MACHINE LEARNING FOR MATERIAL DESIGN OF CONCRETE

PROJECT

by

Krishna Mundra

School of Infrastructure
Indian Institute of Technology Bhubaneswar
Bhubaneswar-752050, Odisha
May, 2023

MACHINE LEARNING FOR MATERIAL DESIGN OF CONCRETE

A PROJECT

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
in
CIVIL ENGINEERING

By

Krishna Mundra

19CE02004

Under the supervision of
Dr. Mohammad Masiur Rahaman

School of Infrastructure
Indian Institute of Technology Bhubaneswar
Bhubaneswar-752050, Odisha
May, 2023

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"MACHINE LEARNING FOR MATERIAL DESIGN OF CONCRETE"**, in the partial fulfillment of the requirement for the award of the Degree of Bachelor of Technology in Civil Engineering and submitted in the School of Infrastructure, Indian Institute of Technology Bhubaneswar, is an authentic record of my own work carried out during a period from July 2022 to February 2023 under the supervision of **Dr. Mohammad Masiur Rahaman**, IIT Bhubaneswar.

The matter presented in the project has not been submitted by me for the award of any degree of this or any other Institute/ University

KRISHNA MUNDRA (19CE02004)

This is to certify that the above statement made by the candidates is correct to the best of our knowledge.

Signature of Supervisor (S) **Signature of H.O.S.**

ABSTRACT

In the fields of multi-scale computational science and engineering, homogenization is a method that is often employed for the purpose of forecasting the collective response of heterogeneous materials and extracting useful mechanical characteristics. In this study, a three-dimensional deep convolutional neural network, also known as a 3D-CNN, is used with the goal of predicting the effective material characteristics of representative volume elements (RVEs) for Polymer Impregnated Concrete (PIC). Training the 3D-CNN models requires the use of a high-fidelity data-set that was produced via a technique that utilized computational homogenization. The inference results of the trained networks on data that has not been seen indicate that the network is capable of capturing the micro-structural features of RVEs and that it generates an accurate prediction of effective stiffness and Poisson's ratio. These findings are based on the fact that the trained networks were given data that they had never seen before. In the following, we will examine, one after the other, the advantages that the 3D-CNN has over the more traditional finite-element-based method of homogenization. These advantages pertain to the computational efficiency, uncertainty quantification, and transferability of the model. The prominent characteristics of the 3D-CNN technique make it a potentially appropriate option for enabling material design with the rapid iteration of product design and effective quantification of uncertainty.

ACKNOWLEDGEMENT

The opportunity I had with Dr. Mohammad Masiur Rahaman as my supervisor conferred me with an invaluable learning and development prospect. I am grateful to him for providing me with this chance and for his constant encouragement and motivation. His continual guidance throughout the project and his immense knowledge and experience motivated me during the course of this project.

I extend my gratitude to the Indian Institute of Technology, Bhubaneswar for giving me this wonderful opportunity. My gratitude also goes to all of my friends who directly or indirectly helped me to complete this project report.

At last but not least I acknowledge with deep sense of reverence my gratitude towards my parents and members of my family, who has always supported me morally.

CONTENTS

Candidate's Declaration i

Abstract iii

Acknowledgement v

Table of Contents vii

List of Figures viii

List of Acronyms xi

1 INTRODUCTION **1**

1.1 General 1

1.2 Motivation 5

1.3 Objective and Scope 6

2 LITERATURE REVIEW **7**

3 METHODOLOGY **9**

3.1 Generation of Dataset and Preprocessing 9

3.2 MATLAB implementation 10

3.2.1 The element stiffness matrix and load vectors 11

3.2.2 Degrees of freedom and periodic boundary conditions 13

3.2.3 Assembly of the stiffness matrix 14

3.2.4 Load vectors and solution 14

3.2.5 Homogenization 14

3.3 Convolutional Neural Network 15

3.3.1 Convolution 16

3.3.2 Data Transformations 17

3.3.3 Input Processing 18

3.3.4 Data Augmentation 18
3.3.5 Network Layers 19
3.3.6 Convolution Layer 19
3.3.7 Pooling or Subsampling Layer 20
3.3.8 Fully connected or dense layer 20
3.3.9 Network Initialization 21
3.3.10 Reguarization 22
3.3.11 Loss Functions 22
3.3.12 Model Visualization 23
3.4 Implementation of 3D Convolution Neural Network 24
3.4.1 3D Convolutional Neural Network 24

4 RESULTS AND DISCUSSION 29
4.1 Design of 3-D CNN Architecture 29
4.2 Prediction of effective properties 30

5 CONCLUSION AND FUTURE SCOPE 35

LIST OF FIGURES

1.1.1 Presence of voids in Pre-cast Concrete 2

1.1.2 Polymer Impregnated Concrete 3

1.1.3 Framework of material structure optimization 5

3.1.1 A section of a microstructure consisting of two materials(white and black)[2] 10

3.2.1 Figure (a) shows the finite element mesh used to discretize the unit cell, while Figure (b) shows the corresponding structure of the indicator matrix x.[3] 11

3.2.2 Structure of the edofMat-matrix for 12 element mesh in (a) a non-periodic version, and (b) periodic version. The bold numbers denote the row numbers and indicate which element the degrees of freedom in the row belong to[3] 13

3.3.1 A typical convolutional neural network with CONV-ReLU, POOL layers repeated and followed by the fully-connected layers at the end.[3] 16

3.3.2 Architecture of a multilayer neural network.[3] 17

3.3.3 Example of convolution Operation[3] 17

3.3.4 Sample CovNet architecture[3] 19

3.3.5 Example of Convolutional Layer[3] 19

3.3.6 Example of Dropout based regularization[3] 22

3.3.7 Visualizing activation of layers[19] 23

3.3.8 Visualizing trained filters[19] 24

3.4.1 Proposed 3D-CNN architecture for effective properties prediction of heterogeneous materials.[19] 25

3.4.2 Convolution operation in the 3D-CNN model.[19] 25

3.4.3 Max pooling operation in the 3D-CNN model. 26

3.4.4 Rectified linear unit (ReLU) function as the activation function[19] 26

4.2.1 Comparison between the 3D-CNN prediction and ground truth (FEA) . . 32

4.2.2 Histogram distribution of predicted effective properties for RVEs with mean VF of 0.07: (a) E_{11} (b) E_{22} (c) E_{33} (d) G_{23} (e) G_{13} (f) G_{12} 33

4.2.3 Histogram distribution of predicted effective properties for RVEs with mean VF of 0.07:(a) ν_{21} (b) ν_{31} (c) ν_{12} (d) ν_{32} (e) ν_{13} (f) ν_{23}. 34

LIST OF ACRONYMS

Fullform of acronyms have been mentioned on it's first instance of appearance. For instance polymer impregnated concrete has been mentioned as PIC on page number 1.

