Primary supervisor	Dr. Miguel A. Bessa	Brown University
Daily supervisor	Dr. Bernardo P. Ferreira	Brown University
Home supervisor	Dr. Baris Caglar	Delft University of Technology

Cover courtesy of CrowdAI [46]

TU Delft

BROWN

Preface

This master's thesis would not have been possible without the incredible support of so many people in my life whom I would like to thank.

First and foremost, I would like to thank Miguel and Bernardo for giving me the opportunity to come to Brown University and work under their supervision. Your methods for conducting research that combine curiosity, passion and rigor helped me throughout this endeavor and will be a guiding force in my professional journey from here on out. The experience at Brown, as well as our many conversations outside of work, also helped me develop a better understanding of the world. It was truly a fantastic experience and I am more than pleased to have been given the ability to pursue it.

I would also like to thank my family and friends for helping me finish this. My parents, who supported my decisions and gave me a high level of autonomy to decide what I wanted to do in life. My friends from Delft, as well as the many others I met around the world over the last few years, whom I've shared great moments with and who have gotten me through difficult times. I owe a great deal to all these people for shaping me into the person I have become. Also, a special shout-out to my friend Kübra for telling me about an "interesting machine learning course" I should take that eventually led me to meet Miguel for the first time.

With this experience drawing to an end, I feel fortunate to have been able to pursue what I love. I hope to continue working within the fields of mechanics and computational modeling in the future, and by doing so, I hope to further expand our understanding of the physical world.

Max S. Kukkola
Providence, May 2024

Abstract

Heterogeneous materials are vital for both the modern engineer and inquisitive scientist alike. They make up a vital material class that can either form inevitably as a result of material processing (such as crystallization in metals) or can be intentionally designed for to gain desirable properties (such as anisotropy in composites). As such, due to their prevalence and applicability to various engineering problems, predicting their behavior has become a topic of great interest in the solid mechanics community over the last few decades.

One particularly pressing challenge is performing fast mechanical simulations along multiple length scales in heterogeneous materials. To this end, the reduced order method known as Self-Consistent Clustering Analysis (SCA) has been proposed as an effective means of striking a balance between accuracy and efficiency. Underpinning this is SCA's ability to decompose a domain into clusters in a preliminary offline stage (learning), efficiently reducing the problem's degrees of freedom. It has been shown to be remarkably accurate in predicting plasticity without significantly degrading accuracy compared to other methods, such as the finite element method. However, the offline stage requires a quantity known as the Cluster Interaction Tensor (CIT) to be computed, whose computational complexity scales quadratically with the number of clusters, thus causing a bottleneck in the method. Additionally, the CIT is recomputed during the online stage (prediction) in the recently proposed extension called adaptive Self-consistent Clustering Analysis (ASCA), which further stresses the need to speed up their computation.

To address this limitation, a data-driven surrogate model is proposed to compute the CIT efficiently. The behavior of the local CIT components is first analyzed, from which it is concluded that a surrogate model shall be developed to predict upper off-diagonal terms. This decision is made due to the bi-modal nature of certain components in the tensor and the quadratic scaling behavior of off-diagonal terms versus the linear scaling of diagonal ones. Following that, a sensitivity study shows how various magnitudes of noise affect the homogenized response's accuracy and convergence performance. Furthermore, it is shown that the solution accuracy and convergence behavior are degraded when the number of clusters is increased. With a proper understanding of the CIT's behavior and its function within SCA, the surrogate can be created.

The surrogate's feasibility is first shown using the ResNet-18 architecture, a CNN model derived from computer vision. It is demonstrated that ResNet-18 can make accurate predictions for a range of different microstructural parameters. This includes the number of clusters and samples in the dataset's distribution and out of distribution. Composite RVEs are used as out-of-distribution samples to test the robustness of the surrogate to realistic inputs. An attempt is made to improve the model's performance by training it on an unbalanced dataset, and although it improves the overall CIT prediction in specific regimes, the resulting stress-strain generalizability is degraded. Subsequently, a lighter version of ResNet-18, known as ResNet-lite, is tested and shown to give faster predictions than the baseline method. This, however, comes at a cost to the accuracy of the solution.

Additionally, deemed as a critical aspect of the study, the efficient generation of a large representative training dataset is discussed. A novel method for generating clustered microstructures efficiently using gradient noise is introduced. By leveraging datasets derived using this method, the surrogate can learn the fundamental interactions needed to make accurate predictions on realistic, out-of-distribution samples. The need for a large dataset is further emphasized using a dataset size sensitivity study.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
ASCA	Adaptive Self-consistent Clustering Analysis
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CIT	Cluster Interaction Tensor
CROM	Cluster-based Reduced Order Model
CRVE	Cluster-reduced Representative Volume Element
DNS	Direct Numerical Simulation
FEM	Finite Element Method
FFT	Fast Fourier Transform
FLOPs	Floating Point Operations
GPU	Graphics Processing Unit
ML	Machine Learning
PINN	Physics Informed Neural Network
POD	Proper Orthogonal Decomposition
ROM	Reduced Order Model
RVE	Representative Volume Element
SCA	Self-consistent Clustering Analysis
SCS	Self Consistent Scheme
TCL	Tensor Contraction Layer
TRL	Tensor Regression Layer

1

Introduction

Heterogeneous materials form a highly relevant material subset used and analyzed by engineers and scientists today. For some material classes, heterogeneities form inevitably as a result of material processing (such as crystallization in metals), whilst in others the heterogeneities can be added on purpose to gain desirable properties (such as anisotropy in composites). Due to their prevalence and applicability to various engineering problems, predicting their behavior has become a topic of great interest in the solid mechanics community over the last few decades.

Materials are said to act in a hierarchical manner, where simple small-scale constituents interact in a reciprocal manner with the complex large-scale compounds they make up. Several models have been suggested to take into account this hierarchical nature in heterogeneous materials, with many aiming to characterize the macro-scale due to its applicability to various engineering problems.

Traditional phenomenological models for example treat the material constituents as one and infer an average behavior at the macro-scale. These methods have been shown to be viable in many applications yet fall short in others due to the need for laborious experimentation to determine and calibrate model parameters.

Concurrent multi-scale methods, on the other hand, have found solutions by establishing a direct link between the macro and micro scales. In this approach, each macroscopic point is evaluated using a high-fidelity microscopic simulation of a Representative Volume Element (RVE). However, performing this analysis using direct numerical simulations can be intractable for many engineering problems due to the computational requirements imposed by the dimensionality of the problem.

As a consequence of the inefficiency of direct numerical simulations in concurrent multi-scale methods, Reduced Order Models (ROM) have been introduced to reduce the dimensionality of the microscopic simulation. They have the promise of providing a two-for-one solution: making accurate predictions based on the constitutive behavior of each separate constituent whilst being computationally efficient.

This thesis project therefore aims to disseminate the state of the art in ROM research and suggest improvements that can help make accurate and efficient predictions of heterogeneous materials. Specifically, by combining already efficient ROM's with a data-driven surrogate modeling approach, which has also shown tremendous potential at making computational methods more efficient, the resulting approach has the potential to be significantly faster than conventional methods whilst maintaining a high level of accuracy. The proposed method could be used by designers to make accurate yet fast predictions with limited reliance on experimental data. Some potential use cases could be multiscale design using topology optimization and material discovery.

2

Literature Review

This chapter presents the literature review conducted to determine a number of knowledge gaps in the field of solid mechanics and introduces a research question for the thesis project. First reduced order modeling and specifically self-consistent clustering analysis are introduced in section 2.1. Then, machine learning methods and their applications in a number of fields are highlighted in section 2.2. Following that, in section 2.3, methodologies for obtaining datasets of clustered microstructures are explored. Next, different metrics for evaluating algorithms are discussed in section 2.4. Finally, in section 2.5, based on the reviewed literature, knowledge gaps are stated and a research question is formulated.

2.1. Reduced order modeling

A variety of Reduced Order Models (ROMs) have been introduced in the continual search to obtain computationally cheap yet accurate mechanical solutions of heterogeneous materials. Arguably the most used are the analytical homogenization methods which have been applied for decades, most notably the Mori-Tanaka method [39] and self-consistent scheme [22, 8]. Additionally, lower and upper material property bounds have been estimated using the Voigt-Reuss [57, 47] and Hashin-Shtrikman bounds [20]. However, underpinning these methods are the assumptions of mean-field and linear superposition which limit their applicability to more complex structures or localized nonlinear material behavior such as plasticity [15].

One particular class of ROMs called the Cluster-based Reduced Order Model (CROM) has gained significant traction over the last decade. First presented by Liu et al. [37] as the self-consistent clustering analysis (SCA), it can estimate solutions at a significantly lower cost than direct numerical simulations (decreasing the wall clock time in orders of magnitude) without incurring a significant reduction in solution accuracy. It has been shown to be particularly well suited for modeling plasticity in heterogeneous materials. The method was further extended and improved by Ferreira et al. [12] to include adaptivity. This section gives a high-level overview of SCA and presents some of its limitations.

2.1.1. Self-Consistent Clustering Analysis

The Representative Volume Element (RVE) is the foundation of many multi-scale methods. However, modeling a high-fidelity RVE can often be computationally intractable due to the fine mesh needed to accurately capture the geometry and mechanical behavior at small length scales. To counter this, one may attempt to decompose the microstructure into larger domains called material clusters. However, when applying the same analysis methods used for the high-fidelity RVE, such as finite element analysis, greater limitations are imposed on the mesh and the solutions accuracy is diminished. Given these challenges, SCA emerges as a notably effective approach due to its ability to model large and irregular material clusters whilst maintaining high solution accuracy.

A high level summary of SCA is provided in Figure 2.1. The method consists of two primary stages: an online stage and an offline stage. The offline stage serves to compress an RVE into a lower dimensional representation known as a Cluster-reduced RVE (CRVE). It does this by grouping material points with similar properties (step 2), with the properties being derived from a set of cheap linear elastic

Direct Numerical Simulation (DNS) in step 1. Using the clustered representation of the microstructure the Cluster Interaction Tensor (CIT) is computed (step 3) to give a stress-strain relationship between every cluster pair. With the CRVE defined, the online stage of the method can commence in which the material response is found given a set of strain and/or stress macro-scale loading constraints. It does so by solving the clustered Lippmann-Schwinger system of equilibrium equations using the Newton-Raphson scheme (step 4). Finally, the macro-scale material mechanical response can be computed, using computational homogenization (step 5).

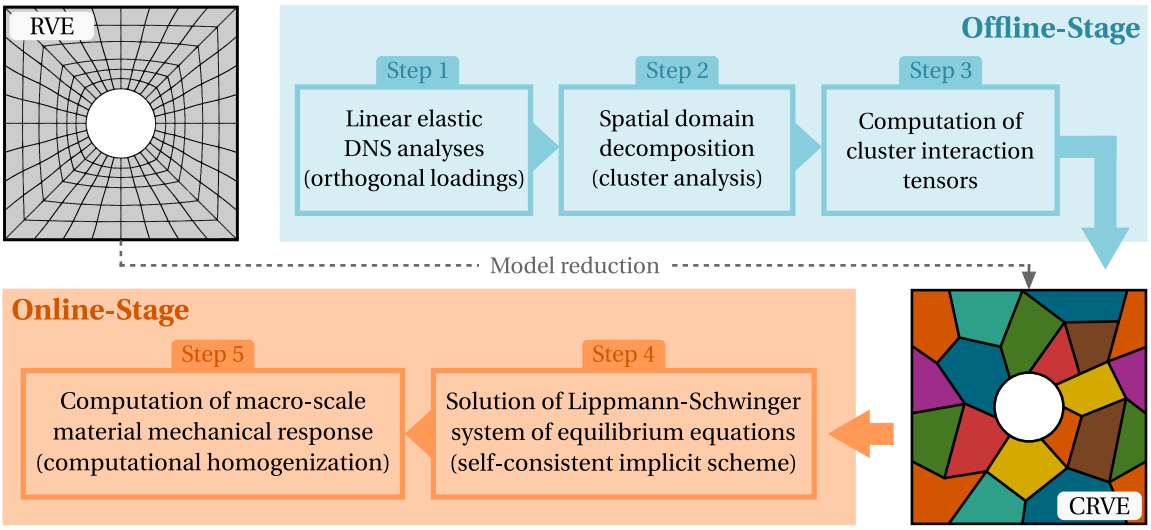


Figure 2.1: Summary of the Self-Consistent Clustering Analysis (SCA) (taken from Ferreira [15]).

2.1.2. Clustering

SCA owes its computational efficiency to its ability to approximate the behavior of an RVE using a finite, lower-dimensional set of material clusters. This CRVE representation relies on a piecewise uniform approximation to a local field of interest (Figure 2.2). Namely, the field of choice is assumed constant within each material cluster. Denoting $\boldsymbol{a}_\mu$ to be a generic field, this assumption can be expressed as

$$\boldsymbol{a}_\mu(\boldsymbol{Y}) = \sum_{I=1}^{n_c} \boldsymbol{a}_\mu^{(I)} \chi^{(I)}(\boldsymbol{Y}), \qquad \chi^{(I)}(\boldsymbol{Y}) = \begin{cases} 1 & \text{if } \boldsymbol{Y} \in \Omega_{\mu,0}^{(I)} \\ 0 & \text{otherwise} \end{cases}, \tag{2.1}$$

where $\boldsymbol{a}_\mu$ is the homogeneous field in the Ith material cluster and $\chi^{(I)}$ is the characteristic function of the Ith material cluster. Based on this definition, the following equivalence can be established:

$$\int_{\Omega_{\mu,0}} \chi^{(I)}(\boldsymbol{Y}) \, [\bullet]_\mu \, \mathrm{d}v = \int_{\Omega_{\mu,0}^{(I)}} [\bullet]_\mu \, \mathrm{d}v. \tag{2.2}$$

Under such assumptions, a system of equations may be established based on the Lippmann-Schwinger equation (originally introduced in the context of Quantum Mechanical Scattering Theory but later adopted for the case of solid mechanics).

The decomposition of the RVE into a CRVE is performed using a material point similarity principle. Namely, the strain concentration tensor $\mathbf{H}^e(\boldsymbol{Y})$ is used as a heuristic to group similar points together.

$$\varepsilon_\mu^e(\boldsymbol{Y}) = \mathbf{H}^e(\boldsymbol{Y}) : \varepsilon^e(\boldsymbol{X}), \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}. \tag{2.3}$$

The strain concentration tensor can be determined by performing a DNS of 3 or 6 linear elastic micro-scale equilibrium problems under orthogonal loading conditions for the case of 2D or 3D, respectively. This heuristic is chosen firstly because the mechanical behavior under any loading condition within the elastic regime is identical for points with the same strain concentration. Furthermore, these points act similarly to each other into the nonlinear plastic regime due to the fact that plastic flow generally occurs in areas of high strain concentration.

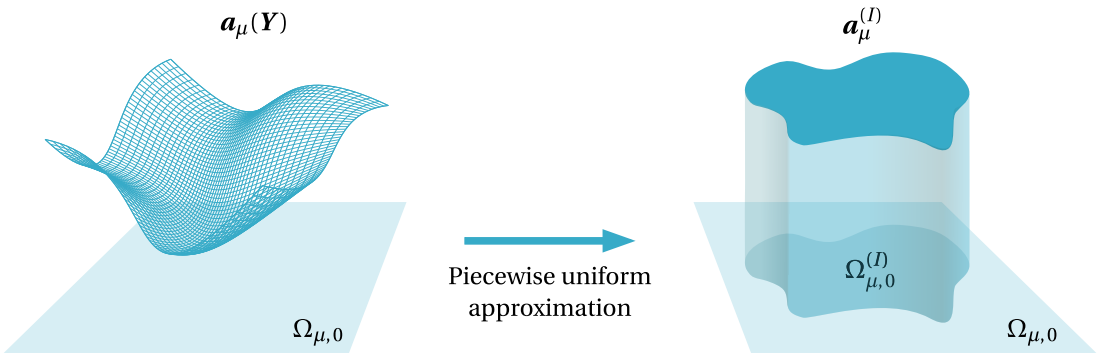


Figure 2.2: Schematic of a material cluster, $\Omega_\mu^{(I)}$, within the micro-scale domain, $\Omega_{\mu,0}$, and the associated uniform assumption for a generic field $\boldsymbol{a}_\mu(\boldsymbol{Y})$ (taken from Ferreira [15]).

Once the strain concentration tensor is found the points are clustered using K-means clustering. It should be noted that using this scheme the points do not need to be adjacent to each other in order to be clustered together. As a consequence, clusters can be discontinuous. An example of CRVE's of two microstructures is shown in Figure 2.3.

2.1.3. Clustered Lippmann-Schwinger homogenization

To solve for the material mechanical response given a CRVE, the clustered Lippmann-Schwinger homogenization needs to be introduced. The following derivation is performed assuming infinitesimal strains. A derivation for the case of finite strains is provided by Ferreira [15], although is considered outside of the scope of the present work.

Consider a heterogeneous RVE of domain $\Omega_{\mu,0}$ and boundary $\partial\Omega_{\mu,0}$ composed several perfectly bonded, linear elastic phases. Assuming periodic boundary conditions, a macro-scale strain $\varepsilon(\boldsymbol{X})$ can be prescribed using

$$\varepsilon(\boldsymbol{X}) = \frac{1}{v_{\mu,0}} \int_{\Omega_{\mu,0}} \varepsilon_\mu(\boldsymbol{Y})\, \mathrm{d}v, \tag{2.4}$$

where $v_{\mu,0}$ is the volume of the RVE. Additionally, the micro-scale strain field can be decomposed into separate components,

$$\varepsilon_\mu(\boldsymbol{Y}) = \varepsilon(\boldsymbol{X}) + \tilde{\varepsilon}_\mu(\boldsymbol{Y}), \tag{2.5}$$

where $\varepsilon(\boldsymbol{X})$ is the average (far field) strain and $\tilde{\varepsilon}_\mu(\boldsymbol{Y})$ is the fluctuation component.

In the absence of body forces, the micro-scale quasi-static equilibrium condition can be expressed as:

$$\mathrm{div}_0\left[\boldsymbol{\sigma}_\mu(\boldsymbol{Y})\right] = \boldsymbol{0} \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}. \tag{2.6}$$

At this step, it is convenient to introduce a reference (fictitious) homogeneous linear elastic material with tangent modulus $\mathbf{D}^{e,0}$. This allows the real stress to be separated into two parts,

$$\boldsymbol{\sigma}_\mu(\boldsymbol{Y}) = \mathbf{D}^{e,0}(\boldsymbol{Y}) : \tilde{\varepsilon}_\mu(\boldsymbol{Y}) + \boldsymbol{\sigma}_\mu^*(\boldsymbol{Y}), \tag{2.7}$$

where $\boldsymbol{\sigma}_\mu^*$ is the eigenstress, which represents the difference in the stress between the actual heterogeneous material and the reference homogeneous material for a given strain.

Using the Green operator $\boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}')$ which corresponds to the strain contributed by a concentrated external stress at $\boldsymbol{Y}'$ in the reference material, the equilibrium condition can be rewritten in integral form:

$$\varepsilon_\mu(\boldsymbol{Y}) = -\int_{\Omega_{\mu,0}} \boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}') : \varepsilon_\mu^*(\boldsymbol{Y}')\, \mathrm{d}v' + \varepsilon_\mu^0, \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}. \tag{2.8}$$

Substituting Equation 2.7 into Equation 2.8,

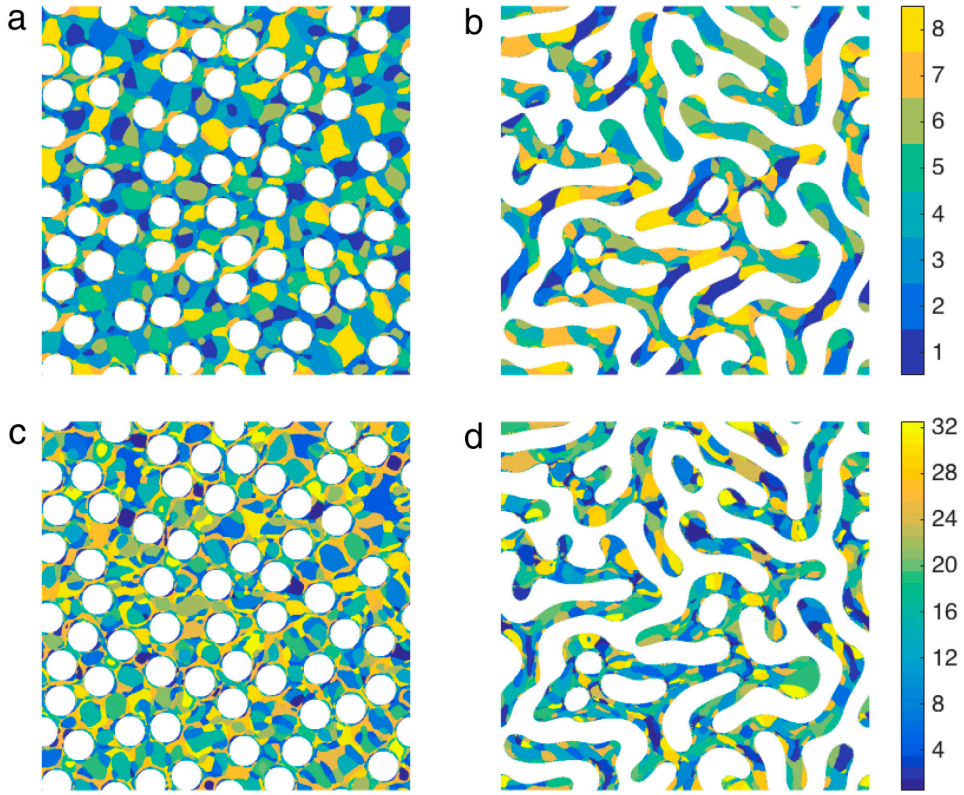


Figure 2.3: Examples demonstrating CRVE's for a 2D simulation. a and c show the cross section of a fiber reinforced composite and b and d show a two phase-field simulation based on the Cahn–Hilliard equation. In a and b the first material phase is discretized into 8 clusters and in c and d into 32 clusters. The second material phase is removed for clarity. Each cluster color corresponds to an individual cluster. Note how clusters can be discontinuous (taken from Liu et al. [37]).

$$\boldsymbol{\varepsilon}_\mu(\boldsymbol{Y}) = -\int_{\Omega_{\mu,0}} \boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}') : \left(\boldsymbol{\sigma}_\mu(\boldsymbol{Y}') - \mathbf{D}^{e,0} : \boldsymbol{\varepsilon}_\mu(\boldsymbol{Y}')\right) \mathrm{d}v' + \boldsymbol{\varepsilon}^0_\mu\,, \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}. \tag{2.9}$$

To solve the equation, boundary conditions from the macroscopic scale must be imposed. This can be done using a strain constraint

$$\frac{1}{v_\mu}\int_{\Omega_{\mu,0}} \boldsymbol{\varepsilon}_\mu(\boldsymbol{Y})\,\mathrm{d}v = \boldsymbol{\varepsilon}(\boldsymbol{X})\,, \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}, \tag{2.10}$$

a stress constraint

$$\frac{1}{v_\mu}\int_{\Omega_{\mu,0}} \boldsymbol{\sigma}_\mu(\boldsymbol{Y})\,\mathrm{d}v = \boldsymbol{\sigma}(\boldsymbol{X})\,, \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}, \tag{2.11}$$

or alternatively as a mixed stress/strain constraint. For numerical applications, the equation can be rewritten in incremental time form

$$\boldsymbol{\varepsilon}_{\mu,m+1}(\boldsymbol{Y}) = -\int_{\Omega_{\mu,0}} \boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}') : \left(\boldsymbol{\sigma}_{\mu,m+1}(\boldsymbol{Y}') - \mathbf{D}^{e,0} : \boldsymbol{\varepsilon}_{\mu,m+1}(\boldsymbol{Y}')\right) \mathrm{d}v' + \boldsymbol{\varepsilon}^0_{\mu,m+1}\,, \quad \forall \boldsymbol{Y} \in \Omega_{\mu,0}. \tag{2.12}$$

This is the Lippmann-Schwinger equation in incremental form and solving it for every material point would be computationally expensive. However using the previously defined assumption given by Equation 2.1 the domain can be decomposed into a finite set of material clusters. In order to make the method efficient, the decomposition requires that the number of material clusters in the CRVE be significantly smaller than the number of material points modeled in the high fidelity RVE. The Lippmann-Schwinger equation can therefore be averaged for each cluster

$$\frac{1}{f^{(I)}v_\mu}\int_{\Omega_{\mu,\,0}}\chi^{(I)}(\boldsymbol{Y})\boldsymbol{\varepsilon}_{\mu,\,m+1}(\boldsymbol{Y})\,\mathrm{d}v=-\frac{1}{f^{(I)}v_\mu}\int_{\Omega_{\mu,\,0}}\left[\int_{\Omega_{\mu,\,0}}\chi^{(I)}(\boldsymbol{Y})\,\boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}'):\right.$$
$$\left.\Big(\boldsymbol{\sigma}_{\mu,\,m+1}(\boldsymbol{Y}')-\mathbf{D}^{e,\,0}:\boldsymbol{\varepsilon}_{\mu,\,m+1}(\boldsymbol{Y}')\Big)\,\mathrm{d}v'\right]\mathrm{d}v+\boldsymbol{\varepsilon}^0_{\mu,\,m+1}\,,\quad I=1,2,\ldots,n_{\mathsf{c}}\,, \tag{2.13}$$

where $f^{(I)}$ is the volume fraction of the cluster and v_μ is the total volume of the CRVE. The cluster piece wise assumption yields

$$\boldsymbol{\varepsilon}_{\mu,\,m+1}(\boldsymbol{Y}')=\sum_{J=1}^{n_{\mathsf{c}}}\chi^{(J)}(\boldsymbol{Y}')\,\boldsymbol{\varepsilon}^{(J)}_{\mu,\,m+1}\,, \tag{2.14}$$

$$\boldsymbol{\sigma}_{\mu,\,m+1}(\boldsymbol{Y}')=\sum_{J=1}^{n_{\mathsf{c}}}\chi^{(J)}(\boldsymbol{Y}')\,\boldsymbol{\sigma}^{(J)}_{\mu,\,m+1}\,, \tag{2.15}$$

using which Equation 2.13 can be simplified to:

$$\boldsymbol{\varepsilon}^{(I)}_{\mu,\,m+1}=-\sum_{J=1}^{n_{\mathsf{c}}}\left(\frac{1}{f^{(I)}v_\mu}\int_{\Omega_{\mu,\,0}}\int_{\Omega_{\mu,\,0}}\chi^{(I)}(\boldsymbol{Y})\chi^{(J)}(\boldsymbol{Y}')\,\boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}')\,\mathrm{d}v'\,\mathrm{d}v\right):$$
$$\Big(\hat{\boldsymbol{\sigma}}^{(J)}_{\mu,\,m+1}-\mathbf{D}^{e,\,0}:\boldsymbol{\varepsilon}^{(J)}_{\mu,\,m+1}\Big)+\boldsymbol{\varepsilon}^0_{\mu,\,m+1}\,,\quad I=1,2,\ldots,n_{\mathsf{c}}\,. \tag{2.16}$$

This can be expressed in a more compact way as:

$$\boldsymbol{\varepsilon}^{(I)}_{\mu,\,m+1}=-\sum_{J=1}^{n_{\mathsf{c}}}\mathbf{T}^{(I)(J)}:\Big(\boldsymbol{\sigma}^{(J)}_{\mu,\,m+1}-\mathbf{D}^{e,\,0}:\boldsymbol{\varepsilon}^{(J)}_{\mu,\,m+1}\Big)+\boldsymbol{\varepsilon}^0_{\mu,\,m+1}\,,\quad I=1,2,\ldots,n_{\mathsf{c}}\,, \tag{2.17}$$

where $\mathbf{T}^{(I)(J)}$ is the cluster interaction tensor

$$\mathbf{T}^{(I)(J)}=\frac{1}{f^{(I)}v_\mu}\int_{\Omega_{\mu,\,0}}\int_{\Omega_{\mu,\,0}}\chi^{(I)}(\boldsymbol{Y})\,\chi^{(J)}(\boldsymbol{Y}')\,\boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}')\,\mathrm{d}v'\mathrm{d}v\,,\quad I,J=1,2,\ldots,n_{\mathsf{c}}\,. \tag{2.18}$$

Applying Equation 2.2 to Equation 2.14 and Equation 2.15 the constraints can be reformulated as follows:

$$\sum_{I=1}^{n_{\mathsf{c}}}f^{(I)}\boldsymbol{\varepsilon}^{(I)}_{\mu,\,m+1}=\boldsymbol{\varepsilon}_{m+1}(\boldsymbol{X})\,, \tag{2.19}$$

$$\sum_{I=1}^{n_{\mathsf{c}}}f^{(I)}\boldsymbol{\sigma}^{(I)}_{\mu,\,m+1}=\boldsymbol{\sigma}_{m+1}(\boldsymbol{X})\,. \tag{2.20}$$

2.1.4. Cluster Interaction Tensor

An important step following the clustering of the RVE and before the material's mechanical response is solved, is the computation of the CIT (in the offline stage). The CIT physically represents the influence of the stress in the Jth cluster on the strain in the Ith cluster. The tensor contains components for every combination of cluster pairs and the clusters can be visualized as images as seen in Figure 2.4.