Chapter 1

INTRODUCTION

1.1 General

The significance of heterogeneous materials in civil engineering cannot be overstated. The use of different materials with varying properties throughout the structure of a building or infrastructure project can lead to improved strength, durability, and overall performance. Heterogeneous materials can also offer greater design flexibility and aesthetic appeal, as well as contribute to more sustainable construction practices. From reinforced concrete to composite materials, the diverse range of heterogeneous materials available to civil engineers provides a powerful toolset for building structures that are safe, efficient, and visually appealing. As such, the study and understanding of heterogeneous materials is of great importance to the field of civil engineering, enabling engineers to push the boundaries of what is possible in construction and design.

The most often utilised building material worldwide is concrete. Sand, natural or crushed aggregates, cement, and water for mixing are the conventional ingredients. Significant environmental harm is caused by the extensive exploitation of sand and aggregates from quarries for the manufacture of concrete and masonry bricks. Many efforts have been undertaken throughout the years to add or substitute different kinds of inclusion elements into concrete mixtures. Such activities are often driven by the desire to: (i) reuse or recycle the inclusion materials; (ii) limit the consumption of natural aggregates; and (iii) adjust the fresh and hardened characteristics of the resultant mixes in ways that are appropriate.

Polymer impregnated concrete (PIC) is a type of concrete that has gained popularity in recent years due to its exceptional durability, strength, and resistance to harsh environ-

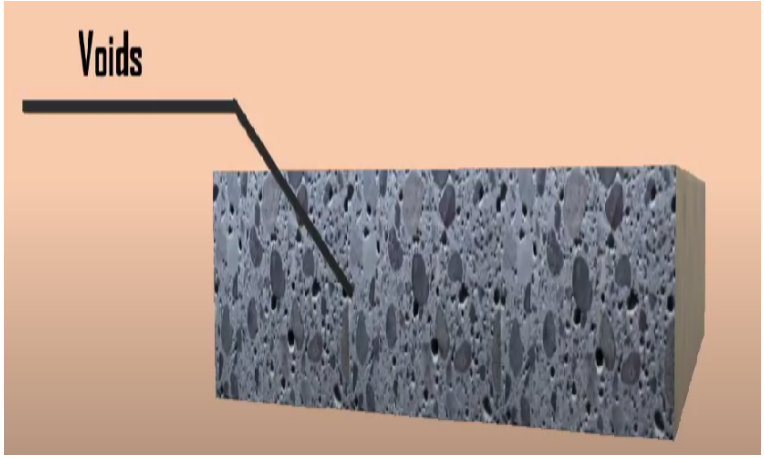


Figure 1.1.1: Presence of voids in Pre-cast Concrete

mental conditions. The manufacturing process of PIC involves injecting a polymer resin into the concrete under high pressure, filling the pores and capillaries of the concrete and making it more durable and resistant to cracking. PIC can also be made by coating the surface of the concrete with a polymer, providing additional protection against wear and tear. PIC is made using polymers such as epoxy, polyester, and acrylics, mixed with a hardener to form a resin. The resin is then infused into the concrete under high pressure. The use of PIC offers numerous advantages over traditional concrete, including resistance to weathering, chemical attacks, and abrasion. Additionally, PIC has a longer lifespan and requires less maintenance, making it a cost-effective solution for infrastructure projects. PIC is frequently used in infrastructure projects such as highways, bridges, and parking garages, where durability and longevity are essential. It is also used in the construction of floors, walls, and roofs in buildings where high strength and resistance to wear and tear are required. Furthermore, PIC can be used to rehabilitate aging concrete structures, providing an economical solution for extending their lifespan.

Homogenization of material properties in polymer impregnated concrete (PIC) is of significant importance in the field of civil engineering. This is because polymer reinforced concrete (PRC) is a composite material that is made up of multiple components with varying mechanical properties at the microscopic level. Homogenization allows for the calculation of effective properties of the composite material at a macroscopic level, which

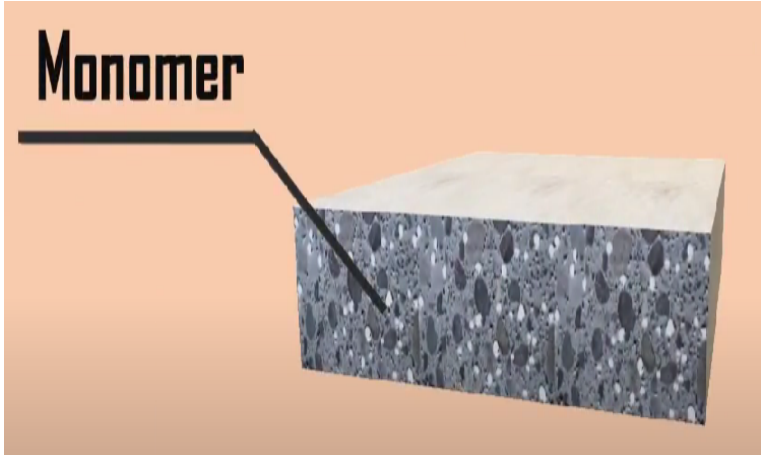


Figure 1.1.2: Polymer Impregnated Concrete

can be used to design and analyze PIC structures. This process helps to simplify the analysis of PIC, which can be challenging due to the heterogeneity of the material and allows engineers to accurately predict its mechanical behavior. This in turn leads to more efficient and accurate design of PRC structures, resulting in safer, more durable, and more sustainable infrastructure. Homogenization also enables the optimization of the properties of PRC, leading to the development of more advanced and effective materials. As such, the study and implementation of homogenization in PIC are of great significance to the field of civil engineering, offering engineers a powerful tool to design and construct safer, more efficient, and more sustainable infrastructure.

One of the most important presumptions in the area of computer modelling of composite materials is that these materials contain periodic microstructures that may be modelled using representative components (RVEs). To create composite materials with remarkable combinations of attributes, it is crucial to have a thorough grasp of how different RVE parameters, such as microstructure, component phase, volume fraction, etc., affect the macroscopic material properties. In the majority of composite design difficulties, effective material properties are used rather than taking into account all the components and microstructure. The development of mathematical and computational approaches for determining efficient and homogenised material qualities has received a lot of attention. These methods are used to predict how composite materials would behave under various

loading scenarios. Therefore, in order to design and produce composite materials with the appropriate features, a thorough knowledge of the RVE and effective material characteristics is essential.

Since the homogenization theory was first created to analyse partial differential equations (PDEs) with fast fluctuating coefficients, it has increasingly been employed to describe the mechanics of periodic microstructures in composite materials. Different homogenization techniques have been developed to provide useful properties that may be used to macroscopic structural analysis. Analytical techniques, semi-analytical methods, and numerical methods may be used to classify these approaches. The Voigt and Reuss models, for example, provide a quick but approximate upper and lower limit for a variety of heterogeneous material properties. The volume proportion of inclusions and the level of phase contrast both influence the distance between the boundaries. While semi-analytical approaches like the generalised method of cells, self-consistent scheme, and Mori-Tanaka method are more sophisticated and computationally intensive than analytical methods, they provide more accurate predictions than the latter. Intricate discretizations and expensive computations are necessary for numerical techniques like finite element, boundary element, and rapid Fourier transforms to be able to manage the homogenization of materials with arbitrary microstructures and constitutive models. In spite of these difficulties, numerical methods have been shown to be successful in simulating multi-scale material behaviour for both linear and nonlinear issues, providing the material components and microstructure are well specified.

The popularity of data science has increased recently, which has led to the application of a number of machine learning (ML) approaches in the planning, evaluation, and modelling of materials. A special framework known as materials knowledge systems (MKS) has been created to capitalise on the advantages of both analytical and numerical techniques. The statistical continuum mechanics theory, which defines the structure-property

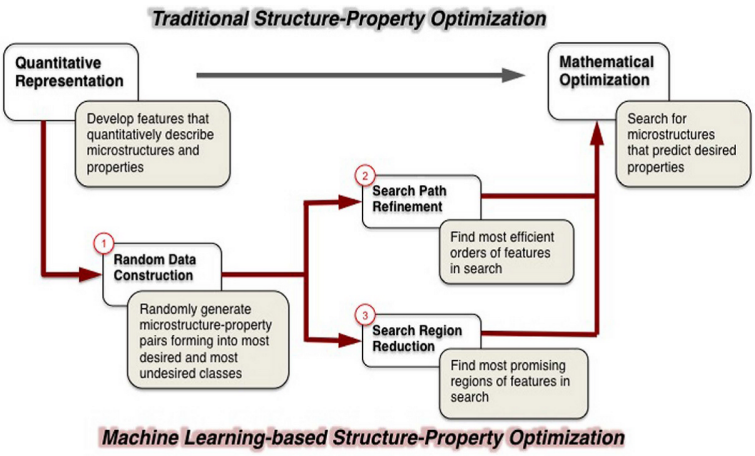


Figure 1.1.3: Framework of material structure optimization

connection of the material as the sum of a polynomial series, serves as the mathematical foundation for MKS. Each term in the series represents the fundamental understanding of the localization relationship, which is a result of statistics relating to local microstructure and the physics coefficients that go along with it. The core of MKS, which uses the outcomes of finite element analysis (FEA) to calibrate these coefficients, is the discrete Fourier transform (DFT). Numerous articles have shown the excellent accuracy, computational effectiveness, and data-driven features of this system. Other ML approaches may be used in computational mechanics and materials science in addition to these applications. For nonlinear solid homogenization, Fritzen and Kunc offered a two-stage data-driven approach, while Lookman and colleagues employed an active learning technique to explore the search space and find potential solutions that could be used to inform calculations or experiments. When choosing from unexplored data, both the utility function and the surrogate model are used.

1.2 Motivation

In order to address the difficulty of feature engineering, which may be time-consuming and needs specialised knowledge, deep learning (DL) algorithms have been created. Long short-term memory (LSTM), convolutional neural networks (CNN), and fully connected neural networks (FCNN) are DL techniques that can automatically identify the most crucial properties to be learned, making them extremely promising in a variety of applications

including voice recognition, computer vision (CV), natural language processing (NLP), and many more. The ability of DL approaches to locate complicated structures buried inside high-dimensional data has been shown. These methods have been used in material science for a variety of tasks, including the categorization of materials, the classification of defects, the identification and reconstruction of microstructures, the prediction of composite strength, and others. In this study, DL will be used to address multiscale composite concerns, particularly in relation to homogeneity.

1.3 Objective and Scope

The goal of this research is to use Deep Learning to address multiscale issues in Polymer Impregnated Concrete (PIC), specifically in the context of homogenization. The methodology to be employed will be discussed, which involves generating training datasets using 200 Representative Volume Elements (RVEs). During pre-processing, the raw data will be transformed into the input format of the 3D-CNN model, a computational homogenization approach will be used to obtain the labels, and the labels will be rescaled. To determine the most effective network configuration, a set of parametric experiments will be conducted on the hyperparameters of the 3D-CNN. Subsequently, a comparison will be drawn between the results of the Finite Element Analysis (FEA) and the predictions made by the 3D-CNN, with a focus on accuracy and efficiency.

Chapter 2

LITERATURE REVIEW

This study [25] was focused on using soft computing techniques to determine the effective elastic properties of concrete mixtures containing randomly oriented chopped inclusions and to identify optimal inclusion ratios. The study [25] also addressed the efficient resolution of the inverse problem, which involves identifying the elastic properties of the constituents from those of the mixture.

[2] stated that the method of numerical homogenization provides an efficient way to determine the effective macroscopic properties of periodic composite materials, such as the elasticity tensor. [2], provided with an explanation of the method with the help of a short, self-contained implementation in Matlab. The code computed the effective elasticity tensor of a two-material composite, where one material can be void, and can be easily extended to include more materials. The paper also described in detail how the homogenization technique can be extended to study conductivity, thermal expansion, and fluid permeability. The unit cell of the periodic material can take different shapes, such as square, rectangle, or parallelogram, thus allowing the study of various 2D periodicities.

In their study,[24] demonstrated the potential of 3D-CNN for determining the effective elastic modulus homogenization of composites with a single component. They found that the method has advantages over traditional approaches that use isotropic assumptions for homogenization. In this work, we utilized the 3D-CNN and created a network architecture to predict the anisotropic effective material properties of composites with complex heterogeneous microstructures. We modeled the microstructure of the composite material as a two-phase representative volume element (RVE) with randomly distributed inclusions. We generated a diverse set of RVEs with different inclusion volume fractions and spatial distributions to cover the intrinsic features of the material. We then used finite element analysis to obtain the effective moduli through linear homogenization for each of the sam-

ples. The geometric information of the RVEs was preprocessed to a structured grid that the 3D-CNN can accept. The networks were trained, verified, and tested on synthetic data. The proposed 3D-CNN approach offers several advantages, such as providing an end-to-end solution for predicting the effective material properties of the composites with high efficiency and accuracy, reproducing the probability distribution of the material properties for input characterized with uncertainty, and offering transferability for training a model with new datasets generated from different microstructure configurations. This approach is particularly advantageous for heterogeneous materials with multiple constituents and extremely complex microstructures, as it has demonstrated extraordinary ability in handling high-dimensional inputs[19][10].