Using Equation 2.18 it can be observed that the CIT exhibits a cluster symmetry along one of its diagonals

$$\mathbf{T}^{(J)(I)}=\frac{f^{(I)}}{f^{(J)}}\,\mathbf{T}^{(I)(J)}. \tag{2.21}$$

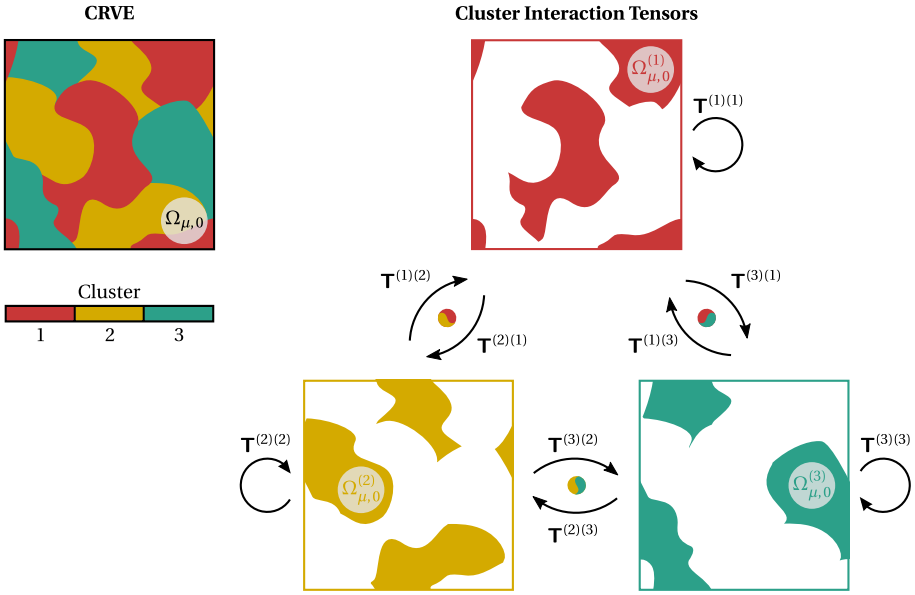


Figure 2.4: Visualization of the cluster interaction tensor (CIT) components in a cluster-reduced RVE (CRVE) with 3 material clusters (taken from Ferreira [15]).

It is assumed that the reference material is linear elastic which enables the Green operator to take a closed-form expression in the frequency domain given by:

$$\breve{\Phi}^{0}_{klij}(\boldsymbol{\zeta}) = \frac{1}{4\mu^0}\frac{\delta_{ki}\zeta_j\zeta_l + \delta_{kj}\zeta_i\zeta_l + \delta_{li}\zeta_j\zeta_k + \delta_{lj}\zeta_i\zeta_k}{|\boldsymbol{\zeta}|^2} - \frac{\lambda^0+\mu^0}{\mu^0(\lambda^0+2\mu^0)}\frac{\zeta_k\zeta_l\zeta_i\zeta_j}{|\boldsymbol{\zeta}|^4}, \tag{2.22}$$

where $\boldsymbol{\zeta}$ is the frequency wave vector corresponding to $\boldsymbol{Y}$ in real space, δ_{ij} is the Kronecker delta, μ^0 and λ^0 are the Lamé constants of the reference material. This can be conveniently split up

$$\breve{\boldsymbol{\Phi}}^{0}(\boldsymbol{\zeta}) = c_1(\lambda^0,\mu^0)\ \breve{\boldsymbol{\Phi}}^{0}_1(\boldsymbol{\zeta}) + c_2(\lambda^0,\mu^0)\ \breve{\boldsymbol{\Phi}}^{0}_2(\boldsymbol{\zeta}). \tag{2.23}$$

where the reference material independent components are defined as

$$(\breve{\Phi}^{0}_1)_{klij}(\boldsymbol{\zeta}) = \frac{1}{|\boldsymbol{\zeta}|^2}\left(\delta_{ki}\zeta_j\zeta_l + \delta_{kj}\zeta_i\zeta_l + \delta_{li}\zeta_j\zeta_k + \delta_{lj}\zeta_i\zeta_k\right), \quad (\breve{\Phi}^{0}_2)_{klij}(\boldsymbol{\zeta}) = \frac{\zeta_k\zeta_l\zeta_i\zeta_j}{|\boldsymbol{\zeta}|^4}, \tag{2.24}$$

with

$$c_1(\lambda^0,\mu^0) = \frac{1}{4\mu^0}, \quad c_2(\lambda^0,\mu^0) = -\frac{\lambda^0+\mu^0}{\mu^0(\lambda^0+2\mu^0)}. \tag{2.25}$$

In this form, the expensive computation of the fourth-order Green operator components $\breve{\boldsymbol{\Phi}}^{0}_1(\boldsymbol{\zeta})$ and $\breve{\boldsymbol{\Phi}}^{0}_2(\boldsymbol{\zeta})$ are carried out only once in the offline stage and later linearly scaled by the coefficients $c_1(\lambda^0,\mu^0)$ and $c_2(\lambda^0,\mu^0)$ during the online stage. In addition, due to the singularity at the zero frequency, a third Green operator component is conveniently introduced by enforcing the condition $\breve{\boldsymbol{\Phi}}^{0}(\boldsymbol{\zeta}=\mathbf{0}) = \mathbf{0}$.

Furthermore the convolution of cluster $\chi^{(J)}$ with the Green operator $\boldsymbol{\Phi}^0$ can be expedited by performing it in the frequency domain using the Fast Fourier Transform (FFT):

$$\int_{\Omega_{\mu,0}} \chi^{(J)}(\boldsymbol{Y}')\,\boldsymbol{\Phi}^0(\boldsymbol{Y}-\boldsymbol{Y}')\,\mathrm{d}v' = \mathscr{F}^{-1}\left(\breve{\chi}^{(J)}(\boldsymbol{\zeta})\,\breve{\boldsymbol{\Phi}}^{0}(\boldsymbol{\zeta})\right). \tag{2.26}$$

The computation of the CIT has been identified as a significant bottleneck in the offline stage of the method [15, 59] and in the online stage when adaptivity is used in the Adaptive Self-consistent Clustering Analysis (ASCA) [15]. Since the computation is permutational in nature, it balloons with a complexity of $\mathcal{O}(n_c^2)$, where n_c is the number of clusters. This observation is supported by Table 2.1

which presents Ferriera's [15] reported results on the partial and total computational times in SCA and ASCA for a composite RVE benchmark under uniaxial tension. Furthermore, the offline stage results are visualized in Figure 2.5 where the bottleneck in the computation of the CIT (step 3), caused by the scaling performance, can be observed.

	Computational time (s)					
Method	**Offline (step 1)**	**Offline (step 2)**	**Offline (step 3)**	**Online (solution)**	**Online (adapt.)**	**Total**
DNS	-	-	-	23800	-	23800
SCA$^{(c)}$	219	22	52	132	-	425
SCA$^{(m)}$	219	108	399	1350	-	2080
SCA$^{(f)}$	219	154	1100	2460	-	3930
ASCA$^{(*)}$	219	22	52	324	637	1010

Table 2.1: Partial and total computational times (s) of the different solution methods associated with a composite RVE benchmark under uniaxial tension. Number of clusters: (c) $n_c = 65$ (coarse), (m) $n_c = 185$ (medium), (f) $n_c = 305$ (fine) and (*) $n_c = 65 \rightarrow 190$. Offline stage: linear elastic DNS simulations (step 1), base cluster analysis (step 2) and computation of cluster-interaction tensors (step 3). [15]

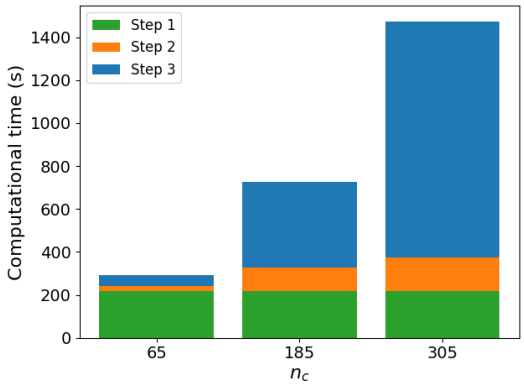


Figure 2.5: Partial computational times (s) of the offline stage in SCA simulations with a composite RVE benchmark under uniaxial tension. Offline stage: linear elastic DNS simulations (step 1), base cluster analysis (step 2) and computation of cluster-interaction tensors (step 3). [15]

A fast algorithm for computing the interaction tensor was proposed by Zhang et al. [59]. The method maps points from the original mesh to a courser mesh using composition ratios to reduce the number of computations needed. Their method decreases the required number of operations yet still scales with quadratic complexity when the number of clusters is increased.

The online stage as well as other aspects of the method are considered outside of the scope of this work. The interested reader is directed to Liu et al. [37] and Ferreira [15] for a more detailed analysis of the physics and algorithmic implementation of SCA.

2.2. Machine Learning

Machine learning (ML) is the artificial intelligence technique in which computers can learn by drawing inferences from patterns in data without following predetermined and explicit instructions. Although it has existed in some form since the 1950s, it has exploded in popularity over the last few years thanks to increased datasets, improvements in algorithms and advances in computer hardware [4].

Whilst ML has been used to solve various problems in fields ranging from language, medicine and finance, to name a few, it would be prudent to limit this literature review to only the most relevant fields. Since the information about the clusters is spatial in nature, the field of computer vision is first discussed. Next, examples of the use of machine learning in the solid mechanics field is highlighted to draw parallels to the topic of research in this manuscript. Finally, the framework of operator learning is introduced which is a useful generalization of mappings between function spaces.

2.2.1. Machine learning in computer vision

Computer vision is a mature field that provides many tools for performing classification or regression tasks on image-like data.

One of the major breakthroughs in the field was the paradigm shift from traditional ML to deep learning [2]. Namely, in traditional ML a successful architecture was in large part determined by the method used to extract features. Many were proposed such as histogram of oriented gradients, scale invariant feature transform, and bag of words. Deep learning on the other hand offered a black box solution for extracting and operating on features in an image without any feature engineering (Figure 2.6). This together with the following characteristics makes deep learning particularly appealing:

1. Universal learning approach: deep learning can provide universal approximations in most application domains.
2. Robustness: since features don't need to be precisely defined, deep learning strategies are more robust to changes in the input data since the new features can be learned in an automated manner.
3. Generalization: the same architectures can be used in different applications which is referred to as transfer learning. This is particularly useful in situations when there is little data in an application of interest but lots of data in a similar application.
4. Scalability: due to the ease of parallelising deep learning algorithms and advances in CPU and GPU technologies, they can be scaled to include large numbers of hidden layers.

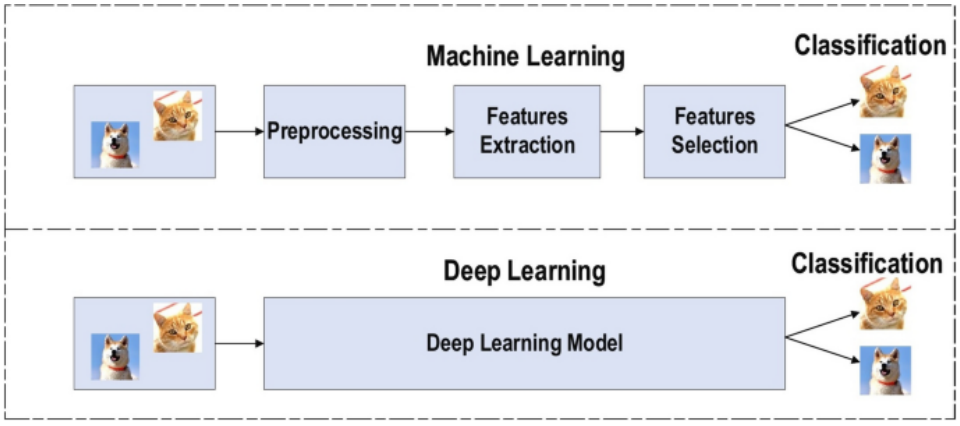


Figure 2.6: The difference between deep learning and traditional machine learning (taken from Alzubaidi et al. [2]).

The most notable limitation of deep learning, however, is its reliance on large amounts of data due to the increased number of parameters that need to be trained. Therefore, it can be said that traditional methods can often offer better performance for smaller datasets whereas deep learning performs better for larger datasets (Figure 2.7).

Arguably the most important discovery in the field of computer vision was the convolutional neural network (CNN). First introduced by Fukushima [17] and later popularized by LeCun et al. [34] it was able to greatly outperform other state-of-the-art architectures at the time. CNNs have allowed for greater feature extraction and have been successfully applied within deep learning frameworks. The strengths of CNNs over other networks, such as fully connected networks, can be attributed to four main concepts: local connections, weight sharing, pooling and multiple layers [3]. CNNs employ convolutional layers that convolve a kernel over a feature space. These kernels are typically smaller than the feature space, allowing them to capture local features within the kernels receptive field. Since these kernel's are small and convolved over an entire feature space, it allows a small number of weights to be shared over the entire feature space. Following a convolutional layer and, typically, a nonlinear activation, the outputs can be pooled to downsample the representation and, as such, reduce the number of computations and weights required in downstream layers. Finally, by stacking multiple convolutional and pooling layers in sequence the network can learn a hierarchical representation of an image at multiple scales.

The ImageNet Large Scale Visual Recognition Challenge [49] has been one of the major drivers of innovation and standardization within the computer vision field. Notably, the object localization task let researchers from around the world train and compare their architectures every year on training

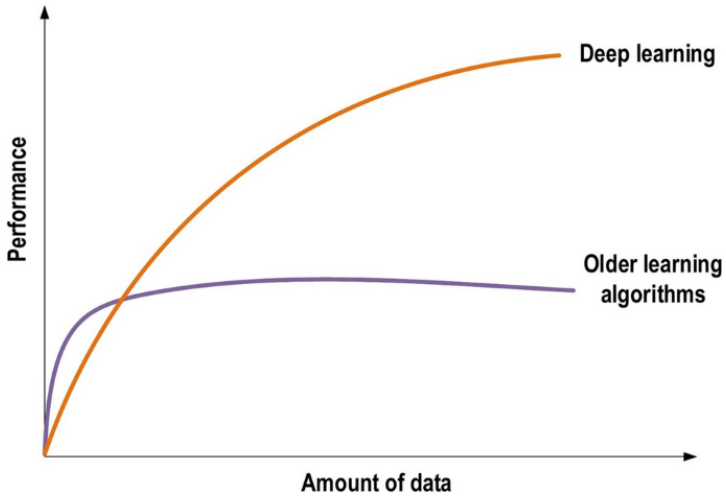


Figure 2.7: Comparison of deep learning and traditional machine learning as a function of dataset size (taken from Alzubaidi et al. [2]).

and validation sets of 1.2 million and 150k images, respectively. The first major breakthrough came with AlexNet which not only improved the accuracy by a wide margin but also showed that CNNs can be efficiently parallelised on a GPU to speed up the training of large networks [31]. The next major breakthrough came with VGGNet which utilized a relatively simple architecture but demonstrated that accuracy can be significantly increased by increasing model depth [53]. Following that, ResNet introduced the concept of residual learning which allowed information to bypass a convolutional layer using an identity mapping [21]. This mitigated the vanishing gradient problem (inherent to deep networks) and, in turn, allowed for even deeper networks to be created than before.

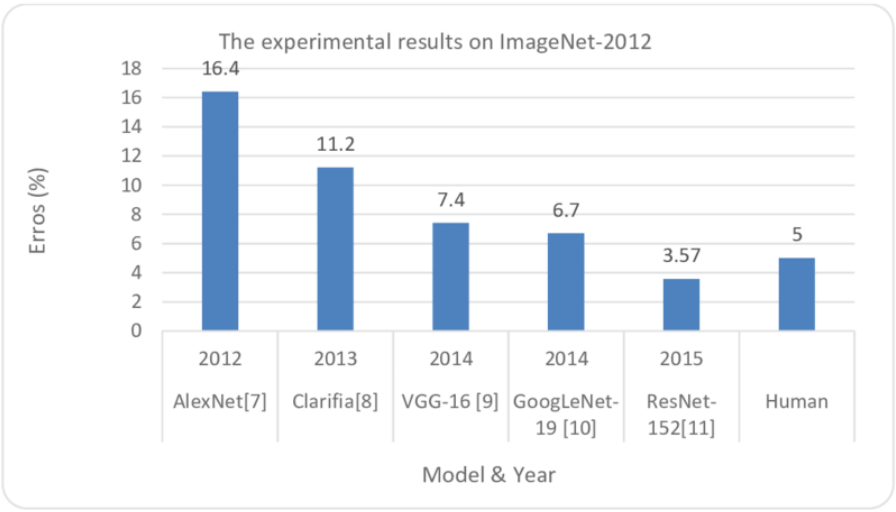


Figure 2.8: Improvements in accuracy in classification of ImageNet dataset (winning architectures 2012-2015) (taken from Alom et al. [1]).

With deeper and larger networks accuracy could be significantly improved, however, this came with the downside of higher computational cost. Out of this limitation came an area of study focusing on optimizing the computational cost of a network whilst ensuring accuracy is not significantly degraded. Lebedev et al. [33] demonstrated that CP decomposition can be used to get an 8.5x CPU speedup at a cost of only 1% drop in accuracy when training AlexNet on the ImageNet dataset. Similarly, tensor methods were utilized by Kossaifi et al. [29] by imposing a low-rank constraint on the activation layers known as Tensor Contraction Layers (TCLs) and on the regression weights known as Tensor Regression Layers (TRL). When applied to VGG and ResNet architectures, TCLs and TRLs reduced the

number of parameters compared to fully connected layers by more than 65% (in turn reducing the computational cost) while maintaining or increasing accuracy. Howard et al. [25] proposed MobileNetV1 which employs depth wise separable convolution to substantially improve computation efficiency over standard convolutional layers. MobileNetV2 [50] built on that by introducing a resource-efficient block with inverted residuals and linear bottlenecks. MobileNetV3 [24] introduced a way in which automated search algorithms (using reinforced learning) can be used to find the optimal architecture that balances computational speed and accuracy. ShuffleNet [60] utilizes group convolution and channel shuffle operations to reduce computational cost. Hinton et al. [23] proposed teaching a large "teacher" network that can then guide smaller "student" networks. Complementary to the development of novel architectures, quantization has been used to improve efficiency through reduced precision arithmetic [26].

2.2.2. Machine learning in solid mechanics

With the success of machine learning in various computing tasks (such as machine vision) it has garnered large interest in the solid mechanics field. A common approach called metamodeling has been extensively used whereby a surrogate model is built to approximate the behavior of numerical techniques such as the finite element method. This can be applied to both forward problems (finding the outcomes of a system given an input) or inverse problems (exploring the inverse input space given an outcome). These methods are often called data-driven since they rely on data to create the model. Several studies have been conducted on data-driven modeling using supervised machine learning.

In simple cases, studies have focused on predicting a single or few quantities of interest given a microstructure. Pathan et al. [43] predicted the homogenized response of composite RVEs using a gradient-boosted tree regression model with principle component analysis used to obtain features from the two-point correlation function of the microstructure. Gholami et al. [18] utilized ResNet and AlexNet to predict the Young's modulus and Poisson ratios of composite hydrogels. Using a regression tree model, Zhenchao et al. [44] also successfully predicted the elastic properties of composite RVE's.

Other studies have focused on more complex predictions such as full field predictions. Bolandi et al. [3] developed a novel architecture for stress prediction of steel plates. The study extensively compared a large number of parameters such as using various architecture blocks, loss functions and dataset sizes to evaluate and optimize their architecture. Mozzafar et al. [41] used recurrent neural networks to show that plasticity could be precisely and efficiently predicted using deep learning methods. Their model was capable of learning reversible, irreversible, and history-dependent phenomena of different deformation paths. Sepasdar et al. [52] trained a U-Net architecture to predict the stress field of a composite RVE. Using the output from the first model as inputs, they trained a second U-Net model to predict cracks in the RVE.

Although most of the mentioned literature has reported increases in computational speed when comparing the inference of their surrogate models against their corresponding numerical technique, it can be said that there is a lack in consistency when trying to compare their approaches, both in terms of accuracy and time. To better standardize results of machine learning architectures across the solid mechanics field, and similar to ImageNet in computer vision, Lejeune [35] created the mechanical-MNIST dataset that could be used by the community. Its data consists of heterogeneous microstructures of varying stiffness and is derived from the MNIST dataset. Following an FEM simulation of each microstructure, into the plastic regime and under different loading conditions, the dataset targets consist of stress/strain fields and total strain energy which corresponds to full field and single quantity of interest predictions, respectively.

Unsupervised techniques have also been used within solid mechanics. For example in the case of data reduction/clustering with the Self Consistent-clustering Analysis (SCA) [37] and Proper Orthogonal Decomposition (POD) [45]. Another notable example are Physics Informed Neural Networks (PINN's) which incorporate neural networks and autodifferention into the solution scheme of partial differential equations [40].

2.2.3. Operator Learning

Operator learning is a useful framework from which data-driven models can be understood. A popular approach for learning an operator such as $\mathcal{G} : L^2(\mathcal{X}) \to L^2(\mathcal{Y})$, where $L^2(X; \mathbb{R}_d)$ is a set of continuous functions from a set X to $\mathbb{R}^d$, is a description using three maps [51].

$$\mathcal{G} \approx \mathcal{F} = \mathcal{D} \circ \mathcal{A} \circ \mathcal{E} \tag{2.27}$$

The first map, $\mathcal{E} : L^2(\mathcal{X}) \to \mathbb{R}^m$, is the encoder. It takes the input function and maps it to a finite-dimensional feature representation. This can for example be m finite samples from the original function or a projection of the function onto a set of m basis functions. The second map, $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^n$, is the approximation map. It is the finite-dimensional approximation of the actions performed by the operator $\mathcal{G}$. The last map, $\mathcal{D} : \mathbb{R}^n \to L^2(\mathcal{Y})$, is the decoder step, which brings the latent dimensional representation in $\mathbb{R}^n$ back to the function space of the input.

$$\begin{array}{ccc} L^2(\mathcal{X}) & \xrightarrow{\mathcal{G}} & L^2(\mathcal{Y}) \\ \mathcal{E}\downarrow & & \uparrow\mathcal{D} \\ \mathbb{R}^m & \xrightarrow[\mathcal{A}]{} & \mathbb{R}^n \end{array} \tag{2.28}$$

Using the operator learning methodology it has been proven that neural networks with arbitrary non-linear activation functions serve as a universal approximation for any continuous function [7]. Nonetheless, the proof of universal approximation doesn't provide an estimate for the size of the latent space for which universal approximation can be guaranteed. Instead, it states that universal approximation can be achieved given a large enough latent space.

2.3. Data generation

A big determining factor in the effectiveness of a data-driven model is the data that is provided to it. It can affect a model's accuracy, training time and generalization. Additionally, as highlighted in section 2.2 large amounts of data are typically required to train deep networks. This section highlights some potential ways of creating datasets of heterogeneous microstructures, with specific emphasis on ways in which clustered microstructures (used in SCA) can be obtained or generated.