Chapter 3

METHODOLOGY

First section explains the theory of homogenization which has been use for data generation in this research. Second section includes the data implementation of the theory mentioned in the first section. Third section explain Convolutional Neural Network (CNN), the DL technique used to generate the model. Fourth section mentions the implementation of CNN technique.

3.1 Generation of Dataset and Preprocessing

The theory of homogenization suggests that one can calculate the macroscopic elasticity tensor E^H_{ijkl} of a periodic composite material.

$$E^H_{ijkl} = \frac{1}{|V|} \int_V E_{pqrs} \left(\varepsilon^{0(ij)}_{pq} - \varepsilon^{(ij)}_{pq} \right) \left(\varepsilon^{0(kl)}_{rs} - \varepsilon^{(kl)}_{rs} \right) dV \tag{3.1}$$

The calculation of the macroscopic elasticity tensor E^H_{ijkl} for a periodic composite material is based on the homogenization theory. The volume of the unit cell is denoted by $|V|$, while the locally varying stiffness tensor is represented by E_{pqrs}. The prescribed macroscopic strain fields are denoted by $\varepsilon^{0(ij)}_{pq}$, and the locally varying strain fields $\varepsilon^{(ij)}_{pq}$ are defined accordingly.

$$\varepsilon^{(ij)}_{pq} = \varepsilon_{pq}\left(\chi^{ij}\right) = \frac{1}{2}\left(\chi^{ij}_{p,q} + \chi^{ij}_{q,p}\right) \tag{3.2}$$

To determine the locally varying strain fields $\varepsilon^{(ij)}_{pq}$, one can solve the elasticity equations using a prescribed macroscopic strain, which leads to displacement fields χ^{kl}. Based on these displacement fields, the locally varying strain fields $\varepsilon^{(ij)}_{pq}$ can then be computed using

appropriate strain-displacement relations.

$$\int_V E_{pqrs}\varepsilon_{ij}(v)\,\varepsilon_{pq}\left(\chi^{kl}\right)dV = \int_V E_{pqrs}\varepsilon_{ij}(v)\,\varepsilon_{pq}^{0(kl)}dV\forall v\varepsilon V \tag{3.3}$$

The macroscopic elasticity tensor E^H_{ijkl} can be computed using the theory of homogenization, where v denotes a virtual displacement field. In practice, numerical homogenization is commonly employed to solve equation 3.3 by discretizing it using techniques such as the finite element method.

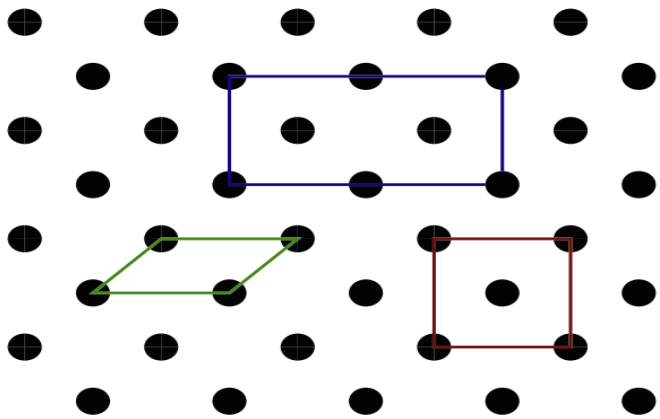


Figure 3.1.1: A section of a microstructure consisting of two materials(white and black)[2]

We have computed the homogenized elasticity tensor for a two material composite. A detailed description of the implementation is provided in the next section.

3.2 MATLAB implementation

The homogeneity of a two-material composite is the focus of the Matlab implementation that is provided here. Bilinear finite elements are utilised to discretize the unit cell, with plane strain elements being the default choice. However, plane stress may be chosen by supplying changed material data. When determining if a particular finite element has material 1 or material 2, an indicator matrix, represented by x, is utilised. 3.2.1 illustrates the indicator matrix x and the mesh construction. The Matlab code specifies the shape of the unit cell. The homogenization function needs eight inputs that the user specifies.

The width, height, and length of the unit cell are represented, respectively, by the first three inputs (l_x, l_y, l_z). A vector containing Lame's initial parameter for materials 1 and 2 is included in the fourth argument, λ. For the two materials, Lame's second parameter is a vector in the fifth input, *mu*. The angle between the horizontal axis and the left wall of the unit cell is represented by the sixth and seventh arguments, ϕ and θ, respectively. The indicator matrix x is the last parameter. With the number of rows equaling the number of items in the vertical direction and the number of columns equaling the number of elements in the horizontal direction, the discretization is dictated by the size of x. To prevent highly deformed parts, the angle ϕ should be provided in degrees and should not be lower than 45 or more than 135.

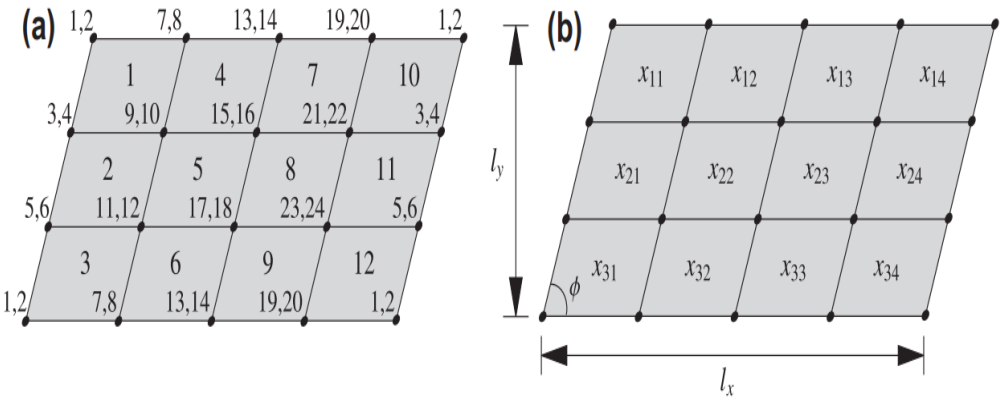


Figure 3.2.1: Figure (a) shows the finite element mesh used to discretize the unit cell, while Figure (b) shows the corresponding structure of the indicator matrix x.[3]

3.2.1 The element stiffness matrix and load vectors

The stiffness matrix of the discretized elasticity equation from 3.3 is obtained using the finite element method.

$$K = \sum_{e=1}^{N} \int_{V_e} B_e^T C_e B_e dV_e \tag{3.4}$$

The equation in 3.4 represents the assembly of N finite elements where the matrix B_e is the element strain-displacement matrix, V_e is the volume of element e, and C_e is the constitutive matrix for the element. The constitutive matrix C_e is for an isotropic material,

which we assume is the case for the materials used in building the composite.

$$C_e = \lambda_e \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \mu_e \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

The parameters λ_e and μ_e, used in the constitutive matrix for element e, are Lamé's first and second parameters for the material of the element, respectively. They can be determined from Young's modulus E and Poisson's ratio ν using the following equations:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \mu = \frac{E}{2(1+\nu)} \tag{3.6}$$

To obtain the plane stress properties, Lame's first parameter must be adjusted by a factor of $(1-\nu)$.

$$\hat{\lambda} = \frac{2\mu\lambda}{\lambda + 2\mu} \tag{3.7}$$

The element stiffness matrix is initialized by splitting it into two parts, where each part corresponds to the material properties of the respective element.

$$K = \sum_{e=1}^{N} k_e = \sum_{e=1}^{N} \left(\lambda_e k_\lambda + \mu_e k_\mu\right) \tag{3.8}$$

To simplify the extension to more materials, the element stiffness matrix is split into two parts, with the stiffness matrix being a function of the material properties in the elements. The loads f^i, which correspond to macroscopic volumetric straining, are discretized on the right-hand side of 3.3.

$$f^i = \sum_e \int_{V_e} B_e^T C_e \varepsilon^i dV_e \tag{3.9}$$

The load vectors are assembled using a split for the stiffness matrix:

$$f^i = \sum_e \left(\lambda_e f^i_\lambda + \mu_e f^i_\mu \right) \tag{3.10}$$

Finally, the displacement fields are computed solving the finite element problem with six load cases:

$$K\chi^i = f^i, i = 1, ..., 6 \tag{3.11}$$

The functions for computing $k_\lambda, k_\mu, f^i_\lambda, f^i_\mu$ have been completed. The elements are assumed to have identical shapes, which simplifies the implementation by executing the function only once. The displacement vectors χ^i are assumed to be V-periodic.

3.2.2 Degrees of freedom and periodic boundary conditions

The load vectors and global stiffness matrix are put together using the edofMat index matrix. The structure of edofMat would be as indicated in 3.2.2, where each row i (in bold) holds the global degrees of freedom related to element i. If the degrees of freedom were not periodic. We index into a periodic version of the mesh using the non-periodic edofMat to produce a periodic edofMat. The periodic edofMat's structure is seen in 3.2.2. Use of the same nodes on two different sides allows for the elimination-based enforcement of periodic boundary requirements. For a complete, regular grid, a periodic version of the grid is indexed using the edofMat matrix.

(a)

1	3	4	11	12	9	10	1	2
2	5	6	13	14	11	12	3	4
3	7	8	15	16	13	14	5	6
4	11	12	19	20	17	18	9	10
⋮	⋮	⋮	⋮	⋮				
12	31	32	39	40	37	38	29	30

(b)

1	3	4	9	10	7	8	1	2
2	5	6	11	12	9	10	3	4
3	1	2	7	8	11	12	5	6
4	9	10	15	16	13	14	7	8
⋮	⋮	⋮	⋮	⋮				
12	19	20	1	2	5	6	23	24

Figure 3.2.2: Structure of the edofMat-matrix for 12 element mesh in (a) a non-periodic version, and (b) periodic version. The bold numbers denote the row numbers and indicate which element the degrees of freedom in the row belong to[3]

3.2.3 Assembly of the stiffness matrix

Triplets are used to create the stiffness matrix for finite elements. First, edofMat is used to build vectors with row and column indices of non-zero items. Next, depending on the indicator matrix x, the element material characteristics λ_e and μ_e are assigned. The keLambda and keMu element matrices are multiplied by the respective element characteristics, and combined together, and then the triplets are used to put up the global stiffness matrix. It is crucial to remember that improper indicator matrix x values might result‘ in erroneous material property assignments.

3.2.4 Load vectors and solution

The procedure used to create the stiffness matrix is also used to assemble the load vectors. A linear solver is used to determine the value of χ^i in the resultant system in equation 3.11. Due to minute numerical variations, Matlab could sometimes fail to recognize K as a symmetric matrix and fall back on a more broad but slower linear solution.

3.2.5 Homogenization

The nodal displacements of an element that correspond to the unit strain cases are first determined by solving for the displacement of the element that corresponds to uniform strains while imposing enough restrictions to make the element stiffness matrix non-singular. Since all elements in the mesh are equal, the resultant element displacements are the same for all elements. Following the acquisition of the displacements, the following entries in the homogenised constitutive matrix C^H may be identified:

$$C_{ij}^H = \frac{1}{|V|} \sum_{e=1}^{N} \int_{V_e} \left(\chi_e^{0(i)} - \chi_e^{(i)} \right)^T k_e (\chi_e^{0(j)} - \chi_e^{(j)}) dV_e \tag{3.12}$$

Here, χ_e^0 comprises the three displacement fields corresponding to unit strains, while χ_e contains columns corresponding to the displacement fields resulting from globally enforcing the unit strains. The indices in parentheses denote the column number. $|V|$ refers to the unit cell volume. The summation in equation 3.12 is then calculated, where the summa-

tion is split into λ and μ parts that are multiplied by the corresponding element material properties before being combined. Once the homogenization is complete, the homogenized elasticity tensor is displayed. For meshes in which the elements differ in shape, and numerical integration must be performed for each element, the homogenization may be written as:

$$C^H = \frac{1}{|V|} \sum_{e=1}^{N} \int_{V_e} (I - B_e \chi_e)^T C_e (I - B_e \chi_e) dV_e \tag{3.13}$$

For meshes where the elements differ in shape, the homogenization process can be written using 3.4, where I is a six times six identity matrix. The term $B_e \chi_e$ can be interpreted as the strains caused by the non-homogeneous material distribution.