One method by which clustered microstructures could be generated is using representative material microstructures that undergo a clustering procedure. The material microstructures could be obtained from real microstructures or generated synthetically. Since SCA operates on a regular grid, it has been suggested as an effective tool for modeling microstructures from micrographs of real material microstructures [15]. However, this has not been done in practice and as such advances would need to be made in the method (in areas such as image denoising) before it can be considered robust enough for a data-driven approach. Furthermore collecting a sufficiently large sample of micrographs can be prohibitively costly and time consuming. In practice, SCA has generally been used with synthetically generated material microstructures. This can be done using the Cahn-Hilliard equations [6] to create a binary material phase field, which has typically been used for generating microstructures of metal alloys [19]. Several methods have also been suggested for generating microstructures of composite microstructures such as RAND_uSTRU_GEN [38] and AMINO [14]. Although generating a dataset based on these microstructures would make it representative of similar microstructures, it might fail to generalize to other microstructures. Furthermore, generating the dataset using material microstructures would be computationally expensive due to the additional step of clustering the microstructure as well as the high cost associated with the generation of each microstructure for some of the methods highlighted (which can be in the order of minutes for a single microstructure).

A viable alternative could be generating the clustered microstructures directly using noise. In the simplest case white noise (Figure 2.9) can be generated by sampling a uniform random value at each voxel. However, this assumes that every voxel is independent of one another and therefore doesn't include spatial correlations between neighboring points. To create spatially correlated noise, gradient noise fields (Figure 2.10) have been successfully utilized in many disciplines. Notably, Gaussian noise has been used for soil mechanics to create random soil structures for permeability studies [58] and fluid dynamics for creating random vorticity fields [30] (in both cases this was later used within a data-driven modeling framework). Another example of the use of Gaussian noise is the work by Vel and Groupee [56] who used Gaussian noise to create biphasic material microstructures to evaluate a homogenization scheme. Gaussian noise can be generated quickly by initializing and masking white noise in the frequency domain and transforming this into the spatial domain using the FFT [32]. Jakes et al. [27] presented a similar framework for dataset generation by using Perlin noise to make fibrosis patterns. Although fast, Perlin noise suffers from axis-aligned anisotropy which adds artifacts to the field [32]. Interestingly, Jakes et al. [27] noise method used a root finding algorithm to pick the binarization plane that enforced a predetermined volume fraction. They also superimposed different noise

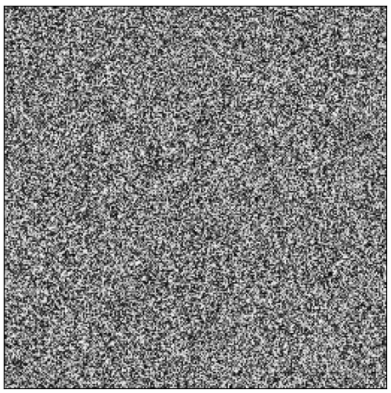


Figure 2.9: White Noise.

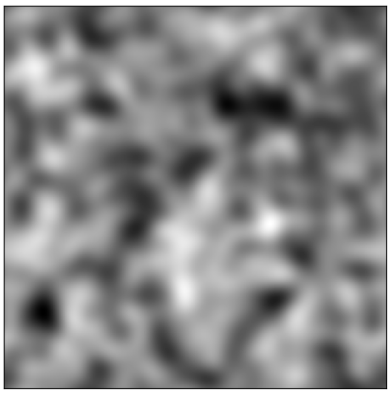


Figure 2.10: Gradient Noise.

fields, with various bases, to control the feature size and anisotropy of the final structure. Altogether these methods offer a fast way of generating random spatially correlated fields at varying length scales. However, to the best knowledge of the author, no attempt has yet been made at generating multi-phase microstructures using gradient noise. Currently, the clustering of fields has been restricted to simple binarization.

2.4. Algorithmic complexity

Algorithmic analysis is the study of how an algorithm's efficiency at solving a particular problem compares to others. Most commonly this efficiency is measured in terms of time complexity or space complexity [9]. This section introduces some commonly used metrics for measuring the complexity of an algorithm.

A simple approach for measuring time complexity is the wall clock time [5]. In this approach an algorithm is executed on a machine and the time it takes the algorithm to evaluate the problem to completion is noted. One downside of using wall clock time to design algorithms is that an algorithm needs to be fully programmed to obtain this metric. This makes the process time consuming when comparing different algorithms. Furthermore, the wall clock time can be influenced by how an algorithm is implemented, e.g. the machine it is run on or the compiler that is used. Lastly, the wall clock time is not informative for considering other (similar) problems that the algorithm may encounter in the future.

To counter the limitations of wall clock time the big-O ($\mathcal{O}$) notation is used [5]. It is a metric typically applied to P class (polynomial time) problems and considers the algorithm's asymptotic efficiency. This can intuitively be understood to be how much an algorithm's time complexity grows as a function of input size (n). Some of the most common complexity classes are seen in Figure 2.11.

Formally big-O can be defined as follows:

> **Definition**. "A function g belongs to the complexity class $\mathcal{O}(f)$ if there is a number $n_0 \in N$ and a constant $c > 0$ such that for all $n \geq n_0$, we have that $g(n) \leq c * f(n)$. We say that the function g is 'eventually smaller' than the function $c * f$." [5]

This implies that constant factors do not change the growth class. Furthermore, it means that the behavior of the function for small inputs is irrelevant. Through this definition, the complexity classes seen in Figure 2.11 can be defined as elements of a set:

$$\mathcal{O}(1) \in \mathcal{O}(\log n) \in \mathcal{O}(n \log n) \in \mathcal{O}(n^2) \in \mathcal{O}(2^n) \in \mathcal{O}(n!) \tag{2.29}$$

As such, for functions formed from different growth classes, the largest growth class defines the complexity of the function. For example, a function with growth class $C(n) = 500000 \log n + 4n^2 + 0.3n + 100$ would have the component $4n^2$ as its largest growth class. Furthermore, as a result of the definition, the constant term would not affect the complexity. Hence, the complexity class of function $C(n)$ would be $\mathcal{O}(n^2)$.

Nevertheless, big-O notation has limitations. Namely, that it only considers an algorithm's asymptotic performance. This is best exemplified by galactic algorithms which are algorithms with a small

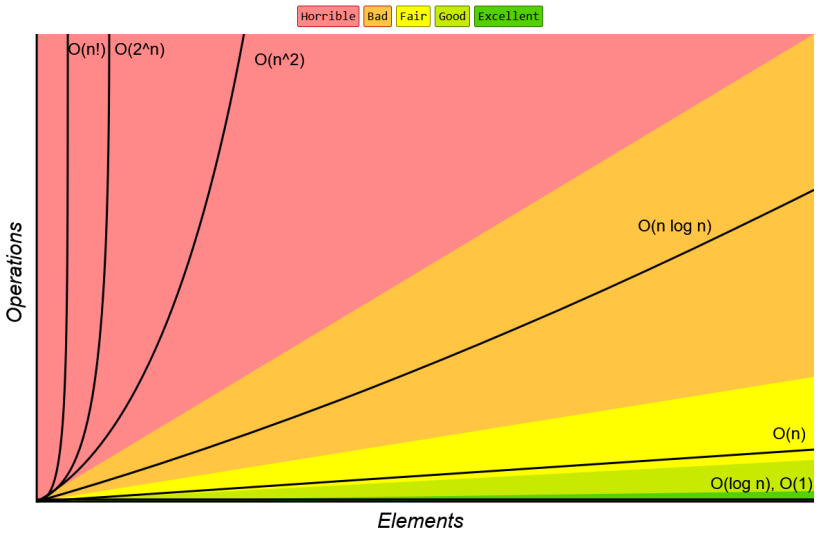


Figure 2.11: Growth of most common complexity classes (taken from Rowell et al. [48]).

complexity class that are never used in practice due to their high cost for realistic input sizes, potentially requiring datasets on the scale of the universe [36].

To counter this limitation, the number of multiplications is often used as an estimate of complexity. The reason why multiplications are considered over other operations such as additions is because they are one of the most computationally expensive floating point operations. This can be understood by considering the big-O complexity of an example operation on two integer values. When multiplying the two integers the complexity would be $\mathcal{O}(n^2)$ whereas adding them would have a complexity of $\mathcal{O}(n)$. This metric assumes that the algorithms being compared operate using the same datatype e.g. 32-bit floats. It is limited to considering an input of a single size but can be convenient for evaluating an algorithm when the range of input sizes is known [16].

However, it has also been argued that multiplications and additions have a compute time of similar order and can therefore be categorized in the same group. In such a metric additions would be included when evaluating the complexity of an algorithm. Due to the extensive use of multiplications and additions in neural network algorithms, this is the most commonly used metric in the ML community. It can be referred to as Floating Point Operations (FLOPs) when additions or multiplications each count as one unit or AddMul when a pair of an addition and a multiplication operation is counted as one. The AddMul unit comes from the idea that neurons in a neural network scale an input by a weight and translate by a bias, making it convenient to count computational complexity simply by summing the number of connected neurons [10].

2.5. Research question

Based on the conducted literature review, two primary knowledge gaps can be identified:

KG1 **Fast computations of the cluster interaction tensor:** The computation of the cluster interaction tensor scales with $\mathcal{O}(n^2)$ as a function of number of clusters. Therefore, it is the driving bottleneck in the offline stage of SCA when the number of clusters is increased (as seen in Figure 2.5). It also significantly affects the solution time in the online stage when adaptivity is used. Increasing the computational speed of this step can further improve upon the computational efficiency provided by SCA in making material predictions.

KG2 **Efficient generation of large and diverse multi-phase microstructure datasets:** Given the interest in training data-driven models on heterogeneous microstructures, different ways of generating the datasets and their effect on the models need to be explored. For example, a dataset of representative microstructures such as composites or metals can be slow and limit the generalizability of a model trained on said dataset. Gradient noise has been used to counter this limitation, although so far microstructures have been limited to simple biphasic structures.

Based on these knowledge gaps the following research question is formulated and guides the thesis.

To what extent can a data-driven surrogate model accurately predict and accelerate the computation of the cluster interaction tensors (CIT) in self-consistent clustering analysis (SCA)?

To answer the primary research question and address the knowledge gaps further, several sub-questions can also be formulated:

SQ1 To what extent can a data-driven surrogate model accurately predict the CIT?

SQ2 To what extent can a data-driven surrogate model accelerate predictions of the CIT?

SQ3 How can data for a surrogate model's training be generated?

3

Synthetic Data Generation

This chapter introduces a novel method for generating large datasets of clustered microstructures using gradient noise. Section 3.1 highlights the motivation behind the use of gradient noise for the problem of clustered microstructure generation. Section 3.2 discusses how a single cluster can be generated. Section 3.3 expands upon the cluster generator to the generation of clustered microstructures. Section 3.4 concludes by showing how full datasets of synthetic clustered microstructures are generated.

3.1. Motivation

To motivate the methodology needed to generate the synthetic data one must first revisit the offline stage of the SCA introduced in chapter 2. SCA utilizes an offline linear elastic DNS simulation to compute the strain concentrations along all material points of a domain. It then clusters points using the k-means minimization which results in points with similar strain concentration being clustered together. This entails that the geometry of the clustered microstructure is a product of the strain field exhibited by the structure.

In most cases, a material undergoing deformation can be assumed to have a smooth and continuous strain field (and by extension the strain concentration). Under such an assumption the points in the field are said to be spatially correlated i.e. points close to one another will exhibit a similar strain value. As a result of this, SCA produces clusters that are spatially correlated, even though clusters don't need to be continuous. This motivates the need for a spatially correlated method of generating synthetic data such as Gaussian noise introduced in section 2.3. It's noted that the strain concentration field is disturbed when different material phases are present [42]. The strain concentration of a single composite RVE can be seen in Figure 3.1. Notice the strain concentration field's smoothness in the matrix phase and abrupt change between matrix and fiber phases.

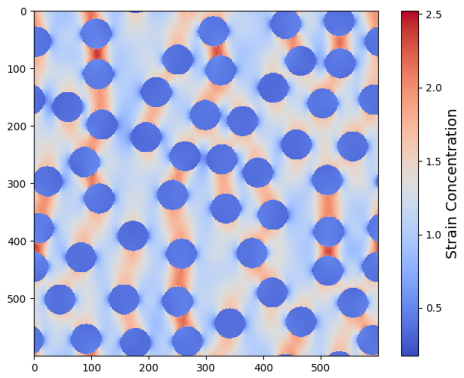


Figure 3.1: Strain concentration field of a composite RVE undergoing uniaxial tension. Composite consists of circular fibers embedded in a matrix.

Furthermore, as highlighted in section 2.2, data-driven methods require large datasets to train. Current RVE microstructure generators can take in the order of seconds or even minutes to generate a microstructure and also require undergoing the offline stage DNS simulation and clustering step in order to obtain a clustered microstructures. Hence a method for generating large datasets of clustered microstructures efficiently is required. Once again, with Gaussian noise being very cheap to initialize, it presents itself as a viable solution for the problem at hand.

3.2. Cluster generator

Following the observations made in section 3.1, it can be concluded that most clusters tend to clump together due to the continuous nature of the strain in the medium. Therefore if the strain field is described using the frequency domain, it can be assumed that the lower frequencies should yield a sufficiently good approximation of the field. Only under sharp jumps in the field (such as at phase boundaries) are high-frequency components needed to yield a sufficiently good approximation. However, this can be emulated by simulating discrete changes in the field by obtaining noise from separate seed states for different parts of the domain. Furthermore, periodic boundary conditions can be enforced using Fourier spectral synthesis to be consistent with the formulation of SCA. Following this reasoning, the following methodology for generating clusters has been developed.

A schematic of the cluster generator algorithm is shown in Figure 3.2. It begins by initializing an array of white noise in the complex domain ($\mathbb{C}$). To aid in visualization, the complex array has been split into the real and imaginary components in the figure. A Gaussian filter of given roughness is then applied. The roughness parameter corresponds to the variance of the Gaussian filter and the mean of the filter is centered at zero frequency. The complex array is then inverse Fourier transformed to get an array in the real domain ($\mathbb{R}$). This array is then binarized according to a threshold to get an array of ones and zeros. The binarization step sets the threshold such that, after masking by the available domain array, the output is of a desired volume fraction.

Some relevant remarks are in order:

- Since the noise is composed of a set of frequency components the resulting image is periodic in the real domain. This is particularly useful as the SCA simulations performed in subsequent sections will enforce periodic boundary conditions on the domain.
- The inverse Fourier transform is efficiently performed using the fast Fourier transform method.
- A standard inverse zero-frequency shift is applied on the filtered noise prior to the inverse Fourier transform.
- The volume fraction of the cluster is computed by calculating the ratio of ones present in the matrix over the total entries in the matrix.
- A root-finding algorithm is implemented to compute the binarization threshold at which a cluster with the desired volume fraction is found from the filtered noise. In this case, Brent's method is used since it provides a guaranteed solution, at a faster rate than the bisection method, and does not require a gradient.

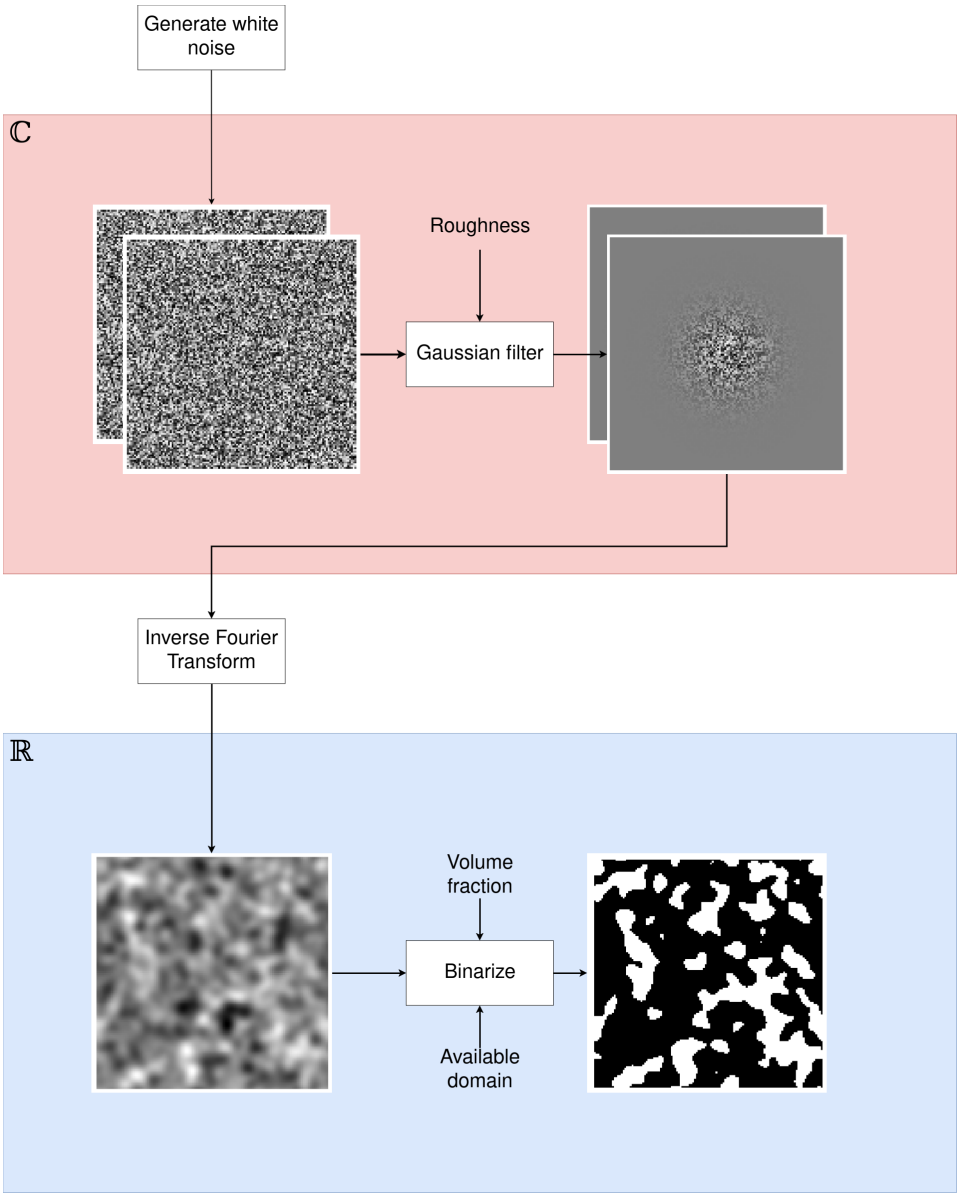


Figure 3.2: Cluster generator implementation flow chart. The program first generates noise in the complex domain ($\mathbb{C}$). Complex arrays are separated in the figure into real and imaginary components (for visualization purposes). The program then applies a Gaussian filter followed by a transformation into the real domain ($\mathbb{R}$) concluded by a binarization step.

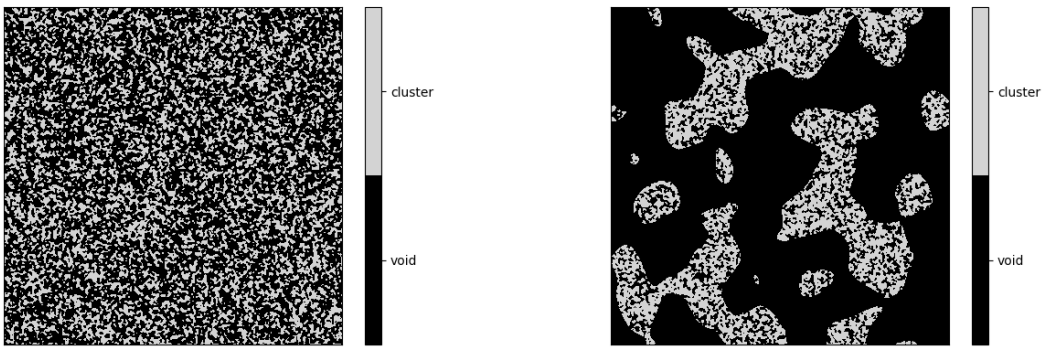
3.3. Microstructure generator

The previously explained cluster generator can then be utilized to generate microstructures consisting of multiple clusters (clustered microstructures). To explain the microstructure generator, an example where a microstructure with three clusters is generated is demonstrated in Figure 3.3.

The available domain is tracked throughout the computation. This acts as a mask in the binarization step and limits the domain that a newly generated cluster can fill. It serves as the condition that ensures that no clusters overlap.

The cluster generator takes as inputs the available domain, desired volume fraction and desired roughness parameters. The volume fraction and roughness parameters are initialized for all clusters in a microstructure from a distribution a priori. The volume fraction is taken from a uniform distribution and then normalized so that the sum of all volume fractions adds up to unity (the full microstructure domain). The roughness parameter is initialized from a log-normal distribution. The log-normal is used in order to generate microstructures where a majority of clusters are generated with small roughness values, whilst including a few clusters with higher roughness values (that approach white noise). The cluster generator step outputs a new cluster and updates the available domain to exclude the domain of the newly created cluster. The final cluster is simply computed by filling the remaining available domain. Finally, the generated clusters are added together to form the clustered microstructure.

Since the microstructure consists of layers of clusters stacked on top of each other, a ”biting” bias presents itself in clusters that are generated following the first one (as shown in Figure 3.4). This bias gets compounded the more clusters are generated due to a decrease in available domain.



(a) Cluster 1 **(b)** Cluster 2

Figure 3.4: First two generated clusters in microstructure. Biting bias can be seen present in cluster 2.

The purpose of the microstructure generator is to generate large datasets to train surrogate models. To demonstrate its efficiency, the generation of microstructures with different resolutions and numbers of clusters was timed. The wall clock time to generate 100 microstructures given a set of parameters is found and averaged to estimate the wall clock time required to generate a single microstructure. The results are shown in Table 3.1.

Resolution	**Clusters**		
	5	**20**	**80**
56x56	0.004s	0.016s	0.065s
112x112	0.007s	0.047s	0.157s
224x224	0.025s	0.129s	0.564s

Table 3.1: Average wall clock time (in seconds) to generate a microstructure with given resolution and number of clusters.

It is observed that the wall clock time is at most in the order of 10^{-1}s which is faster than composite RVE generators that can take in the order of seconds or even minutes to generate a single RVE of the same size [14, 38]. Furthermore, the microstructure generator provides microstructures in a clustered form, obviating the need to run the offline DNS simulation and clustering procedure.

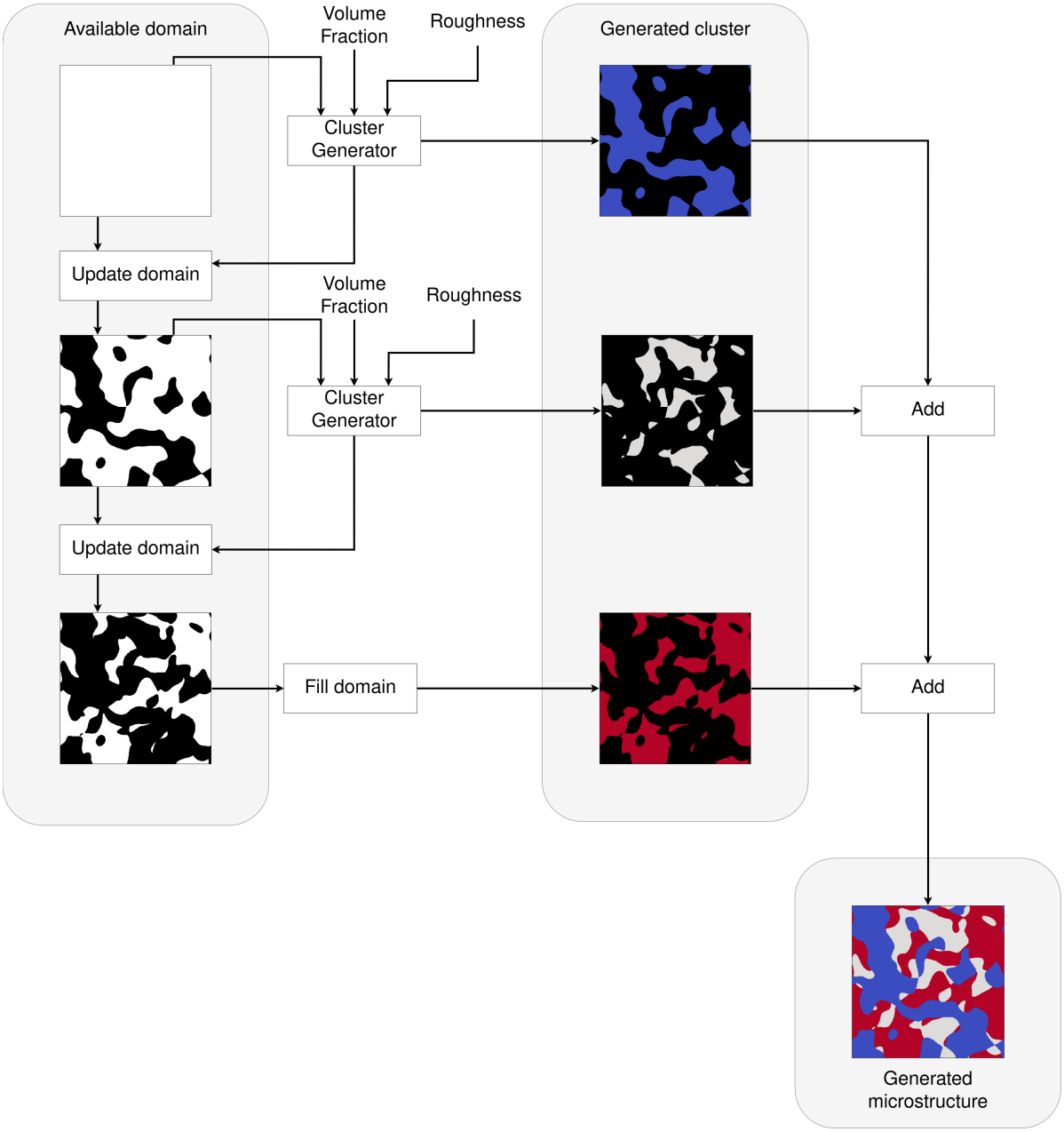


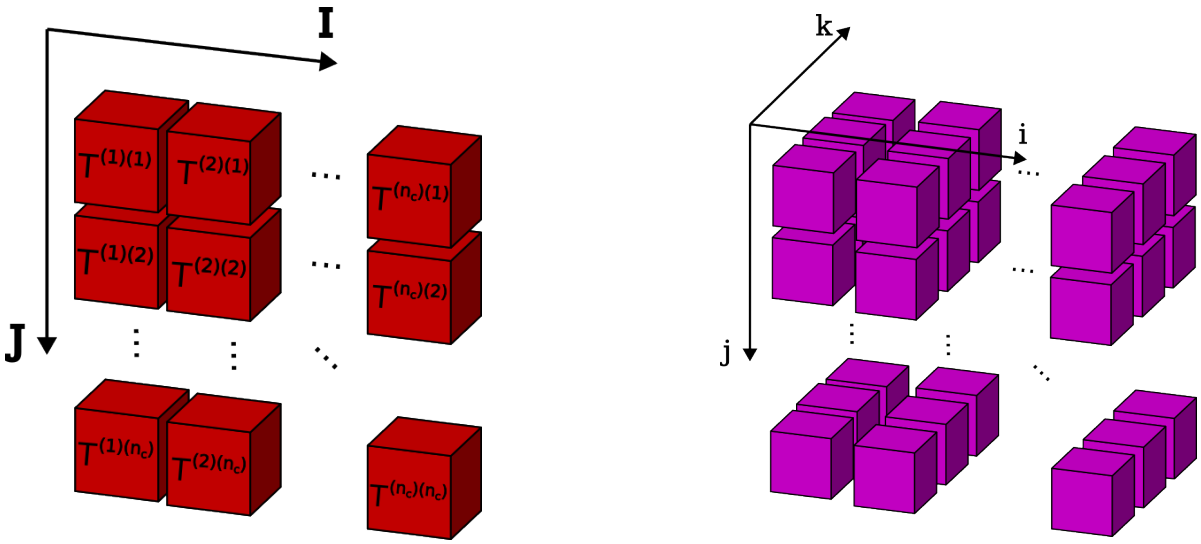
Figure 3.3: Microstructure generator flow chart (example for microstructure with 3 clusters). Clusters are generated given a volume fraction and roughness parameter. The available domain is kept track of to ensure no clusters overlap. The final cluster is generated by filling the remaining available domain. Microstructures are created by adding all clusters together.