3.3 Convolutional Neural Network

Convolutional Neural Networks are versions of the multilayer perceptron that are influenced by biology. They have shown to be particularly successful in fields such as image recognition and classification. ConvNets have been used effectively in the identification of faces, objects, and traffic signs. They have also been used to successfully power the vision systems of robots and autonomous vehicles. CNNs make use of the spatially local correlation that exists between neurons of neighboring layers by establishing a local connection pattern between them. Put another way, the inputs of hidden units in layer m come from a subset of units in layer m-1, namely units that have receptive fields that are spatially adjacent to one another. There are a few fundamental building components of CNN:

- Convolutional layer (CONV)
- Activation layer (nonlinearity, for example, ReLU)
- Pooling or sub-sampling layer (POOL)
- Fully Connected layer (FC, using softmax)

The most typical configuration for a ConvNet involves stacking on many pairs of CONV-ReLU layers, each of which is then followed by a POOL layer. This process will continue

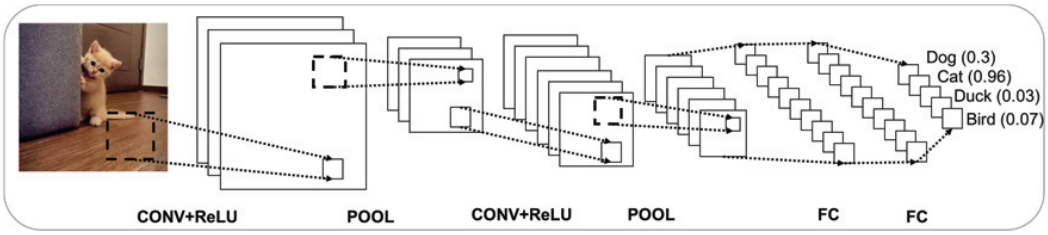


Figure 3.3.1: A typical convolutional neural network with CONV-ReLU, POOL layers repeated and followed by the fully-connected layers at the end.[3]

to recur until the whole of the input picture has been aggregated and spatially changed into a collection of little patches. Then, as it gets to the very last layer, it transitions into a fully-connected layer, which often uses softmax to output probabilities, particularly when it's a multi-way classification issue, as is seen in the picture that follows:

Neurons are the building blocks of neural networks, and each neuron in a neural network has its own weights and biases that it has acquired during training. The neurons that make up each layer of this network are diverse, as shown by the fact that it is structured into layers. Neurons in each layer are linked to neurons in the next layer by a set of edges that carry a weight that is learned from a training dataset. These weights are carried by the edges that connect the neurons in each layer to the neurons in the next layer. In addition, each neuron has an activation function that has been pre-selected. A neuron will create a response by computing its dot product with its learning weight for each and every input that it gets and then passing that information via its activation function. Despite the fact that this design performs well for datasets of moderate size, there is a problem with its scalability:

3.3.1 Convolution

The convolution process is one of the most significant concepts, and it is for this reason that CNN was given its name. A procedure known as convolution may be conceptualised as the interpolation of two signals using real-valued data. Convolution is a method that assesses the influence that one signal has on another signal when they are both integrated together. Finding patterns in signals is one of its principal uses, among other applications. One such example is a filter known as an averaging filter, which is used rather often in

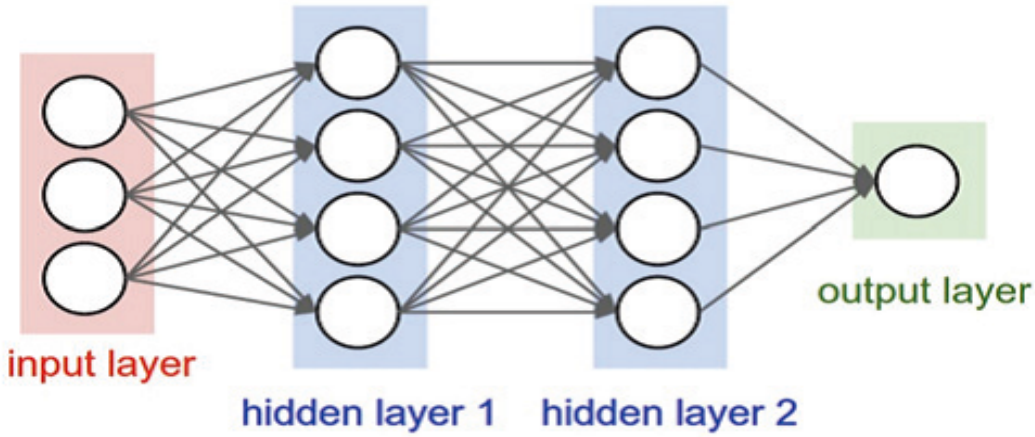


Figure 3.3.2: Architecture of a multilayer neural network.[3]

image processing. If the photographs that were recorded have extremely sharp edges, you could wish to apply a blurring effect to them by using the averaging filter, which is another name for this filter. To accomplish this look, convolution is often the technique that is used the majority of the time. When the matrix on the left, which contains the input data, is convolved with the matrix (kernel) on the right, an output, also known as a feature map, is produced. This is seen in the image entitled ”Example of convolution operation.”

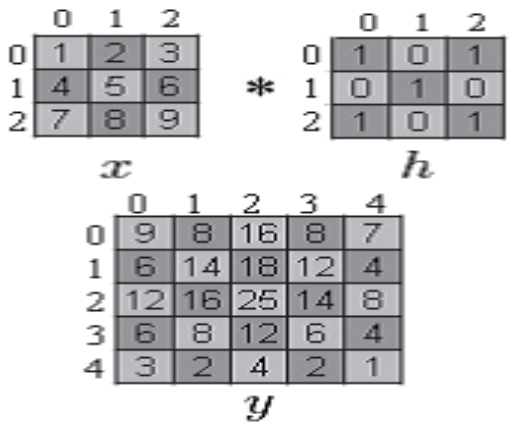


Figure 3.3.3: Example of convolution Operation[3]

3.3.2 Data Transformations

When implementing a CNN in the real world, data processing and transformation is often one of the most important steps in the process of reaching high accuracy.

3.3.3 Input Processing

Let's suppose that our dataset, X, has N photos with D flattened pixels per image. Typically, the following three processing stages are done on :

- Mean subtraction: In this stage, the mean image of the whole dataset is computed and subtracted from each picture. This action has the effect of centering the data over the origin along each dimension of the feature.

- Normalization: Typically, the mean subtraction step is followed by a normalisation step, which scales each feature dimension to the same scale. This is accomplished by dividing each column of features by their standard deviation.

- PCA whitening: Whitening using Principal Component Analysis is an additional modification step used for neural networks in general (PCA). Despite the fact that this technique is not often employed with CNNs, it is an essential step worth addressing here. Whitening is the process of decorrelating data by calculating the covariance matrix and utilising it to decrease the data's dimensionality to the highest principal components, as desired.

3.3.4 Data Augmentation

Intelligently enhancing the training data is one of the most prevalent techniques for increasing recognition performance. There are several ways to get this result:

- Translation and rotation invariance: For a network to learn translation and rotation invariances, perspective-transformed pictures are commonly included to a training dataset. You may flip a training dataset picture horizontally, for example. Other transformations include horizontal flips and pixel translations.

- Scale invariance: CNNs can't distinguish things at various scales. To remedy this, random cropping of input photographs should be included to the training set. Random crops subsample training pictures. You may also upsample random cropping to the input image's original size.

- Color perturbation: Changing the input image's colour values is an interesting data transformation.

3.3.5 Network Layers

A typical CNN architecture contains of layers that turn an input image tensor into an output tensor. Each layer may have many classes. Each network layer has a function. Sample ConvNet design demonstrates a network with input, convolutional, pooling, and fully connected layers (FC). [INPUT -¿ CONV -¿ POOL -¿ FC] is a common ConvNet design.

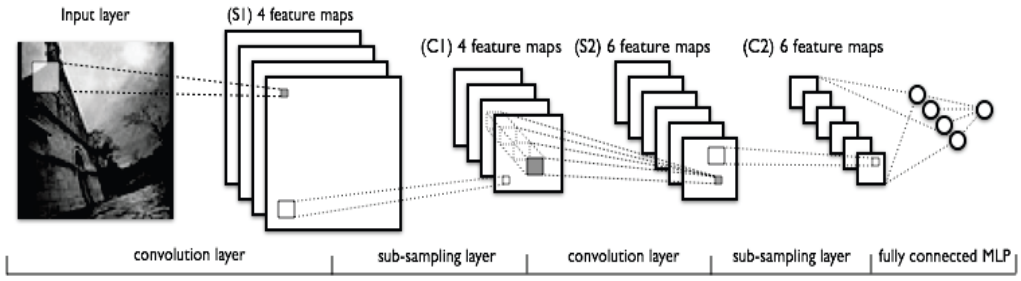


Figure 3.3.4: Sample CovNet architecture[3]

3.3.6 Convolution Layer

The convolutional layer applies a convolution filter on an input image. The local connectivity parameter of the layer defines each sub-region of the picture for this filter. Each filter application provides a scalar value for a given pixel position, which is called a feature map.

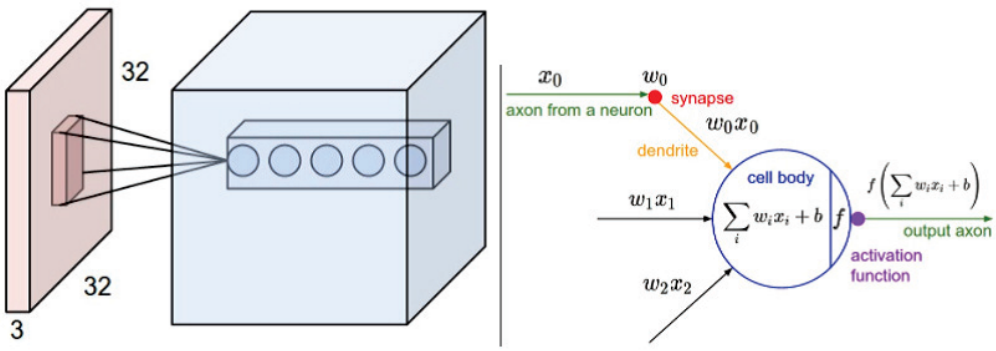


Figure 3.3.5: Example of Convolutional Layer[3]

This filter is a learnable model parameter over a specified training dataset. As such, designing this filter is crucial to ensure network performance. A 5x5 filter is typically used. Combining such filters over the whole input volume creates a powerful feature detector. Such filters are easy to train. First, choose the number and size of filters for each convolution layer. Random starting values are selected for these filters during training. During the backpropagation algorithm's forward pass, each filter is convolved at every possible pixel value to build a feature map. These feature maps are the input image tensor for succeeding layers, which extract higher-level image characteristics from the raw picture.

3.3.7 Pooling or Subsampling Layer

CNN layers commonly follow convolution with pooling or subsampling. It downsamples the convolution layer's height and width output. Pooling laters minimise the network's learning load. This reduces overfitting, improving network speed and accuracy. Pooling processes vary. Pooling methods include:

- Max pooling: Feature maps for each pooled region (2 x 2 in the preceding example) are replaced by a single value, the maximum of the four values within the pooled area.
- Average pooling: Feature maps for each pooled region (2 x 2 in the previous example) are replaced by a single value, the average of the four values within the pooled area.

3.3.8 Fully connected or dense layer

Fully connected layers are usually the last layers of a CNN. This layer's neurons are related to all prior activations. The number of neurons in this layer generally matches the number of classes. A CNN converts an input picture to class scores using the described layers. Each layer has various parameter needs. Back propagating a gradient descent technique teaches these layers' parameters.

3.3.9 Network Initialization

CNN training includes network initialization, which seems easy but is critical. Every CNN layer has training-set-trained parameters or weights. SGD is the most prevalent weight-learning method. SGD requires weights, a loss function, and labelled training data. SGD uses starting weights to calculate a loss value given training data labels and adjusts its weight to lower the loss. This changed weight will now be supplied to the next iteration where the same procedure repeats until convergence is reached. The starting weight for network initialization affects the quality and speed of network training convergence. To overcome this problem, many solutions have been used. Few examples are as follows:

- Randomize: All weights in this method are allocated randomly. Random assignment should use a zero-mean, unit-standard-deviation Gaussian distribution. The principle behind randomization is simple, imagine if the network had extremely similar or equal weights given to each neuron, every neuron will compute precisely the same loss value and execute the same gradient update at every iteration. This implies every neuron will learn identical characteristics and the network won't be varied enough to discover interesting patterns from the input. To ensure diversity in a network, random weights are utilised. This ensures asymmetric weighting, resulting to heterogeneous network learning. Random initialization's network variance is tricky. Randomly defining weights increases the variance of neuronal output. Normalize weights using layer inputs.