3.4. Dataset generator

In order to facilitate the dataset generation, a choice of target must first be established. Naturally, this will be the CIT but there is a multitude of ways in which the quantity can be expressed. In the form given by Equation 2.18, the CIT is a function of the reference materials Lamé parameters and, as such, is subject to the self-consistent optimization step of the online stage. Predicting the CIT in this form is a considerably difficult task given the complexity of the physics involved. Therefore, the offline parameters of the CIT are predicted instead. By using the split version of the Green operator given by Equation 2.23 the CIT can be reformulated in the form

$$\mathbf{T}^{(I)(J)} = c_1(\lambda^0, \mu^0)\ \mathbf{T}_1^{(I)(J)} + c_2(\lambda^0, \mu^0)\ \mathbf{T}_2^{(I)(J)}, \tag{3.1}$$

where c_1 and c_2 are the material dependent parameters given by Equation 2.25. The component $\mathbf{T}_3^{(I)(J)}$ is also used and corresponds to the zero frequency of the Green operator. In the computational implementation, the offline CIT component is denoted using $\mathrm{T}_{ijk}^{(I)(J)}$ (with the subscripts ijk) where i and j are the tensor directions and k is a component index (1, 2 or 3). From here on, the tensor $\mathbf{T}$ is also referred to as a "global tensor" whereas the component with a set I and J, denoted by $\mathbf{T}^{(I)(J)}$, is referred to as a "local tensor". This is visualized in Figure 3.5. The global tensor contains $n_c \times n_c$ local tensors (where n_c is the number of clusters).



(a) Structure of global tensor $\mathbf{T}$. n_c is the total number of clusters.

(b) Structure of local tensor $\mathbf{T}^{(I)(J)}$. Notice that the length of k is fixed whereas i and j are variable.

Figure 3.5: Array structure of global and local cluster interaction tensors.

Another decision to be made is whether to predict global or local tensors. One issue in predicting the global tensor is that the output needs to vary in size as a function of the input size. This is because the number of local tensors grows quadratically as a function of the number of clusters. This imposes a severe limitation on the possible architectures that can be considered. Additionally, although architectures such as transformers [55] are capable of handling variable input and output sequences, these methods typically rely on padding and masking operations to convert a fixed-size architecture into a variable-size setting. This imposes a limitation on the current application since the number of parameters in such an architecture is dictated by the highest number of clusters, which could result in the model being over-parameterized for cases with few clusters. As such, for the current study, a "pairwise" strategy is chosen whereby the local tensor is predicted for cluster pairs, as seen in Figure 3.6. Nevertheless, the global tensor strategy will be discussed again in chapter 6.

Attention is called upon the fact that the CIT contains n_c^2 number of local tensor components. It is these local components that the surrogate will later predict in its pairwise approach (with each local component corresponding to a data point). Due to the scaling of the global tensor, care must be taken when dealing with datasets containing microstructures of varying cluster numbers. This is because a microstructure containing many clusters yields significantly more data points than a microstructure

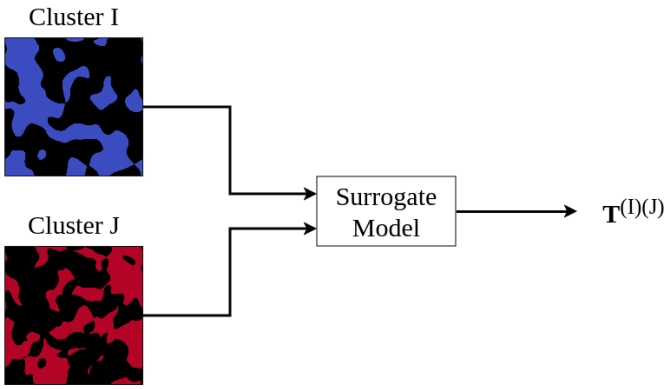


Figure 3.6: Data-driven modeling pairwise approach where the surrogate model predicts the local cluster interaction tensor between two material clusters from the corresponding support functions.

containing few clusters. The average volume fraction of a cluster is also proportional to the number of clusters in a microstructure. Microstructures with more clusters on average have clusters with smaller volume fractions. Therefore actions need to be taken in order to ensure a balanced dataset.

Two different approaches are considered to correct for this. In the first, microstructures are be binned into groups defined by the number of clusters. The number of microstructures in each bin is set such that the amount of local tensor components they contributed is equal across bins. This method, however, is not chosen due to the rigid structure imposed on the binning and the number of microstructures permitted in each bin. Additionally, this approach can lead to an unbalanced dataset since bins contain different total amounts of clusters. This, for example, results in clusters being repeated more in bins containing microstructures with many clusters.

Instead, an approach where the global tensor is truncated is proposed. This method involves truncating the volume fraction and roughness lists to a length of $n_{c,cut}$. For these microstructures, the "fill domain" operation shown in Figure 3.3 is not executed, effectively generating microstructures with voids in them. This, however, does not affect the surrogate model as training occurs on a pair-wise basis. To obtain a balanced dataset, $n_{c,cut}$ is set to the minimum number of clusters in the range considered. Essentially, the method only generates local tensors up to a certain point (retaining those values) and does not generate other local tensors (effectively discarding those values). This is visualized in Figure 3.7. This method has the additional benefit of providing more flexibility on the number of clusters each microstructure could take on. The Sobol sequence is therefore used, within a given range, to determine the number of clusters in each microstructure, as such, filling the space more efficiently.

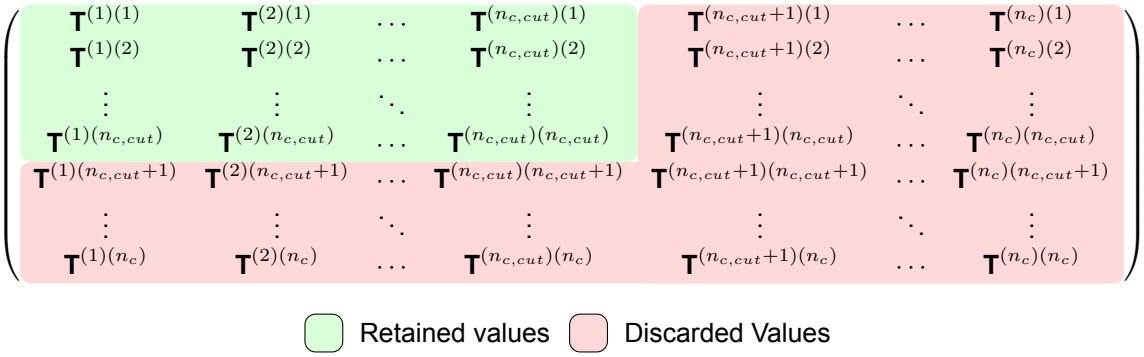


Figure 3.7: Truncation of the global tensor used to obtain a balanced dataset.

4

Data Driven Predictions

This chapter explores data-driven predictions of the CIT. First, section 4.1 introduces the software used to calculate the CIT and the algorithms behind its implementation. Then, in section 4.2, the underlying distributions in the CIT are explored, and a strategy for CIT predictions is proposed. Next, in section 4.3, a sensitivity analysis shows how uncertainty in the CIT could affect SCA. Following that, the methods pertaining to surrogate model evaluation are introduced in section 4.4. With the methods established, a standard surrogate model is introduced, trained and evaluated in section 4.5. A potential limitation is observed in the standard surrogate model and section 4.6 aims to address the limitation by modifying the dataset. This is followed by a convergence analysis with respect to the dataset size in section 4.7. Finally, an accelerated version of the surrogate is presented in section 4.8.

4.1. Baseline model

A baseline (analytical) model must first be implemented against which the surrogate (data-driven) model for solving the CIT can be compared. The physics and derivation of the necessary equations for such a method are highlighted in section 2.1. Conveniently, an open-source implementation of SCA called CRATE has been provided by Ferreira et al. [13]. It includes a step where the CIT is computed given a microstructural description. The code is tested and well documented and many of the decisions made to optimize it have been summarized by Ferreira [15].

Since the CIT, specifically, is the quantity of interest in this study, the algorithmic implementation of its computation in CRATE is highlighted in algorithm 1. Note that $[Y]_{2d}$ is an array representing the material domain. Colors have been provided to illustrate the iterations over various indices better. Notice that the I and J indices form a nested for-loop. It is due to this nesting that the algorithm scales with $\mathcal{O}(n_c^2)$.

Algorithm 1: CIT pseudocode.

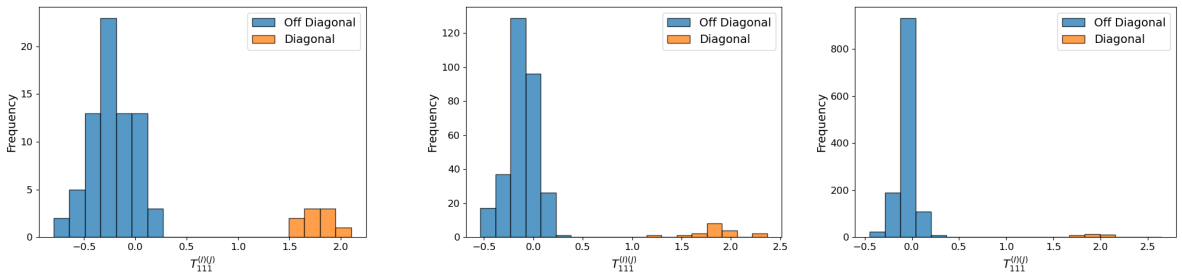
Input : $[Y]_{2d}$, n_c, n_{comp}
Output : $T^{(I)(J)}$

Obtain $\breve{\Phi}_{ijkl}$;
for $J = 1$ *to* n_c **do**
  $[\breve{X}]^J_{2d} \leftarrow \mathscr{F}(\chi([Y]_{2d}))$;
  for ij *in* n_{comp} **do**
    for kl *in* n_{comp} **do**
      $[M_{ijkl}]^J_{2d} \leftarrow \mathscr{F}^{-1}([\breve{X}]^J_{2d} \circ \breve{\Phi}_{ijkl})$;
    end
  end
  for $I = 1$ *to* n_c **do**
    if $I > J$ **then**
      $[T^{(J)(I)}] \leftarrow \frac{f^{(I)}}{f^{(J)}}[T^{(I)(J)}]$;
    else
      $[X]^I_{2d} \leftarrow \chi([Y]_{2d})$;
      for ij *in* n_{comp} **do**
        for kl *in* n_{comp} **do**
          $[T^{(J)(I)}] \leftarrow \sum_{voxels} [X]^I_{2d} \odot [M_{ijkl}]^J_{2d}$;
        end
      end
      Obtain c_1 ;
      $[T^{(J)(I)}] \leftarrow c_1[T^{(J)(I)}]$;
    end
  end
end

4.2. Data analysis and pre-processing

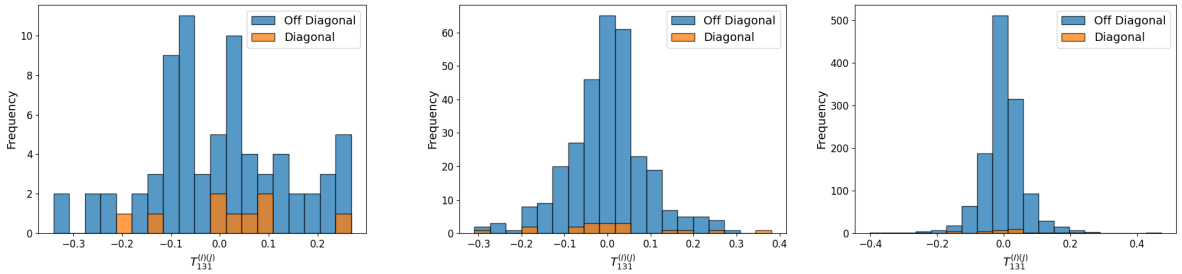
Since data plays a crucial role in any data-driven model, it must first be analyzed and pre-processed to ensure easier training and accurate predictions.

In the first step of this stage, histograms showing the CIT values for a set index of a local tensor are created, from which it can be observed that some local elements exhibit a bi-modal distribution while others exhibit an unimodal distribution. An example of a bi-modal distribution is shown in Figure 4.1 and a unimodal distribution is shown in Figure 4.2. Upon further inspection, it can be determined that the two modes in the distribution could be distinguished through their position within the global tensor. Namely, one mode corresponded to the diagonal terms of the global tensor, whereas another corresponded to the off-diagonal terms. The diagonal terms correspond to the self-interaction case. This result, and the fact that the diagonal mode is higher than the off-diagonal, makes intuitive sense since it is to be expected that the stress within a cluster has a higher importance in determining its own strain when compared to the stresses in other clusters.



(a) $n_c = 9$ **(b)** $n_c = 18$ **(c)** $n_c = 36$

Figure 4.1: Discrete frequency distribution showing bi-modal distribution (T^{IJ}_{111}).



(a) $n_c = 9$ **(b)** $n_c = 18$ **(c)** $n_c = 36$

Figure 4.2: Discrete frequency distribution showing unimodal distribution (T^{IJ}_{131}).

This observation informed the decision to predict only the upper off-diagonal terms using a surrogate model. Given the bimodal distribution, the performance of a data-driven model can be enhanced by predicting the two modes separately, particularly when the classes corresponding to each mode (diagonal or off-diagonal) are known a priori. Additionally, the upper off-diagonal terms scale with $\mathcal{O}(n_c^2)$, whereas the diagonal terms only scale with $\mathcal{O}(n_c)$. By predicting off-diagonal terms, any reduction in the inference times of a cluster pair will exert an increasingly pronounced effect on the global CIT tensor's computational time as the number of clusters increases. The lower off-diagonal terms are determined through the symmetry condition described by Equation 2.21. The surrogate strategy is described by algorithm 2 and visualized in Figure 4.3.

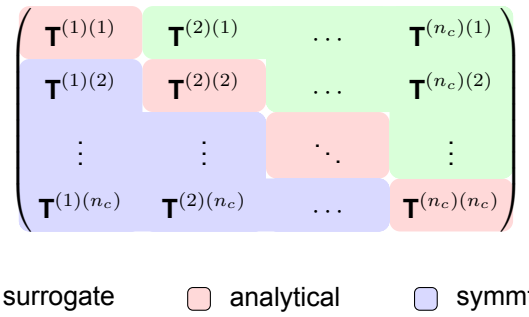


Figure 4.3: Selected strategy for computing the global tensor. The upper off-diagonal terms are computed using the surrogate model, the diagonal terms using the baseline analytical approach and the lower off-diagonal terms using the symmetry condition.

Algorithm 2: CIT with surrogate pseudocode.

Input : $[Y]_{2d}$, n_c, n_{comp}
Output : $T^{(I)(J)}$

Obtain $\breve{\Phi}_{ijkl}$;
for $J = 1$ *to* n_c **do**
  $[\breve{X}]^J_{2d} \leftarrow \mathscr{F}(\chi([Y]_{2d}))$;
  for ij *in* n_{comp} **do**
    for kl *in* n_{comp} **do**
      $[M_{ijkl}]^J_{2d} \leftarrow \mathscr{F}^{-1}([\breve{X}]^J_{2d} \circ \breve{\Phi}_{ijkl})$;
    end
  end
  for $I = 1$ *to* n_c **do**
    if $I > J$ **then**
      $[T^{(J)(I)}] \leftarrow \frac{f^{(I)}}{f^{(J)}}[T^{(I)(J)}]$;
    else
      $[X]^I_{2d} \leftarrow \chi([Y]_{2d})$;
      if $I == J$ **then**
        for ij *in* n_{comp} **do**
          for kl *in* n_{comp} **do**
            $[T^{(J)(I)}] \leftarrow \sum_{voxels} [X]^I_{2d} \odot [M_{ijkl}]^J_{2d}$;
          end
        end
      else
        $[T^{(J)(I)}] \leftarrow$ surrogate$([X]^J_{2d}, [X]^I_{2d})$;
      end
      Obtain c_1 ;
      $[T^{(J)(I)}] \leftarrow c_1[T^{(J)(I)}]$;
    end
  end
end

The next step involved standardizing the dataset. This is done since neural networks train faster when the features and targets follow a distribution with zero mean and unit variance. The standardized values are found using

$$Z = \frac{X - \mu}{\sigma}, \tag{4.1}$$

where X is the value being standardized, μ is the mean and σ is the standard deviation of the dataset. The features are not standardized since they are composed of clusters given in a binary format. The targets, on the other hand, are standardized since they represent the CIT values on a continuous scale. The discrete probability distribution showing the scaled values of every local tensor index from a dataset containing 100k clustered microstructures with 5-80 clusters are shown in Figure 4.4, Figure 4.5 and Figure 4.6. The indices 121 and 211 are not shown since all values are null. By comparing the scaled discrete density distributions to the continuous Gaussian distribution, it can be seen that none of the CIT components follow a normal distribution. Additionally it is observed that the indices 111, 221, 331, 112, 122, 212, 222, 332 and all values where $k = 3$ exhibit a skew. The skew cannot be remedied using a log transform since the distributions take on positive and negative values.

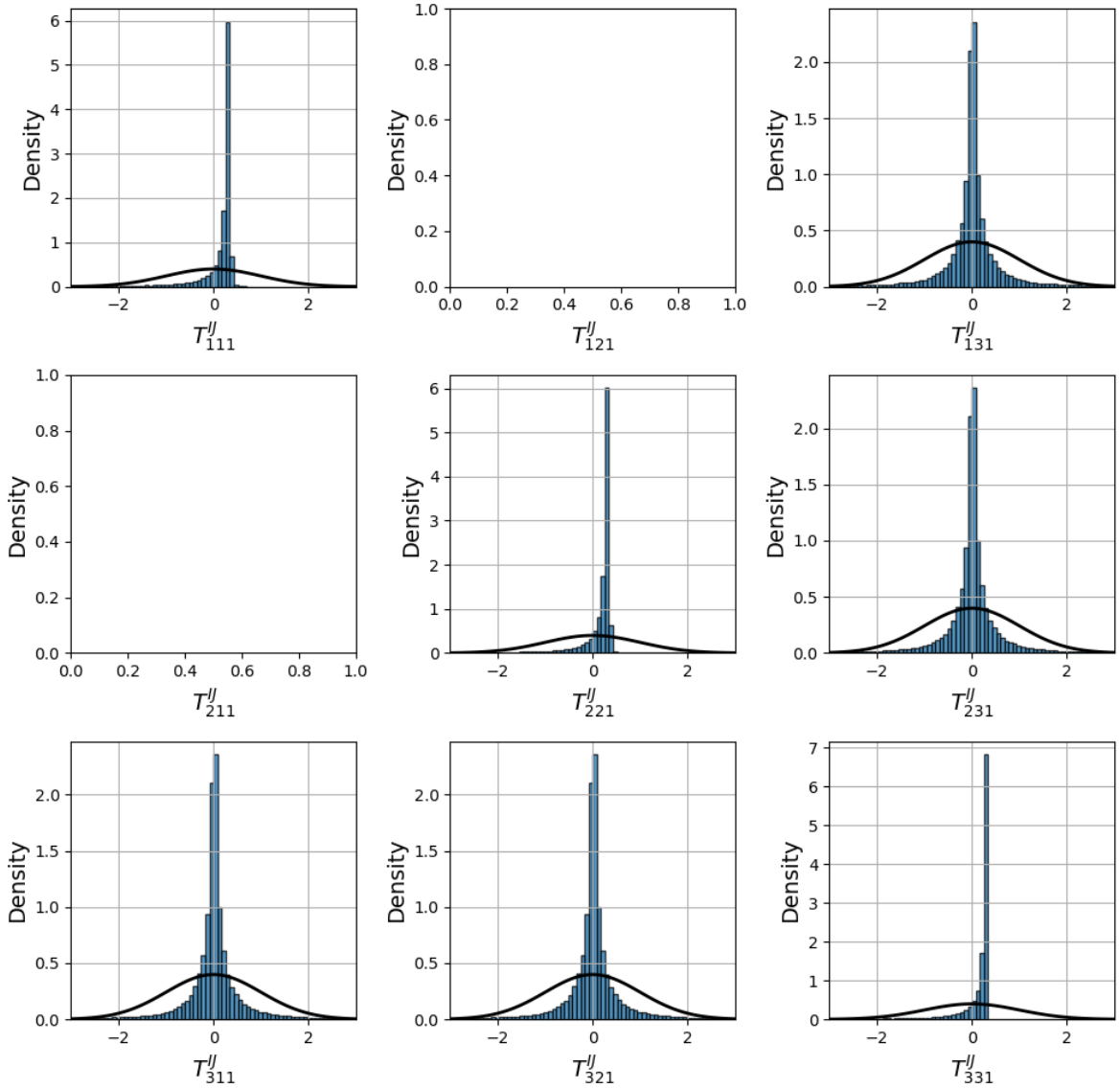


Figure 4.4: Discrete density distribution of local CIT values for $k = 1$ (after scaling) and continuous Gaussian distribution with zero mean and unit variance overlaid on top.

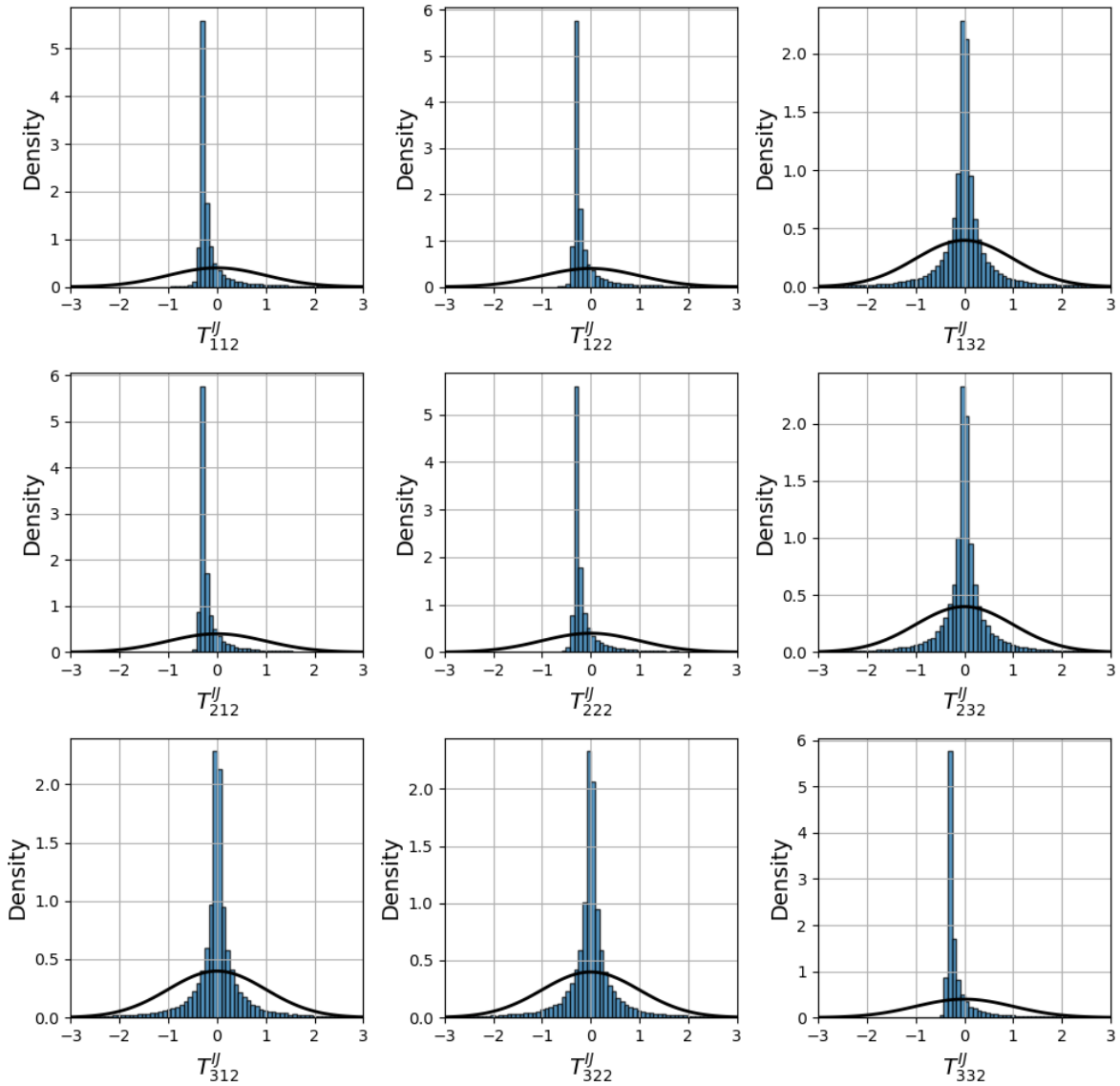


Figure 4.5: Discrete density distribution of local CIT values for $k = 2$ (after scaling) and continuous Gaussian distribution with zero mean and unit variance overlaid on top.

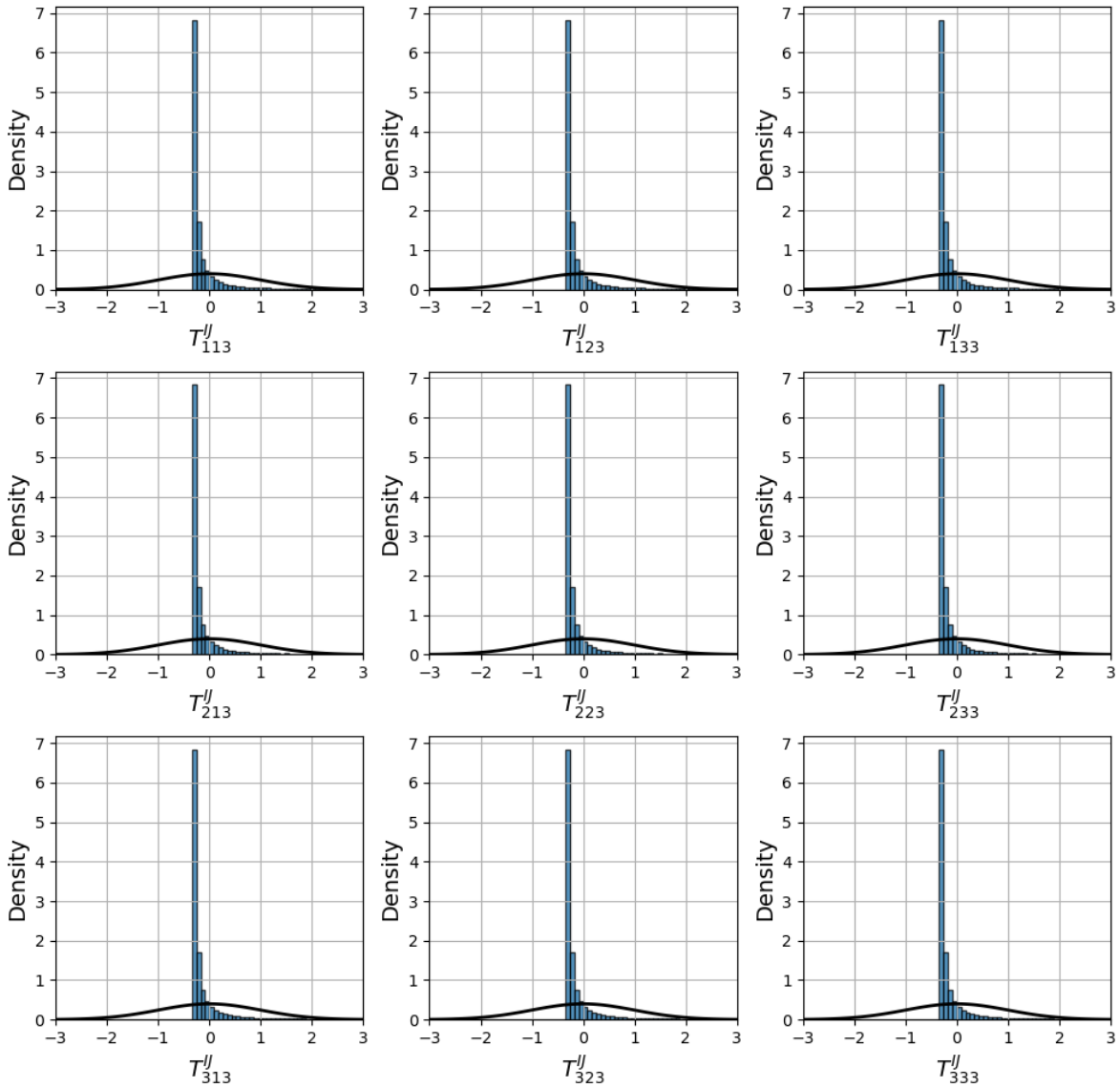


Figure 4.6: Discrete density distribution of local CIT values for $k = 3$ (after scaling) and continuous Gaussian distribution with zero mean and unit variance overlaid on top.

4.3. Sensitivity Analysis

Before building a surrogate, which is, by definition, an approximation, it is useful to gain a qualitative understanding of how the SCA solution behaves when given an approximate CIT. To do this, a sensitivity analysis is conducted to see how the model behaves given changes in certain parameters.

The sensitivity analysis is performed by injecting noise into the CIT of a benchmark example given in CRATE. The benchmark is a long fiber composite RVE undergoing uniaxial tension. The noise is injected by first transforming the CIT using a standard scaler. A noise value is then sampled for each local tensor index from a Gaussian distribution with zero mean (μ) and given standard deviation (α) corresponding to the desired noise level. It is then inverse-transformed to get the noisy solution. Noise is only applied to upper off-diagonal terms, which correspond to the values that will be approximated using the surrogate. This is given using the following equation:

$$\tilde{\mathsf{T}}^{(I)(J)}_{ijk} = \sigma_{ijk}\left(\frac{\mathsf{T}^{(I)(J)}_{ijk} - \mu_{ijk}}{\sigma_{ijk}} + N_{ijk}(\mu = 0, \sigma = \alpha)\right) + \mu_{ijk}, \quad \forall I, J, \quad I < J\,, \tag{4.2}$$

where $\tilde{\mathsf{T}}^{(I)(J)}_{ijk}$ is the noisy CIT and μ_{ijk} and σ_{ijk} are the mean and standard deviation of each local

tensor index for the single clustered microstructure, respectively. The equation can be simplified to,

$$\tilde{\mathsf{T}}^{(I)(J)}_{ijk} = \mathsf{T}^{(I)(J)}_{ijk} + \sigma_{ijk} N_{ijk}(\mu = 0, \sigma = \alpha), \quad \forall I, J, \quad I < J\,. \tag{4.3}$$

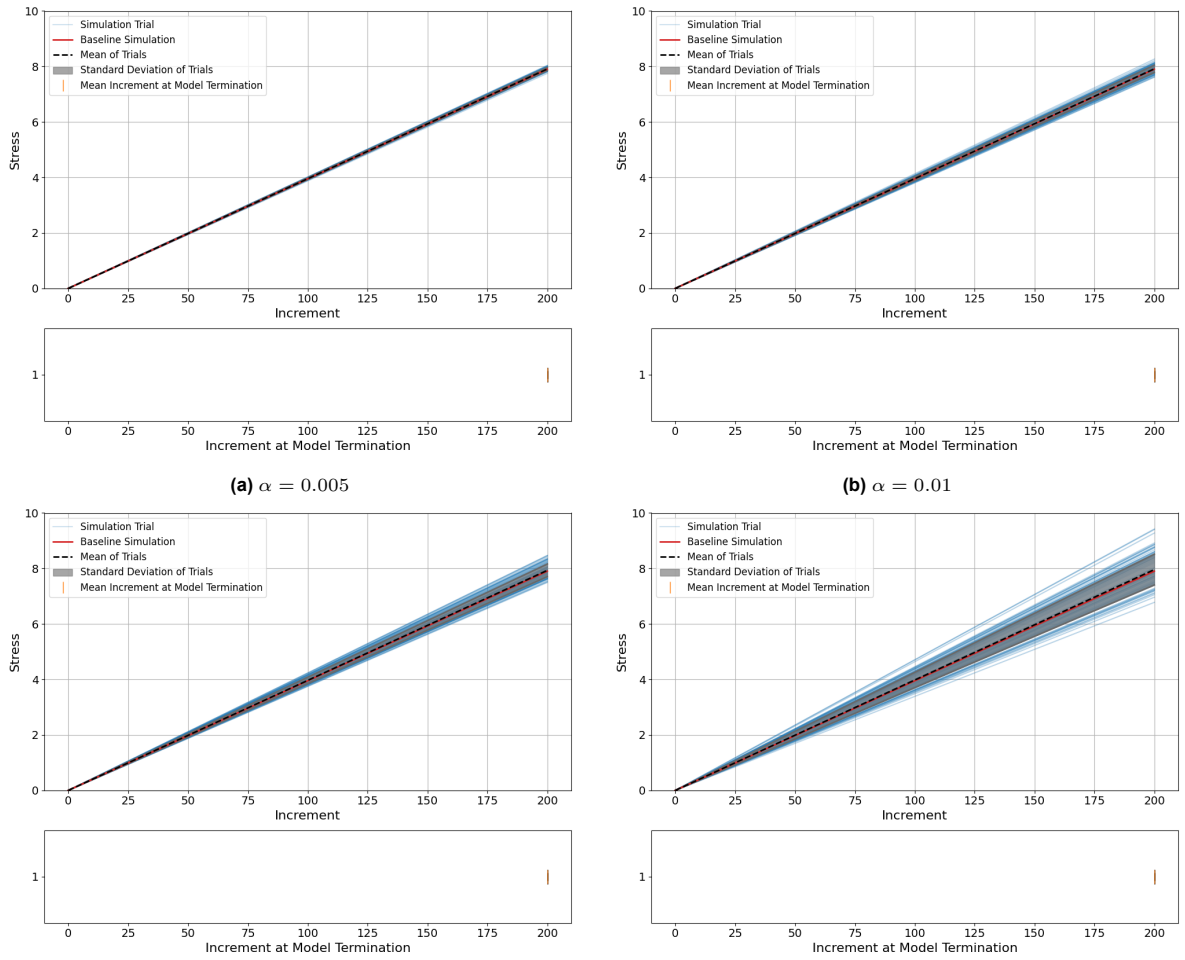
Note that the lower off-diagonal terms are computed using the symmetry condition (Equation 2.21) taking the noisy upper off-diagonal values as inputs.

A hundred simulation trials are performed with different seeds of noise. By injecting noise into the solution in this manner, the error given by the surrogate model can be emulated.

4.3.1. Sensitivity to noise

The first sensitivity study considers the homogenized response for different noise values. This is evaluated using plots of the homogenized stress-strain response in the direction of loading. The number of clusters is kept constant at 18 and the noise parameter α is considered at points equal to 0.005, 0.01, 0.02, 0.04.

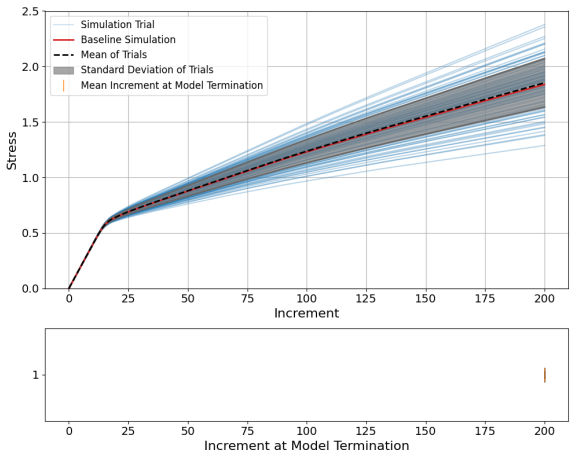
Additionally, three test cases correspond to a higher degree of physical complexity. First and most simple is the elastic case where all materials are assigned elastic properties. Next is the plastic case where materials are assigned either elastic or elasto-plastic properties. CRATE uses the J2 plasticity model for computing plastic responses. Finally, the most complex is plasticity with self-consistent scheme (SCS), or plastic+SCS case for short, which is the same as the plastic case but with SCS. This is computationally challenging since SCS requires an optimization step to be performed in the online stage.



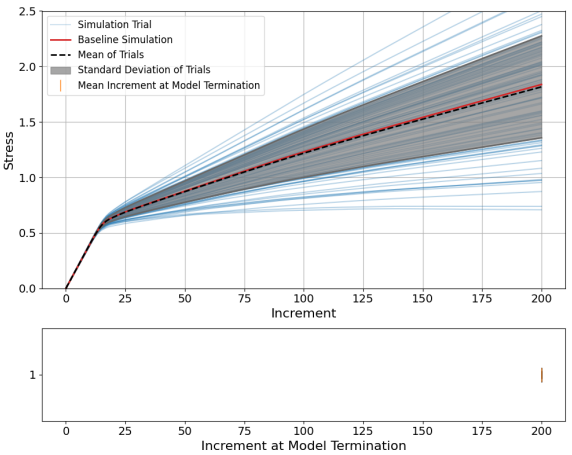
(a) $\alpha = 0.005$ **(b)** $\alpha = 0.01$

(c) $\alpha = 0.02$ **(d)** $\alpha = 0.04$

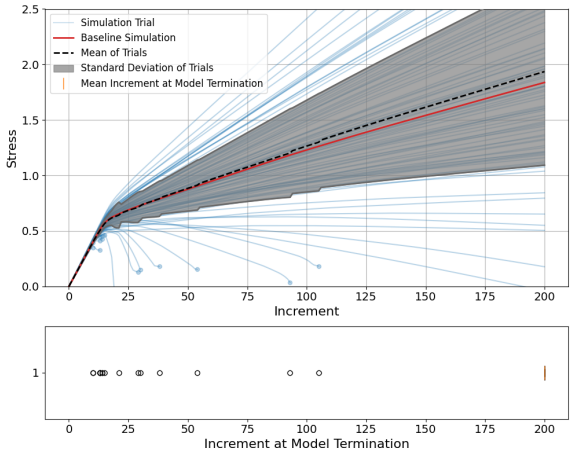
Figure 4.7: Sensitivity to noise in elastic case. Homogenized response (upper figures), box plot of model termination (lower figures).



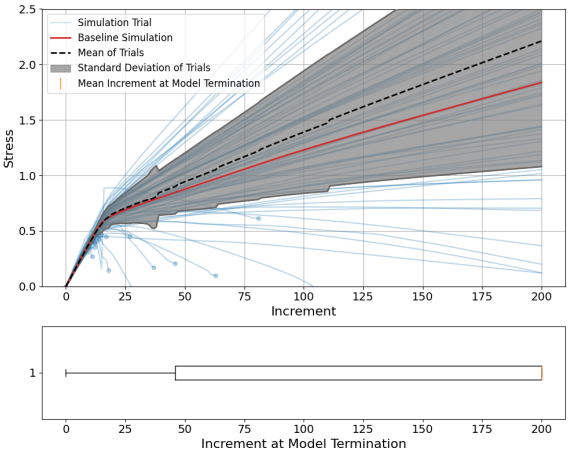
(a) $\alpha = 0.005$



(b) $\alpha = 0.01$

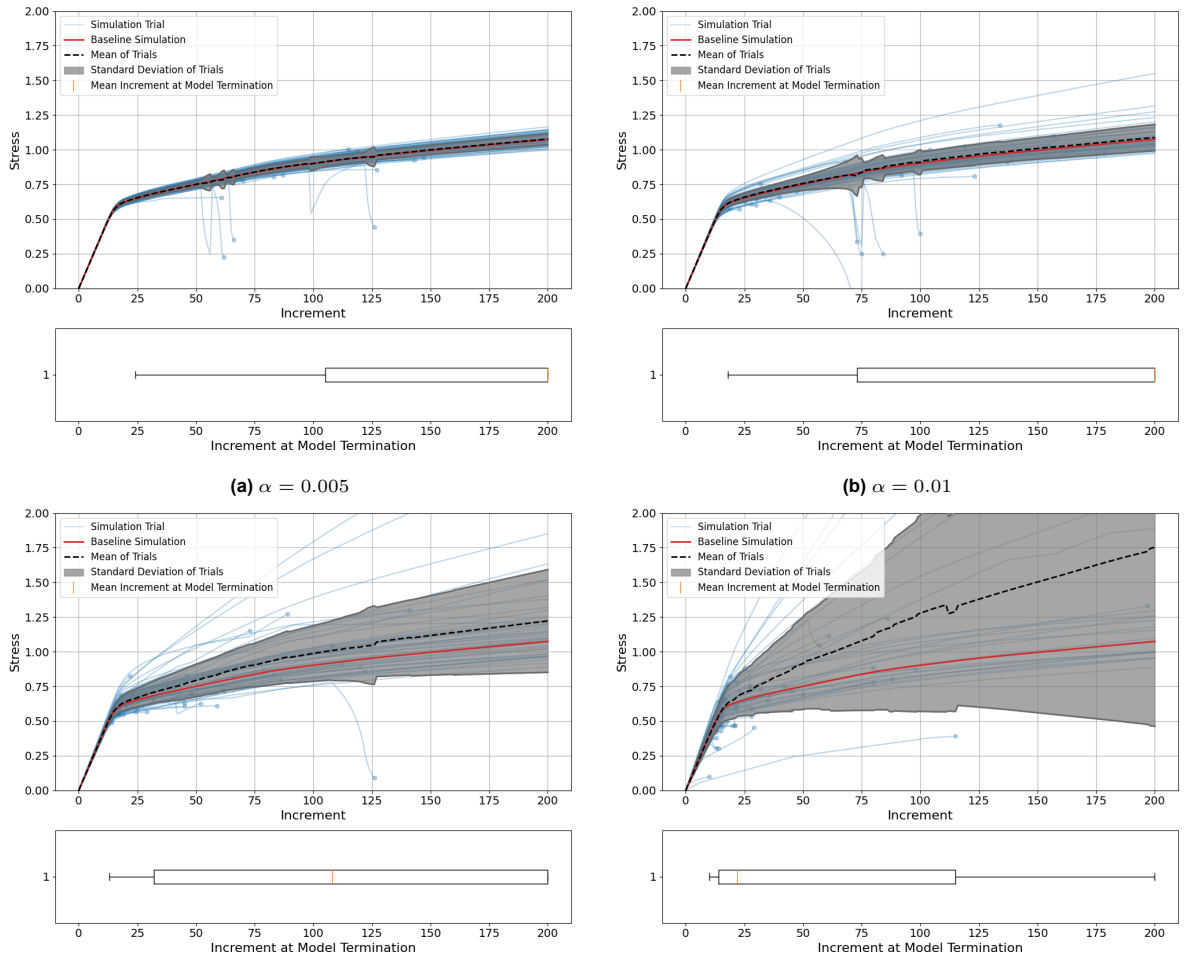


(c) $\alpha = 0.02$



(d) $\alpha = 0.04$

Figure 4.8: Sensitivity to noise in plastic case. Homogenized response (upper figures), box plot of model termination (lower figures).



(a) $\alpha = 0.005$ **(b)** $\alpha = 0.01$

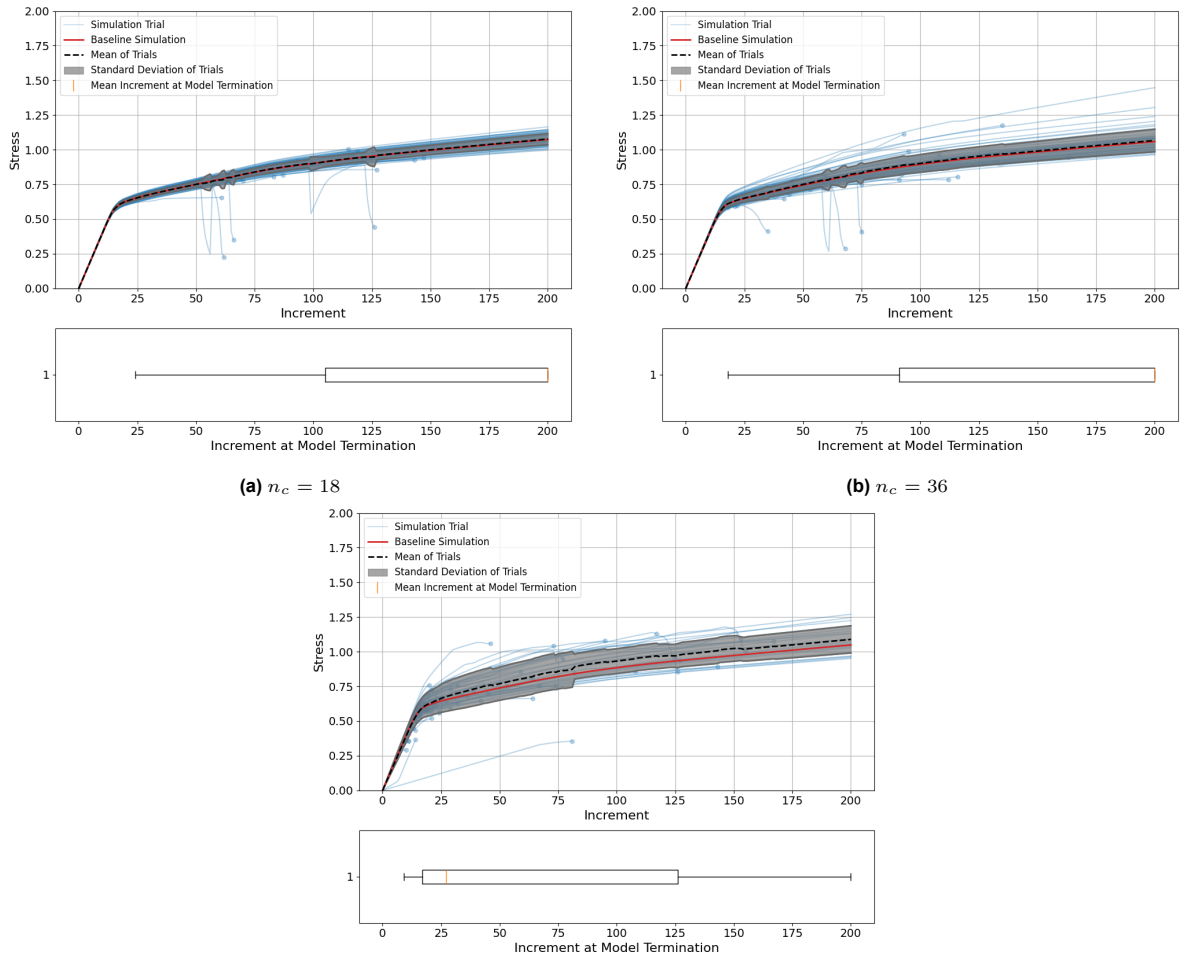
(c) $\alpha = 0.02$ **(d)** $\alpha = 0.04$

Figure 4.9: Sensitivity to noise in plastic+SCS case. Homogenized response (upper figures), box plot of model termination (lower figures).

Several statements can be made regarding the different cases based on the simulations performed. In the elastic case (Figure 4.7), increasing the noise changes the solution accuracy linearly and does not affect convergence. In the plastic case (Figure 4.8), the solution accuracy is degraded in a linear manner for small noise values and in a nonlinear manner for larger noise values. The model's ability to converge can be affected by large noise values. The plastic+SCS case shows that for smaller noise values, the solution accuracy is, surprisingly, better than that of the plastic case. This can be attributed to the SCS having an optimization step that can potentially self-correct for errors. However, it is also observed that, in this case, the model is very sensitive to noise when it comes to convergence, with many simulations unable to converge even for small noise values. For $\alpha = 0.04$, it is even observed that, on average, the models terminate at the yielding point.

4.3.2. Sensitivity to number of clusters

Next, since the surrogate model will be tested for different numbers of clusters and should preferably be scalable to large numbers of clusters, the sensitivity is evaluated for varying numbers of clusters. It should be noted that σ_{ijk} changes for different values of n_c and is, therefore, an uncontrollable variable that can affect the simulation results.



(a) $n_c = 18$

(b) $n_c = 36$

(c) $n_c = 72$

Figure 4.10: Sensitivity to number of clusters in plastic-SCS case. Stress-strain response (upper figures), box plot of model termination (lower figures).

Based on Figure 4.10, it can be observed that the solution accuracy degrades with increasing numbers of clusters. This is attributed to the fact that the number of off-diagonal terms (which are approximate) grows faster than the number of diagonal terms (which are exact), leading to a more significant compounding error in the solution.

4.4. Methods

Several metrics are used to evaluate and compare the surrogate model with the baseline model. Naturally, since the two primary goals of this study are to find a method that is faster than the baseline while still maintaining reasonable accuracy, the metrics must reflect these goals. Additionally, models with different parameters are used to assess the performance given different physics and test cases.

The first metric used to evaluate the computation speed is the number of FLOPs. This represents the theoretical complexity of each algorithm given a set number of inputs. The resolution must be fixed to compute this metric. These inputs are chosen in such a way that they are representative of the resolution that a user of SCA could encounter. FLOPs are selected over other theoretical complexity metrics because the number of FLOPs of most popular architectures can be readily found in ML literature. Additionally, FLOPs are simple to compute by hand.

The second metric used to evaluate computation speed is the wall clock time. Naturally, as highlighted in section 2.4, this metric is highly dependent on the system used. However, it is still indicative of the computational speed that a potential user of an algorithm can expect. To keep conditions comparable, the surrogate and baseline are evaluated on an Intel Core i7-6700K, 4-core CPU and averaged

over three evaluations. The wall clock time of the baseline is given by $t_{baseline}$ and the difference in wall clock time between the surrogate and baseline is reported as a speedup factor,

$$\text{Speedup} = \frac{t_{baseline}}{t_{surrogate}} \,. \tag{4.4}$$

Additionally, metrics related to the model's training are provided. The total training time of the model and the number of training epochs are shown. Models with documented training time are trained on an NVIDIA H100-NVL GPU using the OSCAR Supercomputer at the Center for Computation and Visualization, Brown University.

The first metric used for evaluating the accuracy is the Mean Absolute Error (MAE) of the CIT prediction,

$$\text{CIT MAE} = \frac{1}{n_c^2 \cdot n_{comp}^2 \cdot 3} \sum_{I=1}^{n_c} \sum_{J=1}^{n_c} \sum_{i=1}^{n_{comp}} \sum_{j=1}^{n_{comp}} \sum_{k=1}^{3} (\mathbf{T}_{ijk}^{(I)(J)} - \tilde{\mathbf{T}}_{ijk}^{(I)(J)}) \,, \tag{4.5}$$

is essentially the averaged error over every tensor element. It should be noted that $n_{comp} = 3$ when a 2D simulation is performed under infinitesimal strains. This metric evaluates how well a model has trained for its objective. Additionally, this is supported by the log loss diagrams, which give a qualitative overview of the model training process.

The second metric used for evaluating the accuracy is the Mean Absolute Percentage Error (MAPE) in the stress at every increment. It is given by,

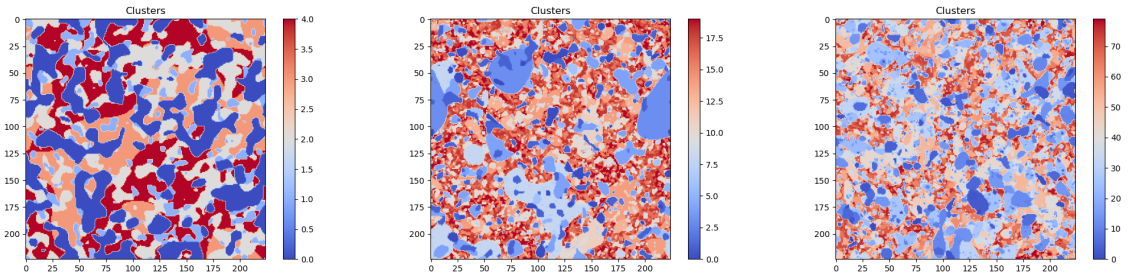
$$\text{Stress MAPE} = \frac{1}{n_{increments}} \sum_{i=1}^{n_{increments}} |\frac{\sigma_i - \tilde{\sigma_i}}{\sigma_i}| \,, \tag{4.6}$$

where $n_{increments}$ is the total number of increments in a simulation, σ_i is the stress at increment i given by the baseline model and $\tilde{\sigma_i}$ is the stress at increment i given by the surrogate model. These values are given for a 2D uniaxial tension benchmark and pure shear benchmark, with the stress being reported in the respective strain loading direction of the benchmark.

In addition, relevant plots are provided to evaluate the model's accuracy. A plot comparing the L2 norm of the local tensor between the baseline and surrogate is shown. This gives more context to the CIT MAE value since it visualizes every local tensor rather than computing a single value for the global tensor. Furthermore, stress-strain plots are shown to give more context to the Stress MAPE values and verify that the final result behaves as expected.

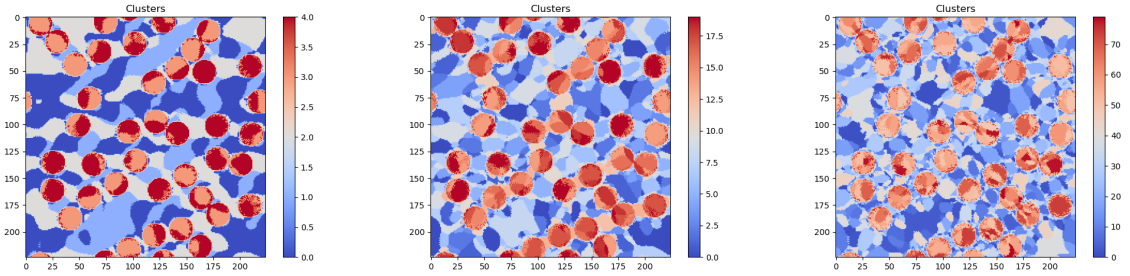
SCA simulations are performed using several parameters. Firstly, similarly to section 4.3, three test cases are considered in increasing levels of physical complexity: elastic, plastic and plastic+SCS. Secondly, the models are evaluated for a range of clusters: 5, 20 and 80. Additionally, two sample groups are considered: in-distribution and out-of-distribution. In this case, distribution relates to the range of possible clustered microstructures that can be generated using the synthetic generation technique highlighted in chapter 3. An in-distribution sample is generated using the same parameters as the training dataset but is a sample that the model has not seen during training. By design, it is provided in an already clustered state as seen in Figure 4.11. An out-of-distribution sample is generated using a representative long fiber composite microstructure that undergoes a clustering procedure in CRATE to generate the clustered microstructure seen in Figure 4.12. An in-distribution sample shows how accurate a model is at predicting with samples similar to the training data and reflects how well it has trained. An out-of-distribution sample shows how well a model can generalize to more realistic use cases.

To evaluate models fairly, the CIT MAE and Stress MAPE values are averaged for 5 in-distribution samples and 5 out-of-distribution samples. The accompanying L2 norm plots and stress-strain plots are given for a single sample. To enable comparison between different studies (section 4.5, section 4.6, section 4.8), these plots are obtained from the same sample across the studies.



(a) $n_c = 5$ **(b)** $n_c = 20$ **(c)** $n_c = 80$

Figure 4.11: Examples of in-distribution clustered microstructures.



(a) $n_c = 5$ **(b)** $n_c = 20$ **(c)** $n_c = 80$

Figure 4.12: Examples of out-of-distribution clustered microstructures.

4.5. Standard surrogate model

First, it is important to establish if a data-driven surrogate model can be trained to predict the CIT accurately enough so that it can be used in SCA.

4.5.1. Experimental Setup

ResNet-18, which is an off-the-shelf model discussed in section 2.2, is selected for the experiment. This choice comes from the model having high accuracy in the ImageNet competition and applicability to other problems conducted in solid mechanics. It requires 1.79 GFLOPs to compute a local tensor, which is significantly higher than the 0.015 GFLOPs needed for the baseline model. Nevertheless, this experiment aims to demonstrate the feasibility of using an data-driven model to make predictions of the CIT.

The model has also been slightly modified from their original implementation. The first layer is modified to accept two channels (corresponding to the two clusters) instead of three channels. Additionally, it is padded using circular padding instead of zero padding to reflect the periodic boundary conditions of the problem. The last layer is modified to regress 27 values (corresponding to each element in the local CIT) instead of 1000 values. The modified architecture is shown in Table 4.1 in the same format presented by He et al. [21].

Layer Name	Block
conv1	7x7, 64, stride 2
pool	3x3 max pool, stride 2
conv2_x	$\begin{bmatrix} \text{3x3, 64} \\ \text{3x3, 64} \end{bmatrix} \times 2$
conv3_x	$\begin{bmatrix} \text{3x3, 128} \\ \text{3x3, 128} \end{bmatrix} \times 2$
conv4_x	$\begin{bmatrix} \text{3x3, 256} \\ \text{3x3, 256} \end{bmatrix} \times 2$
conv5_x	$\begin{bmatrix} \text{3x3, 512} \\ \text{3x3, 512} \end{bmatrix} \times 2$
average pool	
27-d fc	

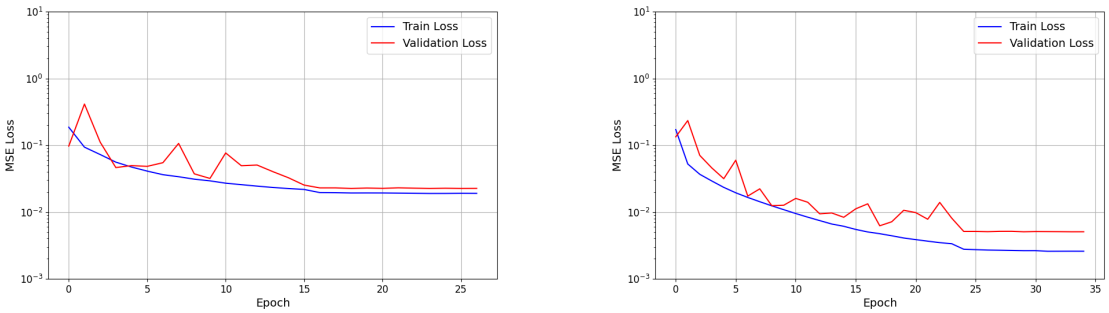
Table 4.1: Modified ResNet-18 architecture.

The dataset comprises 100k clustered microstructures and 750k data points (cluster pairs), split into 80% training data and 20% validation data and has a resolution of 224×224 pixels. Some parameters are also set for the clustered microstructure generation algorithm. Each clustered microstructure has between 5 and 80 clusters, corresponding to the range of clusters being evaluated in the subsequent SCA simulations. The roughness distributions are sampled from a log distribution with a mean of 2 and a standard deviation of 1. These values are obtained by trial and error through visual comparison with clustered microstructures given in literature.

4.5.2. Training

Several training parameters are set for the training strategy. Training is performed for 20 hours or 50 epochs or until an early stopping condition is reached, whichever comes first. The early stopping condition is set to a 10% improvement in validation loss with a patience of 10 epochs. The learning rate is initialized with a value of 0.01 and an exponentially decayed function with $\gamma = 0.9$ is applied. A step decay with $\gamma = 0.1$ is applied at a patience of 5 epochs within the early stopping condition. MSE is used as the loss function and the batch size is set to 256.

ResNet-18 converges to a lower validation loss using Adam (as seen in Figure 4.13), and as such, the weights of this model are used for the standard surrogate. This model required 2h:18m:21s and 34 epochs to converge.



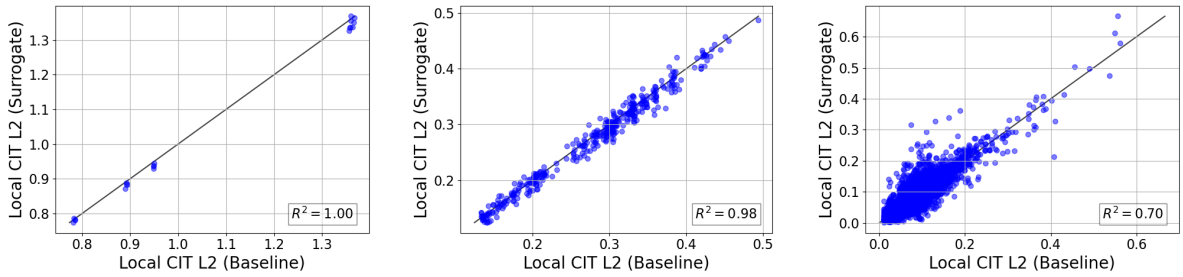
(a) SGD **(b)** Adam

Figure 4.13: ResNet-18 log loss diagrams using different optimizers.

4.5.3. In-distribution prediction results

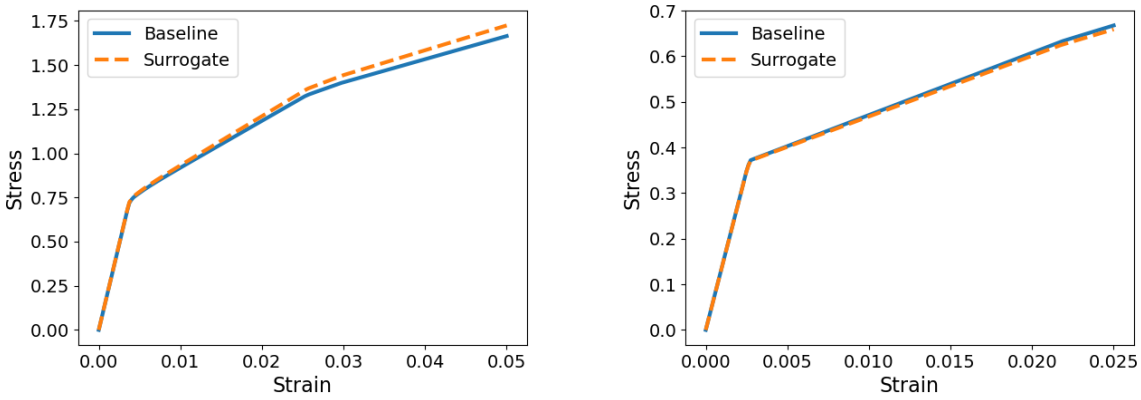
n_c	Physics	CIT MAE	$t_{baseline}$	Speedup	Stress MAPE (uniaxial tension)	Stress MAPE (pure shear)
5	elastic	0.059	2.60E-01s	0.53	0.13%	0.14%
	plastic				0.48%	0.46%
	plastic + SCS				0.79%	0.64%
20	elastic	0.058	1.71E+00s	0.32	0.03%	0.17%
	plastic				0.18%	0.54%
	plastic + SCS				0.16%	0.30%
80	elastic	0.099	2.00E+01s	0.26	0.55%	0.53%
	plastic				1.91%	2.33%
	plastic + SCS				1.90%	1.62%

Table 4.2: Model summary ResNet-18 (in-distribution).



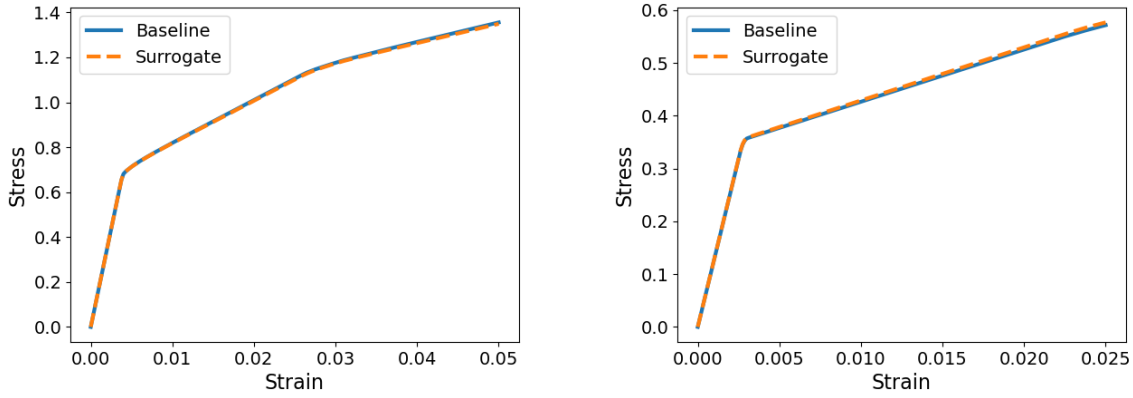
(a) $n_c = 5$ **(b)** $n_c = 20$ **(c)** $n_c = 80$

Figure 4.14: L2 Ground Truth vs Prediction (in-distribution).



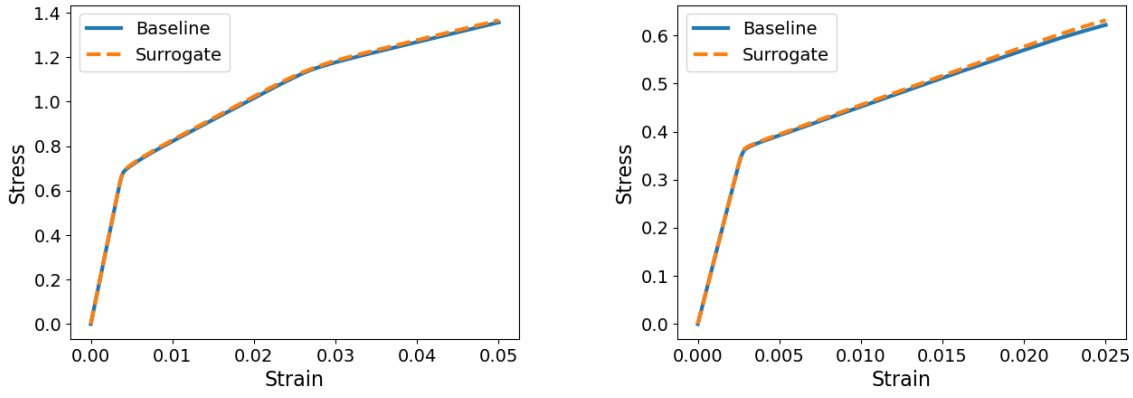
(a) Uniaxial tension **(b)** Pure Shear

Figure 4.15: Stress-Strain response (plastic + SCS, in-distribution, $n_c = 5$).



(a) Uniaxial tension **(b)** Pure Shear

Figure 4.16: Stress-Strain response (plastic + SCS, in-distribution, $n_c = 20$).



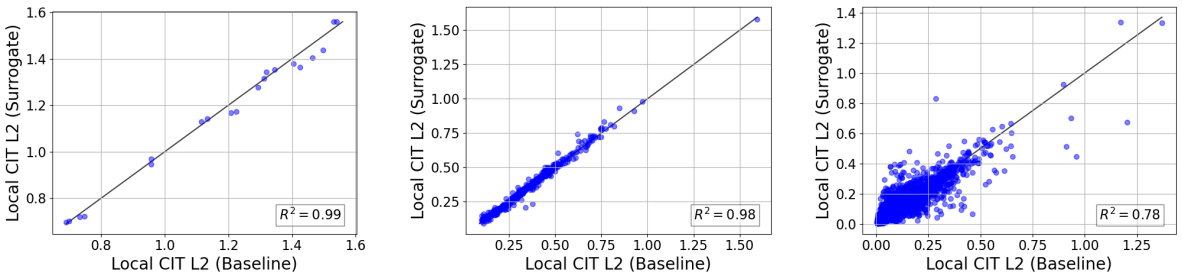
(a) Uniaxial tension **(b)** Pure Shear

Figure 4.17: Stress-Strain response (plastic + SCS, in-distribution, $n_c = 80$).

4.5.4. Out-of-distribution prediction results

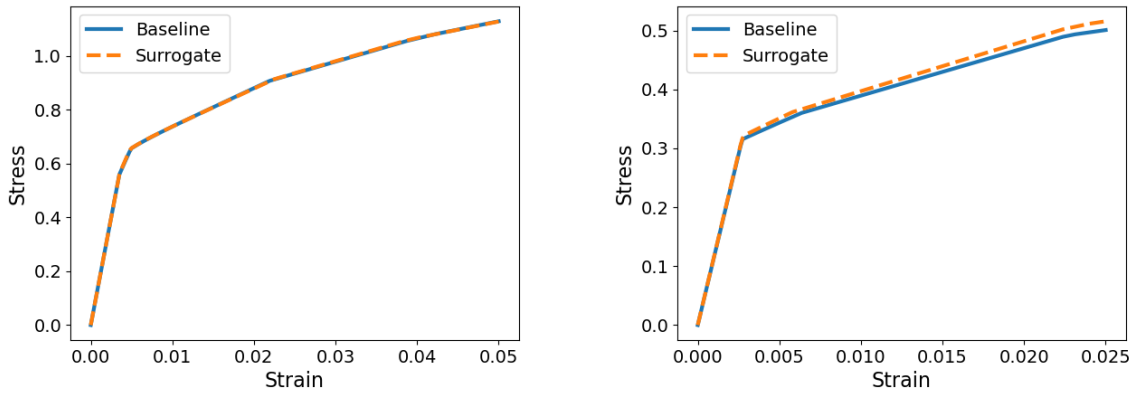
n_c	Physics	CIT MAE	$t_{baseline}$	Speedup	Stress MAPE (uniaxial tension)	Stress MAPE (pure shear)
5	elastic	0.137	2.70E-01s	0.62	0.26%	0.44%
	plastic				0.63%	2.95%
	plastic + SCS				0.41%	1.77%
20	elastic	0.107	1.70E+00s	0.32	0.39%	0.27%
	plastic				2.44%	3.96%
	plastic + SCS				0.54%	2.08%
80	elastic	0.153	1.90E+01s	0.25	0.41%	1.45%
	plastic				2.82%	11.16%
	plastic + SCS				0.62%	6.33%

Table 4.3: Model summary ResNet-18 (out-of-distribution).



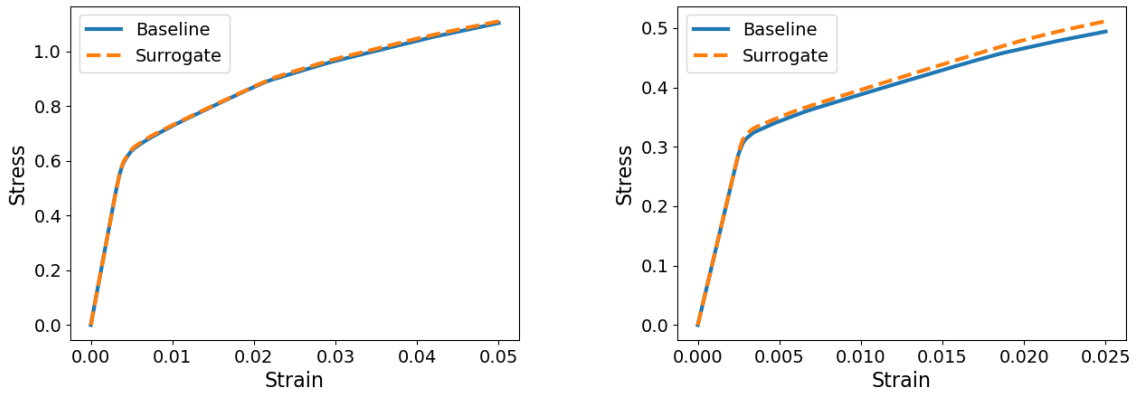
(a) $n_c = 5$ **(b)** $n_c = 20$ **(c)** $n_c = 80$

Figure 4.18: L2 Ground Truth vs Prediction (out-of-distribution).



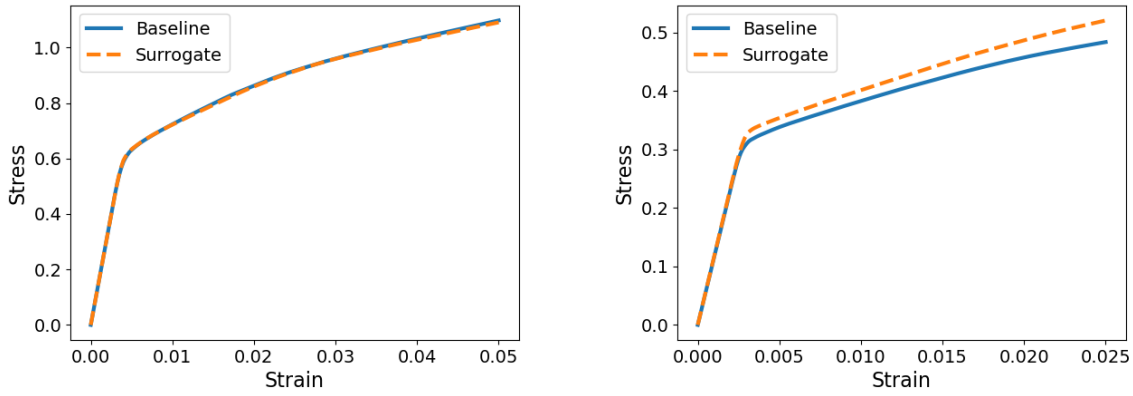
(a) Uniaxial tension **(b)** Pure Shear

Figure 4.19: Stress-Strain response (plastic + SCS, out-of-distribution, $n_c = 5$).



(a) Uniaxial tension **(b)** Pure Shear

Figure 4.20: Stress-Strain response (plastic + SCS, out-of-distribution, $n_c = 20$).



(a) Uniaxial tension **(b)** Pure Shear

Figure 4.21: Stress-Strain response (plastic + SCS, out-of-distribution, $n_c = 80$).

4.5.5. Discussion

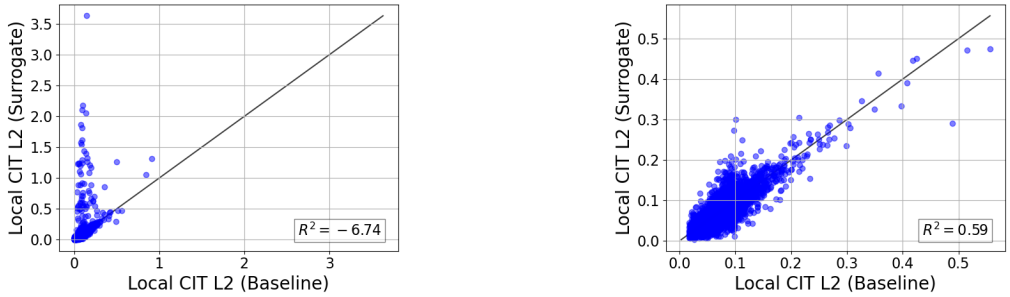
With the results of the surrogate shown for in-distribution and out-of-distribution samples in Table 4.2 and Table 4.3, respectively, a number of observations can be made based on the data as well as the supporting figures.

Remarkably, the surrogate yields highly accurate homogenized stress-strain responses. For the in-distribution results, it is within 2% and 3% for the uniaxial tension and pure shear, respectively. The out-of-distribution results give slightly poorer predictions that are at least within 3% and 12% for the uniaxial tension and pure shear, respectively. As expected, based on the sensitivity analysis, the plastic+SCS cases generally give more accurate solutions than the plastic cases, especially when the number of clusters is increased. Strikingly, the out-of-distribution uniaxial, plastic+SCS case gives almost exact predictions (within 1%). Nevertheless, the results are less impressive in the pure shear case, where the surrogate cannot predict with the same degree of accuracy. A potential reason why the pure shear predictions are less accurate than the uniaxial tension cases could be the dataset's scaling and the local tensor elements underlying distributions. It can be seen in Figure 4.4 that indices corresponding to shear ($i = 3$ and $j = 3$) have different underlying distributions when compared to their normal counterparts and, therefore, a different architecture or scaling may be needed to predict these components more accurately.

Nevertheless, as expected based on the number of GFLOPs, the surrogate cannot offer a speedup compared to the baseline method. Taking a closer look at the speedup values, it can be observed that they change with the number of clusters. This is an expected result since the ratio of diagonal to off-diagonal terms in the global tensor changes with the number of clusters.

4.5.6. Challenges and Limitations

When testing in-distribution samples, it is observed that the model cannot make accurate predictions for ones containing very small clusters. This can be seen in Figure 4.22a, where the surrogate significantly overpredicts the local CIT L2 norms when small clusters are present. Due to this over-prediction, the simulations failed to converge in the plastic+SCS case. Note that this sample is not one of the five used during model evaluation. After filtering for clusters with less than 10 pixels in total (by merging these small clusters into larger clusters), it can be seen in Figure 4.22b that the R^2 values improve significantly. Note that although only a single cluster is removed in the clustered microstructure, multiple points are filtered out in Figure 4.22 due to the cluster being permuted with all other clusters in the clustered microstructure. No convergence issues are observed after filtering. Some potential reasons for this limitation could be due to a more challenging physical interaction or higher required precision in the small cluster regime. Additionally, the limitation could arise due to the dataset used to train the surrogate, which contained less than 0.5% of data points with clusters of size 10 pixels or smaller. While this limitation isn't observed in any of the out-of-distribution cases during testing, as none contain small enough clusters, it could become an issue as the number of clusters is increased beyond the upper limit considered in this study ($n_c = 80$).



(a) Unfiltered for small clusters

(b) Filtered for small clusters

Figure 4.22: L2 Ground Truth vs Prediction (in-distribution sample with small cluster).

4.6. Unbalanced dataset study

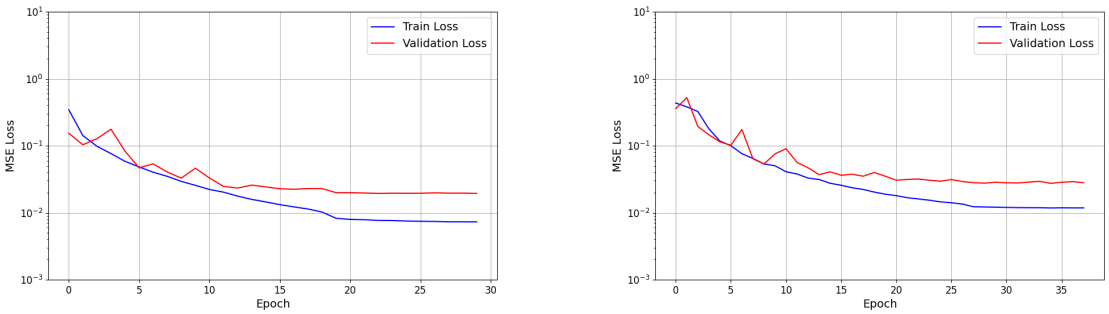
To attempt to resolve the limitation caused by small clusters, a fundamental assumption made during dataset generation is challenged: the need for a balanced dataset. By training on an unbalanced dataset, more emphasis could be given to smaller clusters, increasing the accuracy in the regime.

4.6.1. Experimental setup

The standard surrogate is trained on two additional unbalanced datasets and compared to the results of the balanced dataset. The unbalanced datasets contain the same number of data points as the balanced dataset but scale the input from the CIT differently (in a quadratic or linear manner). The reader is reminded that the dataset for the standard surrogate is made by first generating several clustered microstructures and then splitting the global tensor of each into local tensors, with each local tensor corresponding to a single data point. In the balanced case, described in chapter 2, the global tensor is truncated such that each clustered microstructure contributes the same number of data points irrespective of the number of clusters. This can be summarized as $d \propto c$, where d is the number of data points contributed by each clustered microstructure and c is a constant. The parameter $n_{c,cut}$ (introduced in section 3.4) is set to the minimum number of clusters in the range considered ($n_{c,cut} = 5$). The quadratic dataset scales proportionally to the size of the CIT, i.e. $d \propto n_c^2$. In this case, $n_{c,cut} = n_c$. The linear dataset truncates the CIT such that $d \propto n_c$. To mimic a linear relationship, $n_{c,cut} = \text{int}(\sqrt{n_c})$, where int is a function that rounds the value to the nearest integer value.

4.6.2. Training

The same training strategy is used as described in section 4.5. The model architecture and weights from section 4.5 are reused for the balanced dataset case. The log loss plots of the two new surrogates trained on unbalanced datasets (linear and quadratic) can be seen in Figure 4.23.



(a) Linear

(b) Quadratic

Figure 4.23: Log loss diagrams for dataset study.

4.6.3. In-distribution prediction results

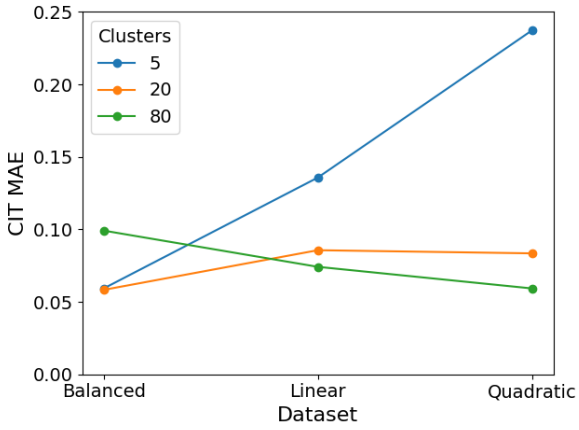
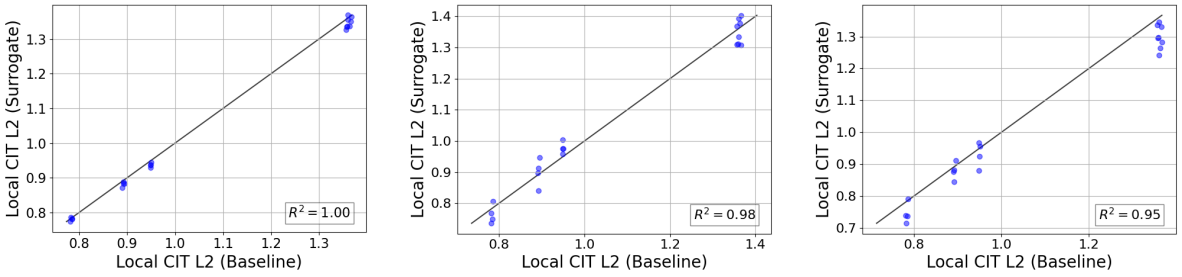


Figure 4.24: CIT MAE for dataset study (in-distribution).

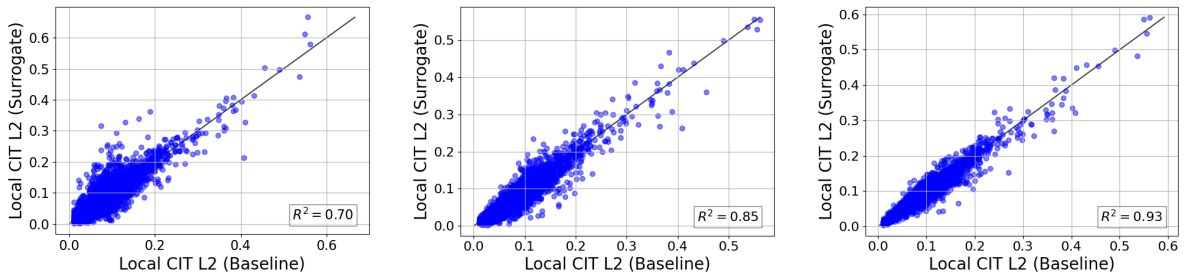
Clusters	Physics	Stress MAPE (uniaxial tension)			Stress MAPE (pure shear)		
		Balanced	Linear	Quadratic	Balanced	Linear	Quadratic
5	elastic	0.13%	0.59%	0.59%	0.14%	0.45%	0.71%
	plastic	0.48%	2.07%	1.97%	0.46%	1.71%	2.21%
	plastic + SCS	0.79%	3.31%	2.98%	0.64%	1.61%	2.72%
20	elastic	0.03%	0.13%	0.08%	0.17%	0.27%	0.12%
	plastic	0.18%	0.86%	0.32%	0.54%	1.27%	0.46%
	plastic + SCS	0.16%	0.52%	0.37%	0.30%	0.60%	0.28%
80	elastic	0.55%	0.43%	0.44%	0.53%	0.22%	0.42%
	plastic	1.91%	2.38%	2.33%	2.33%	1.07%	1.91%
	plastic + SCS	1.90%	2.14%	2.11%	1.62%	0.92%	1.56%

Table 4.4: Model summary dataset study (in-distribution).



(a) Balanced dataset **(b)** Linear dataset **(c)** Quadratic dataset

Figure 4.25: L2 Ground Truth vs Prediction (in-distribution, $n_c = 5$).



(a) Balanced dataset **(b)** Linear dataset **(c)** Quadratic dataset

Figure 4.26: L2 Ground Truth vs Prediction (in-distribution, $n_c = 80$).

4.6.4. Out-of-distribution prediction results

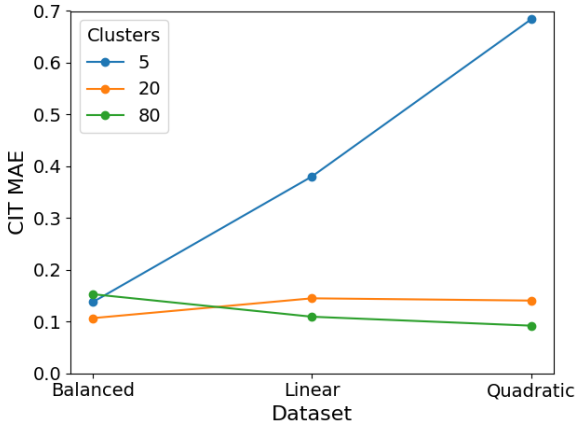
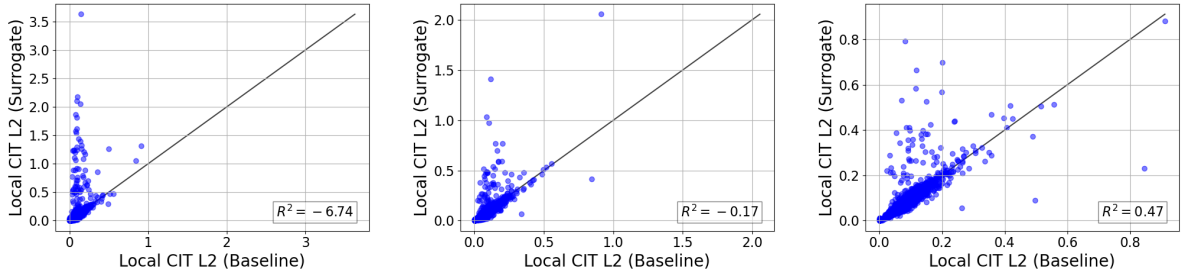


Figure 4.27: CIT MAE for dataset study (out-of-distribution).

Clusters	Physics	Stress MAPE (uniaxial tension)			Stress MAPE (pure shear)		
		Balanced	Linear	Quadratic	Balanced	Linear	Quadratic
5	elastic	0.26%	0.28%	0.12%	0.44%	0.55%	1.28%
	plastic	0.63%	3.21%	3.87%	2.95%	5.16%	10.45%
	plastic + SCS	0.41%	2.84%	2.63%	1.77%	3.02%	5.73%
20	elastic	0.39%	0.17%	0.27%	0.27%	0.52%	0.66%
	plastic	2.44%	9.21%	9.61%	3.96%	9.38%	10.06%
	plastic + SCS	0.54%	2.78%	2.75%	2.08%	3.66%	4.34%
80	elastic	0.41%	0.29%	0.17%	1.45%	1.09%	1.02%
	plastic	2.82%	18.80%	17.59%	11.16%	17.80%	15.66%
	plastic + SCS	0.62%	3.62%	3.73%	6.33%	6.05%	5.55%

Table 4.5: Model summary dataset study (out-of-distribution).

4.6.5. In-distribution prediction with small clusters



(a) Balanced dataset **(b)** Linear dataset **(c)** Quadratic dataset

Figure 4.28: L2 Ground Truth vs Prediction (in-distribution with small cluster).

4.6.6. Discussion

Several observations can be made about the surrogates trained on the three datasets. The quadratic and linear datasets are both able to improve the R^2 results in the outlier case (Figure 4.28). However, the model still exhibits abnormal behavior by over-predicting the CIT values. As expected, the R^2 values for samples with a high number of clusters are improved (Figure 4.26), but predictions in samples with small numbers of clusters are degraded (Figure 4.25) when going from balanced to linear to quadratic. The same relationship is observed in the CIT MAE values (Figure 4.24 and Figure 4.27). No significant change is observed in the Stress MAPE values of the in-distribution case (Table 4.4). Interestingly, in the out-of-distribution case, the Stress MAPE values more than doubled for most experiments when comparing the linear and quadratic datasets against the balanced dataset. This result shows that a balanced dataset is needed for the surrogate to generalize.

4.7. Sensitivity to dataset size

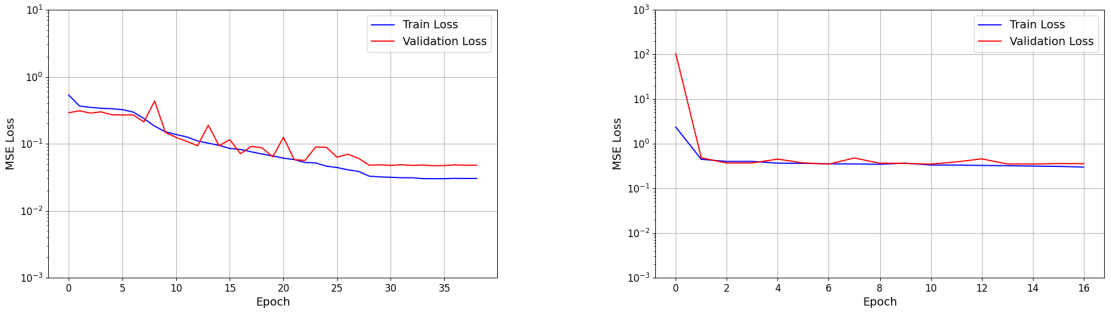
To justify the use of a large dataset as well as the proposed clustered microstructure generator method, the surrogate's sensitivity is tested for different dataset sizes.

4.7.1. Experimental setup

The ResNet-18 model (shown in Table 4.1) is used again for this experiment. The model is trained for datasets with 100k, 10k and 1k clustered microstructures. The smaller datasets are formed by truncating the larger dataset, i.e. they are a subset of the 100k dataset.

4.7.2. Training

The same training strategy is used as described in section 4.5. The model architecture and weights from section 4.5 are reused for the 100k dataset case. The other two models are trained using the Adam optimizer (to enable a fair comparison). The log loss plots can be seen in Figure 4.29.



(a) 10k **(b)** 1k

Figure 4.29: Log loss diagrams for truncated dataset sizes.

4.7.3. In-distribution prediction results

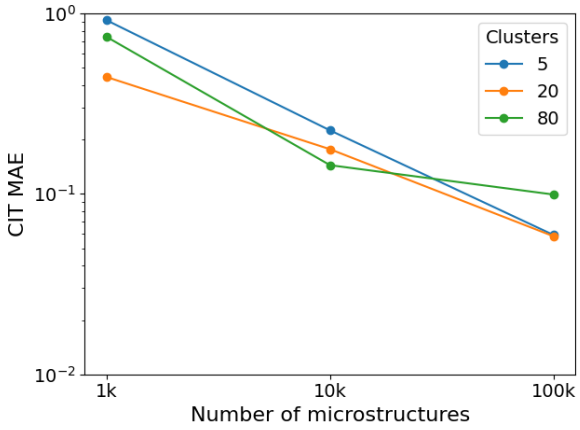


Figure 4.30: CIT MAE sensitivity to dataset size (in-distribution).

Clusters	Physics	Stress MAPE (uniaxial tension)			Stress MAPE (pure shear)		
		100k	10k	1k	100k	10k	1k
5	elastic	0.13%	0.58%	2.77%	0.14%	0.62%	3.07%
	plastic	0.48%	2.09%	9.05%	0.46%	1.87%	8.62%
	plastic SCS	0.79%	2.21%	14.89%	0.64%	2.23%	10.63%
20	elastic	0.03%	0.14%	0.33%	0.17%	0.48%	1.09%
	plastic	0.18%	1.30%	3.31%	0.54%	2.19%	6.03%
	plastic SCS	0.16%	0.66%	NC	0.30%	1.06%	2.40%
80	elastic	0.55%	0.39%	2.23%	0.53%	0.37%	1.62%
	plastic	1.91%	2.91%	NC	2.33%	0.78%	14.64%
	plastic SCS	1.90%	1.24%	NC	1.62%	1.18%	NC

Table 4.6: Stress MAPE sensitivity to dataset size (in-distribution). NC refers to one or more of the simulations not converging to a solution

4.7.4. Out-of-distribution prediction results

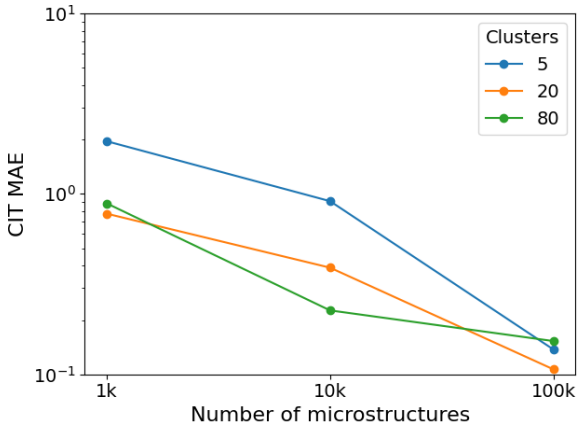


Figure 4.31: CIT MAE sensitivity to dataset size (out-of-distribution).

Clusters	Physics	Stress MAPE (uniaxial tension)			Stress MAPE (pure shear)		
		100k	10k	1k	100k	10k	1k
5	elastic	0.26%	0.25%	1.60%	0.44%	1.06%	2.72%
	plastic	0.63%	4.80%	16.61%	2.95%	10.09%	17.06%
	plastic SCS	0.41%	3.03%	8.43%	1.77%	5.62%	8.77%
20	elastic	0.39%	0.64%	2.12%	0.27%	1.12%	3.65%
	plastic	2.44%	21.96%	34.45%	3.96%	16.55%	30.49%
	plastic SCS	0.54%	5.33%	9.40%	2.08%	6.22%	11.60%
80	elastic	0.41%	0.32%	2.09%	1.45%	0.29%	3.39%
	plastic	2.82%	30.92%	NC	11.16%	19.71%	55.20%
	plastic SCS	0.62%	NC	NC	6.33%	4.71%	NC

Table 4.7: Stress MAPE sensitivity to dataset size (out-of-distribution). NC refers to one or more of the simulations not converging to a solution.

4.7.5. Discussion

As expected, the model's accuracy increases as a function of dataset size. This is evident by the decrease in CIT MAE (seen in Figure 4.30 and Figure 4.31) as well as Stress MAPE (seen in Table 4.6 and Table 4.7) as dataset size increases. It is particularly noteworthy by how much the stress MAPE is affected in out-of-distribution results. For example, it more than doubles in the plastic case for every increment of dataset size. The effect on the in-distribution results is not as severe. This suggests that dataset size has a significant influence on the generalizability of the surrogate.

Naturally, at large enough dataset sizes, the improvement gains will be marginal and likely plateau. However, these results suggest that this plateau cannot be observed within the range of tested dataset sizes for the given model. This motivates the need for large dataset sizes (in the range of 100k clustered microstructures or more) when training this surrogate on the given cluster range.

4.8. Accelerated surrogate model

With the surrogate modeling approach being shown to be sufficiently accurate, the next challenge is making it faster than the baseline.

4.8.1. Experimental Setup

For this experiment, a modified ResNet-18 architecture (called ResNet-lite) is created to test modeling with fewer parameters. This model requires only 0.012 GFLOPs, which is less than the 0.015 GFLOPs needed for the baseline model.

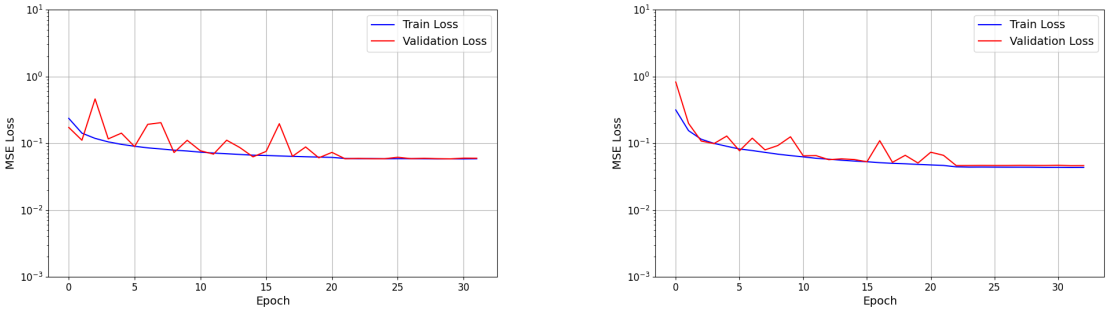
ResNet-lite is based on the ResNet-18 architecture with the only exception being that the first layer has 4 convolutional filters instead 64 (i.e. the output of this layer has 4 channels instead of 64). In the down-sampled convolutional layers the same widening ratio (x2) of the channels is used as in ResNet-18. The architecture can be seen in Table 4.8.

Layer Name	Block
conv1	7x7, 4, stride 2
pool	3x3 max pool, stride 2
conv2_x	$\begin{bmatrix} \text{3x3, 4} \\ \text{3x3, 4} \end{bmatrix} \times 2$
conv3_x	$\begin{bmatrix} \text{3x3, 8} \\ \text{3x3, 8} \end{bmatrix} \times 2$
conv4_x	$\begin{bmatrix} \text{3x3, 16} \\ \text{3x3, 16} \end{bmatrix} \times 2$
conv5_x	$\begin{bmatrix} \text{3x3, 32} \\ \text{3x3, 32} \end{bmatrix} \times 2$
average pool	
27-d fc	

Table 4.8: ResNet-lite architecture.

4.8.2. Training

The same training strategy implemented in section 4.5 is used. ResNet-lite converges to a lower validation loss using Adam (as seen in Figure 4.32), and as such, the weights of that model are used. This model required 1h:57m:37s and 33 epochs to converge.



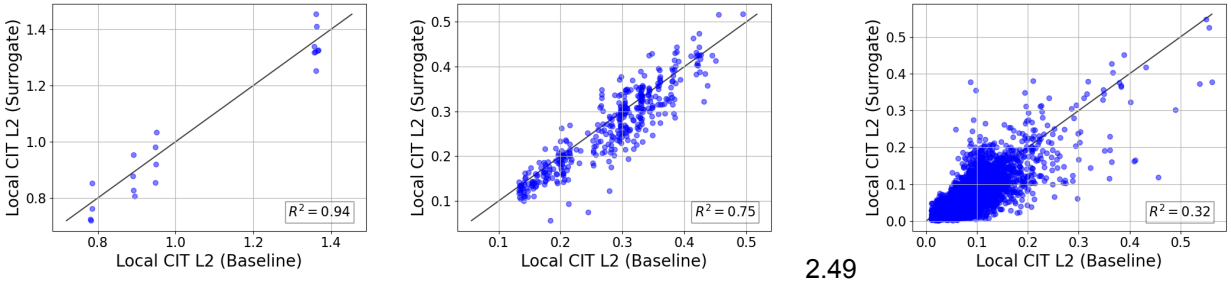
(a) SGD **(b)** Adam

Figure 4.32: ResNet-lite log loss diagrams using different optimizers.

4.8.3. In-distribution prediction results

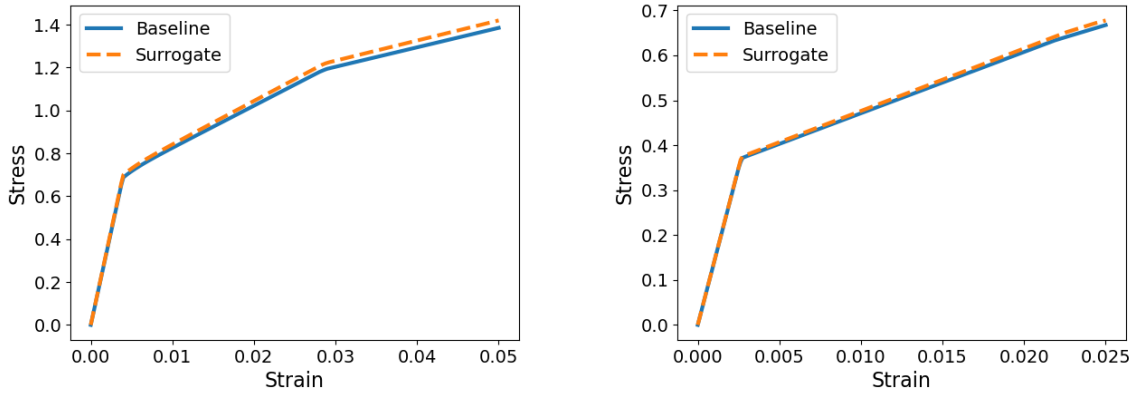
n_c	Physics	CIT MAE	$t_{baseline}$	Speedup	Stress MAPE (uniaxial tension)	Stress MAPE (pure shear)
5	elastic	0.219	2.60E-01s	1.16	0.67%	0.63%
	plastic				1.98%	1.80%
	plastic + SCS				2.56%	2.14%
20	elastic	0.180	1.76E+00s	1.51	0.13%	0.62%
	plastic				1.31%	1.51%
	plastic + SCS				0.61%	1.41%
80	elastic	0.153	1.95E+01s	1.81	0.63%	0.63%
	plastic				2.85%	1.84%
	plastic + SCS				3.07%	1.90%

Table 4.9: Model summary ResNet-lite (in-distribution).



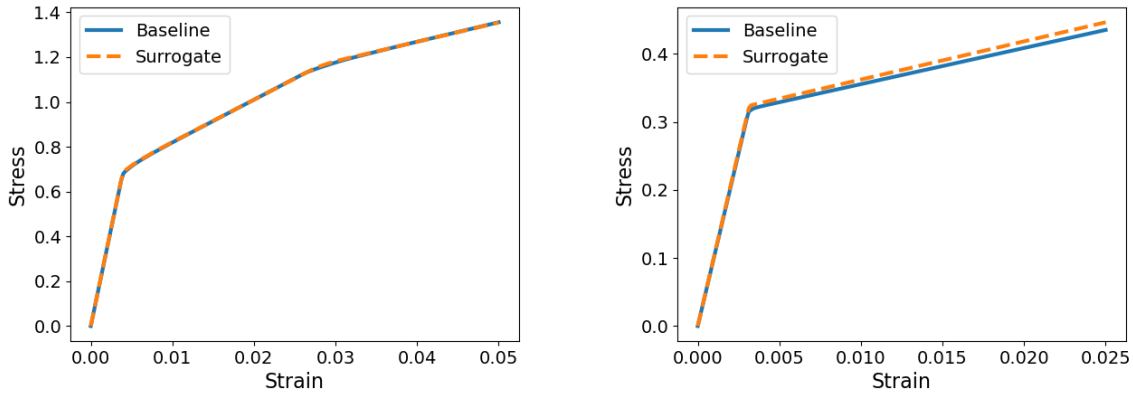
(a) $n_c = 5$ **(b)** $n_c = 20$ **(c)** $n_c = 80$

Figure 4.33: L2 Ground Truth vs Prediction (in-distribution)



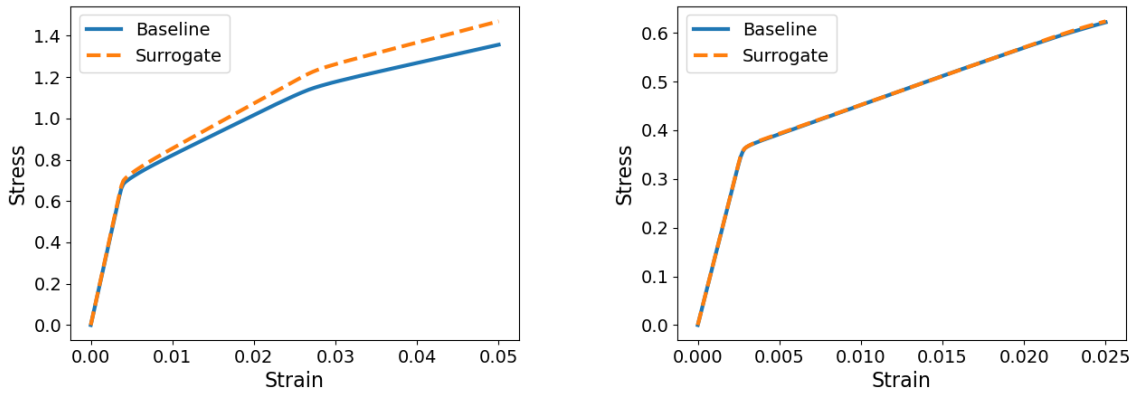
(a) Uniaxial tension **(b)** Pure Shear

Figure 4.34: Stress-Strain response (plastic + SCS, in-distribution, $n_c = 5$).



(a) Uniaxial tension **(b)** Pure Shear

Figure 4.35: Stress-Strain response (plastic + SCS, in-distribution, $n_c = 20$).



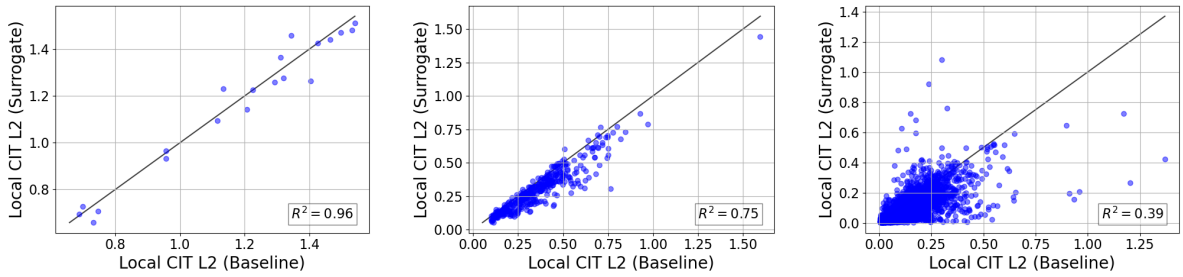
(a) Uniaxial tension **(b)** Pure Shear

Figure 4.36: Stress-Strain response (plastic + SCS, in-distribution, $n_c = 80$).

4.8.4. Out-of-distribution prediction results

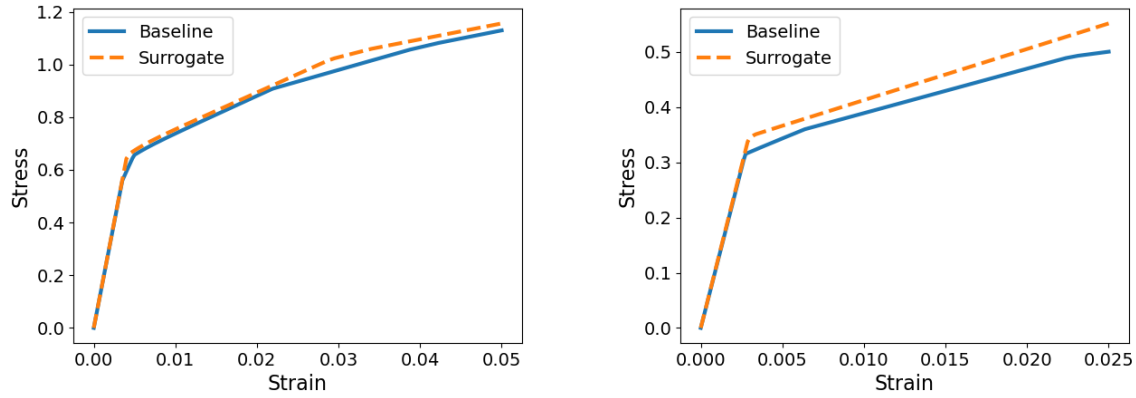
n_c	Physics	CIT MAE	$t_{baseline}$	Speedup	Stress MAPE (uniaxial tension)	Stress MAPE (pure shear)
5	elastic	0.675	2.60E-01s	1.12	0.33%	0.90%
	plastic				8.68%	10.43%
	plastic + SCS				3.15%	5.28%
20	elastic	0.399	1.68E+00s	1.46	0.34%	0.68%
	plastic				21.88%	15.48%
	plastic + SCS				4.84%	5.11%
80	elastic	0.236	1.94E+01s	1.85	0.43%	1.29%
	plastic				34.26%	26.44%
	plastic + SCS				5.83%	7.03%

Table 4.10: Model summary ResNet-lite (out-of-distribution).



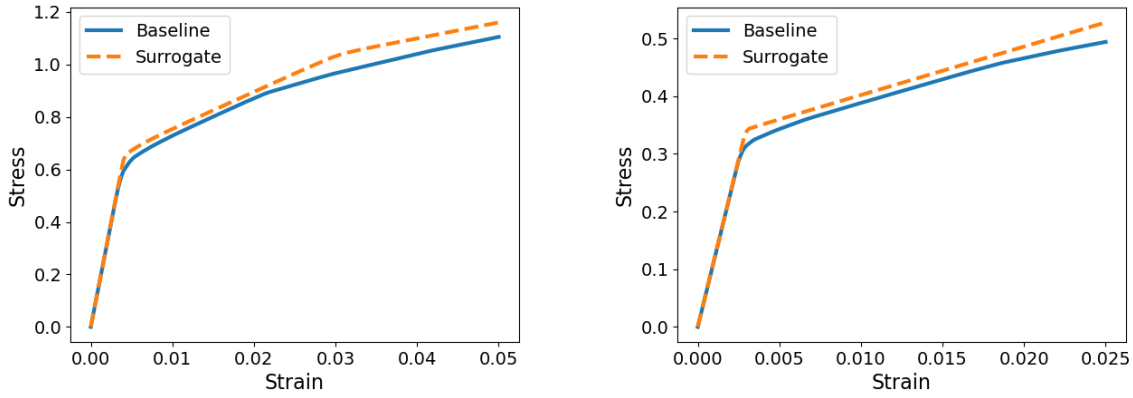
(a) $n_c = 5$ **(b)** $n_c = 20$ **(c)** $n_c = 80$

Figure 4.37: L2 Ground Truth vs Prediction (out-of-distribution).



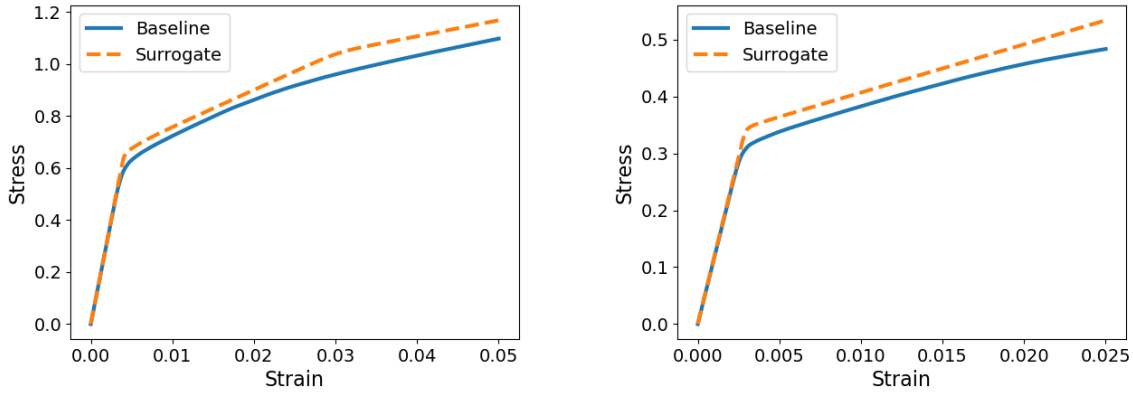
(a) Uniaxial tension **(b)** Pure Shear

Figure 4.38: Stress-Strain response (plastic + SCS, out-of-distribution, $n_c = 5$).



(a) Uniaxial tension **(b)** Pure Shear

Figure 4.39: Stress-Strain for (plastic + SCS, out-of-distribution, $n_c = 20$).



(a) Uniaxial tension **(b)** Pure Shear

Figure 4.40: Stress-Strain response (plastic + SCS, out-of-distribution, $n_c = 80$).

4.8.5. Discussion

With the results of the accelerated surrogate shown for in-distribution and out-of-distribution samples in Table 4.10 and Table 4.9, respectively, a number of observations can be made based on the data as well as the supporting figures

The accelerated surrogate model aims to obtain a faster solution than the baseline. It can be concluded that this requirement is satisfied when considering the speedup factor of both in-distribution and out-of-distribution samples. The speedup again changes as a function of n_c, with higher speedup values observed for higher n_c. This is due to the quadratic scaling in the number of the upper off-diagonal CIT terms as opposed to the linear scaling of the diagonal ones.

As expected, for a model with fewer parameters, the accelerated surrogate's accuracy is degraded compared to the standard surrogate. This is already seen in the log loss plots where it could be observed that the validation loss is almost an order of magnitude lower when comparing Figure 4.13 to Figure 4.32. Interestingly, for in-distribution samples, although the CIT MAE values are lower than out-of-distribution samples, the stress MAPE values are comparable (and even lower in some cases). However, the out-of-distribution results are significantly poorer than the standard surrogate, with stress MAPE values as high as 35% and 27% for the uniaxial tension and pure shear, respectively. This shows that the accelerated model can make accurate in-distribution predictions yet fails to generalize to out-of-distribution samples in most cases.

5 Conclusions

To conclude the exploration of this thesis investigation, the research question posed is again brought up and examined.

> **To what extent can a data-driven surrogate model accurately predict and accelerate the computation of the cluster interaction tensors (CIT) in self-consistent clustering analysis (SCA)?**

A number of sub-questions are also posed to shed light on the answer sufficiently. These sub-questions are discussed one by one, and with them, conclusions regarding the main research question can be made.

SQ1: To what extent can a data-driven surrogate model accurately predict the CIT?

This sub-question is investigated in chapter 4 (specifically section 4.5). Using a ResNet-18 architecture as a surrogate to predict upper-off diagonal terms in the CIT, it is shown that the stress-strain responses could be accurately predicted with the surrogate. The surrogate is able to make accurate predictions with both in-distribution microstructures (similar to the ones it was trained on) and generalizes to out-of-distribution microstructures (that represent realistic specimens the model may encounter when applied in an engineering context).

Additionally, several conclusions can be derived from the sensitivity study in section 4.3. For example, the three cases considered (elastic, plastic and plastic+SCS) exhibit different behaviors. Noise in the CIT only affects the stress-strain solution of the elastic case and does not affect convergence. In the plastic case, it affects its stress-strain solution and convergence behavior. Surprisingly, in the plastic+SCS case, the model appears to self-correct for a noisy CIT (in terms of its stress-strain response) but also struggles more with convergence than in the other two cases. Finally, it is shown that the solution accuracy and convergence behavior is degraded with increasing numbers of clusters. These results must be considered when applying a data-driven surrogate model in SCA. Additionally, a dilemma presents itself whereby the surrogate model is ideally applied to cases containing many clusters to get the fastest speedup yet has the adverse effect of degrading solution accuracy. This requires a more accurate model for simulations with more clusters.

SQ2: To what extent can a data-driven surrogate model accelerate predictions of the CIT?

This sub-question is investigated in chapter 4 (specifically section 4.8). A modified version of ResNet-18 called ResNet-lite is created that has fewer parameters and subsequently requires fewer FLOPs to compute the CIT components compared to the baseline model, increasing computational efficiency. This, however, comes at the cost to the model's accuracy. In particular, the model's generalizability is severely hindered as the out-of-distribution predictions are not accurate.

Although ResNet-lite demonstrates a more efficient implementation of the surrogate, it should not be treated as the end-all, be-all architecture. The appropriate model could lie on a spectrum in terms

of parameters between ResNet-lite and ResNet-18 or even as an entirely different architecture. The choice of the model architecture largely depends on the application in which the surrogate is to be implemented and how vital accuracy and speed are to the application.

SQ3: How can data for a surrogate model's training be generated?

This sub-question is primarily investigated in chapter 3. A novel method for generating clustered microstructures using gradient noise is proposed. It can create a clustered microstructure within milliseconds, and as such, it is particularly well suited for generating large datasets for data-driven surrogate model training. Having large datasets is shown to be important in section 4.7 as the CIT prediction and stress-strain response's accuracy suffers from smaller datasets. Additionally, it is shown in section 4.3 that surrogates trained on these microstructures could make accurate predictions on out-of-distribution microstructures, demonstrating the transferability of knowledge from these datasets.

6

Recommendations

Although the thesis investigation is able to sufficiently resolve the research question, certain areas of interest were uncovered in the process and some questions remain open. A number of potential research directions are therefore highlighted that can help steer future investigations.

Firstly, more consideration can be given to the architecture being used. In this study only ResNet-18 is considered and it proved to be sufficient for demonstrating the proof-of-concept. However ResNet-18 is not considered state of the art anymore when it comes to the ImageNet dataset prediction, with transformer architectures such as ViT having better accuracy's [11]. The downside of using transformer architectures is that they require more data to learn invariances when compared to CNNs which have many of them built in by design (such as translation invariance). On this end, the clustered microstructure generator can be used to generate the larger datasets efficiently.

Tangent to the previous point, a more robust hyperparameter optimization can be performed on the architecture to achieve predetermined objectives regarding accuracy or computational efficiency. This study demonstrated that a ResNet-18 architecture with a reduced number of channels (called ResNet-lite) yields a speedup when compared to the baseline. However other parameters in the model such as stride can be modified. Additionally, methods introduced in chapter 2 for compressing architectures such as CP decomposition or tensor contraction layers can be implemented to gain in efficiency. Alternatively, reinforcement learning approaches that aim to optimize an architecture for computational efficiency such as the work done by Howard et al. [24] when creating MobileNet-V3 can be tried.

Next, full tensor predictions can be attempted. This study focused on pairwise predictions based on the cluster support functions. Therefore the surrogate only had knowledge of the problem at a local tensor level. Using a global approach (for example using graph neural networks) the model may learn dependencies between local components that can potentially improve accuracy. Additionally, pruning could be used in such an approach to remove cluster dependencies that do not have a significant contribution to the final result. If effective pruning could be achieved, the quadratic computational complexity of the algorithm (which is currently considered one of the biggest limitations) may be reduced.

Additionally, a major extension of this research would be expanding the method to accommodate three-dimensional problems. Most of the steps in the process already allow for this. The clustered microstructure generator can easily be expanded to generate datasets with 3D clustered microstructures and SCA has previously been shown to be capable of solving 3D problems [37, 15]. The biggest obstacle in expanding the method to the third dimension would likely be the choice of a suitable architecture for the surrogate model. Although 3D neural networks exist in literature [54, 28], their use has not become as widespread as their 2D counterparts. As such, significant consideration would need to be given to obtain an efficient and accurate model for such an application.

Finally, the dataset generators can also be refined. Firstly, techniques utilizing k-means can be implemented in the binarization step of the cluster generator in order to match SCA's clustering behavior more closely and mitigate biting bias. Furthermore, investigations into tuning the roughness and volume fraction parameters can be conducted to obtain more general datasets or datasets suited for particular applications. This could be done by generating representative out-of-distribution datasets and analyzing the characteristics of their clusters. With a more rigorous investigation into optimal dataset generation processes, better accuracy can be achieved in out-of-distribution predictions.

References

[1] M. Z. Alom et al. “A State-of-the-Art Survey on Deep Learning Theory and Architectures”. In: *Electronics* 8.3 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8030292. URL: https://www.mdpi.com/2079-9292/8/3/292.

[2] L. Alzubaidi et al. “Review of Deep Learning: Concepts, CNN Architectures, challenges, applications, Future Directions”. In: *Journal of Big Data* 8.1 (Mar. 2021). DOI: 10.1186/s40537-021-00444-8.

[3] H. Bolandi et al. “Bridging Finite Element and deep learning: High-resolution stress distribution prediction in structural components”. In: *Frontiers of Structural and Civil Engineering* 16.11 (Nov. 2022), pp. 1365–1377. DOI: 10.1007/s11709-022-0882-5.

[4] E. Brynjolfsson and A. McAfee. *What’s driving the Machine Learning Explosion?* Nov. 2020. URL: https://hbr.org/2017/07/whats-driving-the-machine-learning-explosion.

[5] J. Bullinaria. *Lecture Notes for Data Structures and Algorithms*. https://www.cs.bham.ac.uk/~jxb/DSA/dsa.pdf. School of Computer Science, University of Birmingham. 2019.

[6] J. W. Cahn and J. E. Hilliard. “Free energy of a nonuniform system. I. Interfacial Free Energy”. In: *The Journal of Chemical Physics* 28.2 (Feb. 1958), pp. 258–267. DOI: 10.1063/1.1744102.

[7] T. Chen and H. Chen. “Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems”. In: *IEEE Transactions on Neural Networks* 6.4 (July 1995), pp. 911–917. DOI: 10.1109/72.392253.

[8] R. Christensen and K. Lo. “Solutions for effective shear properties in three phase sphere and cylinder models”. In: *Journal of the Mechanics and Physics of Solids* 27.4 (1979), pp. 315–330. DOI: 10.1016/0022-5096(79)90032-2.

[9] W. Dean. “Computational Complexity Theory”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021.

[10] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo. “Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning”. In: *Sustainable Computing: Informatics and Systems* 38 (Apr. 2023), p. 100857. ISSN: 2210-5379. DOI: 10.1016/j.suscom.2023.100857. URL: http://dx.doi.org/10.1016/j.suscom.2023.100857.

[11] A. Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[12] B. P. Ferreira, F. Andrade Pires, and M. Bessa. “Adaptivity for clustering-based reduced-order modeling of localized history-dependent phenomena”. In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022), p. 114726. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.114726. URL: https://www.sciencedirect.com/science/article/pii/S0045782522000895.

[13] B. P. Ferreira, F. M. A. Pires, and M. A. Bessa. “CRATE: A Python package to perform fast material simulations”. In: *Journal of Open Source Software* 8.87 (2023), p. 5594. DOI: 10.21105/joss.05594. URL: https://doi.org/10.21105/joss.05594.

[14] B. P. Ferreira, J. L. Vila-Chã, and F. A. Pires. “An adaptive multi-temperature isokinetic method for the RVE generation of particle reinforced heterogeneous materials, part II: Numerical assessment and statistical analysis”. In: *Mechanics of Materials* 165 (Feb. 2022), p. 104068. DOI: 10.1016/j.mechmat.2021.104068.

[15] B. P. Ferreira. “Towards Data-driven Multi-scale Optimization of Thermoplastic Blends: Microstructural Generation, Constitutive Development and Clustering-Based Reduced-Order Modeling”. PhD thesis. Repositório Aberto, 2022. URL: https://hdl.handle.net/10216/146900.

[16] P. J. Freire et al. *Computational Complexity Evaluation of Neural Network Applications in Signal Processing*. 2022. arXiv: 2206.12191 [eess.SP].

[17] K. Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. DOI: 10.1007/bf00344251.

[18] K. Gholami, F. Ege, and R. Barzegar. “Prediction of composite mechanical properties: Integration of deep neural network methods and finite element analysis”. In: *Journal of Composites Science* 7.2 (Feb. 2023), p. 54. DOI: 10.3390/jcs7020054.

[19] C. P. Grant. “Spinodal decomposition for the cahn-hilliard equation”. In: *Communications in Partial Differential Equations* 18.3–4 (Jan. 1993), pp. 453–490. DOI: 10.1080/03605309308820937.

[20] Z. Hashin and S. Shtrikman. “A variational approach to the theory of the elastic behaviour of Multiphase Materials”. In: *Journal of the Mechanics and Physics of Solids* 11.2 (1963), pp. 127–140. DOI: 10.1016/0022-5096(63)90060-7.

[21] K. He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[22] R. Hill. “A self-consistent mechanics of Composite Materials”. In: *Journal of the Mechanics and Physics of Solids* 13.4 (1965), pp. 213–222. DOI: 10.1016/0022-5096(65)90010-4.

[23] G. Hinton, O. Vinyals, and J. Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[24] A. Howard et al. *Searching for MobileNetV3*. 2019. arXiv: 1905.02244 [cs.CV].

[25] A. G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[26] B. Jacob et al. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. 2017. arXiv: 1712.05877 [cs.LG].

[27] D. Jakes et al. “Perlin Noise Generation of Physiologically Realistic Patterns of Fibrosis”. In: *bioRxiv* (2019). DOI: 10.1101/668848.

[28] S. Korolev et al. *Residual and Plain Convolutional Neural Networks for 3D Brain MRI Classification*. 2017. arXiv: 1701.06643 [cs.CV].

[29] J. Kossaifi et al. “Tensor Regression Networks”. In: *Journal of Machine Learning Research* 21.123 (2020), pp. 1–21. URL: http://jmlr.org/papers/v21/18-503.html.

[30] N. B. Kovachki et al. “Neural Operator: Learning Maps Between Function Spaces”. In: *ArXiv* abs/2108.08481 (2021). URL: https://api.semanticscholar.org/CorpusID:237213378.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[32] A. Lagae et al. “A Survey of Procedural Noise Functions”. In: *Computer Graphics Forum* 29.8 (2010), pp. 2579–2600. DOI: https://doi.org/10.1111/j.1467-8659.2010.01827.x.

[33] V. Lebedev et al. *Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition*. 2015. arXiv: 1412.6553 [cs.CV].

[34] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[35] E. Lejeune. *Mechanical MNIST: A benchmark dataset for mechanical metamodels*. 2020. URL: https://open.bu.edu/handle/2144/39813.

[36] R. J. Lipton and K. W. Regan. *People, problems, and proofs essays from Gödel’s Lost Letter: 2010*. Springer Berlin, 2016.

[37] Z. Liu, M. Bessa, and W. K. Liu. “Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials”. In: *Computer Methods in Applied Mechanics and Engineering* 306 (2016), pp. 319–341. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2016.04.004.

[38] A. Melro, P. Camanho, and S. Pinho. “Generation of random distribution of fibres in long-fibre reinforced composites”. In: *Composites Science and Technology* 68.9 (July 2008), pp. 2092–2102. DOI: 10.1016/j.compscitech.2008.03.013.

[39] T. Mori and K. Tanaka. “Average stress in matrix and average elastic energy of materials with misfitting inclusions”. In: *Acta Metallurgica* 21.5 (1973), pp. 571–574. DOI: 10.1016/0001-6160(73)90064-3.

[40] B. Moseley. “Physics-informed Machine Learning: From concepts to real-world applications”. PhD thesis. 2022.

[41] M. Mozaffar et al. “Deep learning predicts path-dependent plasticity”. In: *Proceedings of the National Academy of Sciences* 116.52 (Dec. 2019), pp. 26414–26420. DOI: 10.1073/pnas.1911815116.

[42] T. Mura. *Micromechanics of defects in solids*. M. Nijhoff, 1982.

[43] M. V. Pathan et al. “Predictions of the mechanical properties of unidirectional fibre composites by supervised machine learning”. In: *Scientific Reports* 9.1 (Sept. 2019). DOI: 10.1038/s41598-019-50144-w.

[44] Z. Qi et al. “Prediction of mechanical properties of carbon fiber based on cross-scale FEM and machine learning”. In: *Composite Structures* 212 (2019), pp. 199–206. ISSN: 0263-8223. DOI: https://doi.org/10.1016/j.compstruct.2019.01.042. URL: https://www.sciencedirect.com/science/article/pii/S0263822318335529.

[45] A. Radermacher and S. Reese. “POD-based model reduction with empirical interpolation applied to nonlinear elasticity”. In: *International Journal for Numerical Methods in Engineering* 107.6 (2016), pp. 477–495. DOI: https://doi.org/10.1002/nme.5177.

[46] D. Raj. *Why should you care about deep learning?* June 2021. URL: https://www.crowdai.com/blog/why-should-you-care-about-deep-learning.

[47] A. Reuss. “Berechnung der Fließgrenze von Mischkristallen auf Grund der Plastizitätsbedingung für Einkristalle .” In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 9.1 (1929), pp. 49–58. DOI: 10.1002/zamm.19290090104.

[48] E. Rowell et al. *Big-O Complexity Chart*. URL: https://www.bigocheatsheet.com/.

[49] O. Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[50] M. Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

[51] J. Seidman et al. “NOMAD: Nonlinear Manifold Decoders for Operator Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 5601–5613. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/24f49b2ad9fbe65eefbfd99d6f6c3fd2-Paper-Conference.pdf.

[52] R. Sepasdar, A. Karpatne, and M. Shakiba. “A data-driven approach to full-field nonlinear stress distribution and failure pattern prediction in composites using Deep Learning”. In: *Computer Methods in Applied Mechanics and Engineering* 397 (July 2022), p. 115126. DOI: 10.1016/j.cma.2022.115126.

[53] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[54] D. Tran et al. *A Closer Look at Spatiotemporal Convolutions for Action Recognition*. 2018. arXiv: 1711.11248 [cs.CV].

[55] A. Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[56] S. S. Vel and A. J. Goupee. “Multiscale thermoelastic analysis of random heterogeneous materials: Part I: Microstructure characterization and homogenization of material properties”. In: *Computational Materials Science* 48.1 (2010), pp. 22–38. ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2009.11.015.

[57] W. Voigt. “Ueber die beziehung zwischen den beiden elasticitätsconstanten isotroper Körper”. In: *Annalen der Physik* 274.12 (1889), pp. 573–587. DOI: 10.1002/andp.18892741206.

[58] Y. D. Wang et al. *ML-LBM: Machine Learning Aided Flow Simulation in Porous Media*. 2020. arXiv: 2004.11675 [physics.flu-dyn].

[59] L. Zhang et al. “Fast calculation of interaction tensors in clustering-based homogenization”. In: *Computational Mechanics* 64.2 (2019), pp. 351–364. DOI: 10.1007/s00466-019-01719-x.

[60] X. Zhang et al. *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*. 2017. arXiv: 1707.01083 [cs.CV].