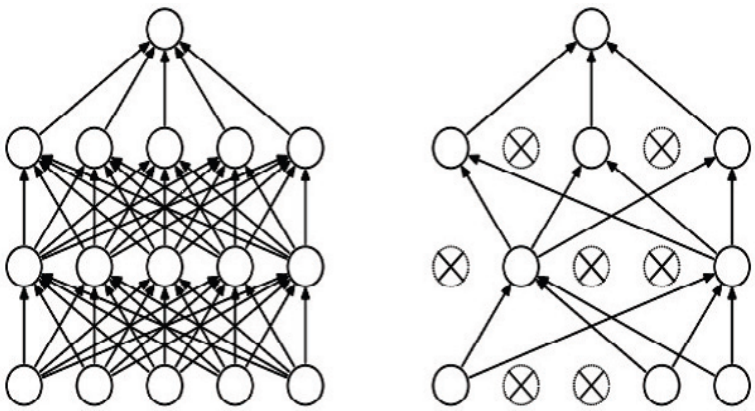
- Initialization sparse: Our approach randomly connects a neuron to K neurons in the preceding layer. As explained, neuron weights are random. The objective is to isolate the amount of connections to each unit from preceding levels. In this situation, initialise biases to 0. In ReLU, consider a tiny constant like 0.01 to propagate gradients.

- Batch Normalization: Batch normalisation was invented to withstand inadequate network initialization. This approach forces the network to normalise on a unit

Gaussian distribution at the start of training. It uses and to build a batch normalised version.

3.3.10 Reguarization

One of the issues in training CNNs is overfitting. Overfitting is when CNN, or any learning system, does well optimising training error but not test data. The most common trick used in the community to handle this problem is regularisation, which is basically adding a penalty to the loss function being optimised. There are several approaches of regularising the network.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.3.6: Example of Dropout based regularization[3]

3.3.11 Loss Functions

A CNN is taught using a gradient descent-based technique that minimises a loss function given training data. This loss function may be chosen in different ways.

- Cross-entropy: This is one of the most widely utilised loss functions for CNNs. It's based on cross-entropy, a distance metric between a true distribution p and approximated distribution q.

- Hinge Loss: It can be simply defined as the hinge loss is a loss function used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for support vector machines (SVMs).

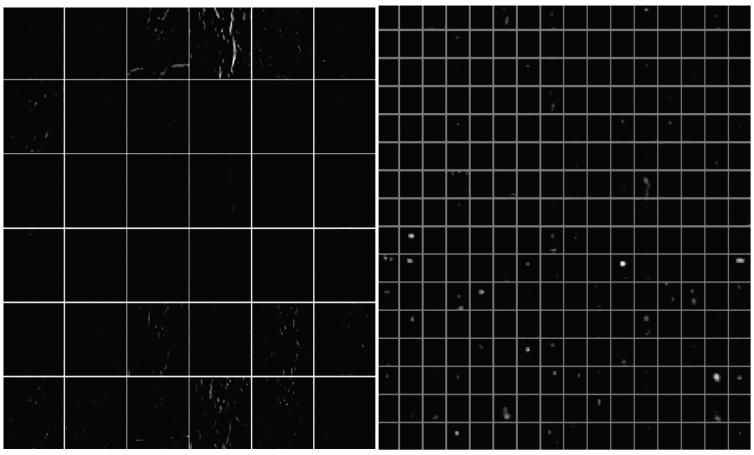


Figure 3.3.7: Visualizing activation of layers[19]

3.3.12 Model Visualization

CNNs learn feature maps or filters that operate as feature extractors on natural scene images once trained. As such, it would be great to display these filters and have an insight of what the network has learned through its training. This is a burgeoning field with several tools and approaches to view CNN's trained filters. The network has two interesting parts:

- Layer activation: This is the most typical network depiction, which shows neuron activity during the forward pass. Multiple reasons make this visualisation essential. It shows how each learnt filter affects each picture. You may debug the network by checking whether most filters are learning relevant features or are blank pictures, indicating training difficulties. This process is shown in Visualizing layer activations.

- Filter visualisation: Visualizing filter values is another typical use. CNN filters may be displayed as feature detectors to show what image features each filter can extract. The image Visualizing layer activations shows how CNN filters may be trained to identify and extract edges in multiple orientations and colours. This approach may also identify noisy filter values to indicate poor network training quality.

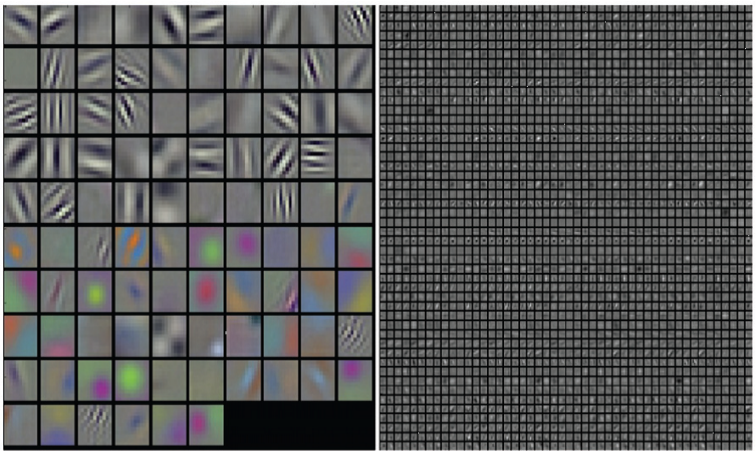


Figure 3.3.8: Visualizing trained filters[19]

3.4 Implementation of 3D Convolution Neural Network

3.4.1 3D Convolutional Neural Network

The convolutional neural network, often known as CNN or ConvNet, was first offered as a solution for issues relating to computer vision. In the 1990s, one of the very first CNNs that could effectively detect handwritten digits was developed. The less powerful computing capabilities available at the time restricted the applications that could be used using CNNs. The CNN method has seen a resurgence in popularity in recent years as a result of significant developments made in computer hardware in recent years, such as the general purpose graphics processing units (GPUs). The weights sharing process used in the CNN is what sets it apart from the more traditional FC-NNs. In this study, we propose a 3D-CNN architecture 3.4.1 for inferring homogenised or effective material properties (such as elastic moduli, shear moduli, and Poisson's ratio) from given microstructure configurations (such as discretized distribution of material phases). This architecture allows us to infer homogenized or effective material properties from given microstructure configurations. As its source of data, the 3D-CNN uses the phase voxels that have been preprocessed. The subsequent many convolutional layers, together with the 3D convolution filters and the pooling process, make up the crucial component of the CNN. The three-dimensional filter 3.4.2 does a sweep across the phase voxels. It creates the feature map by using the convolutional process, often known as the dot product of the tensor.

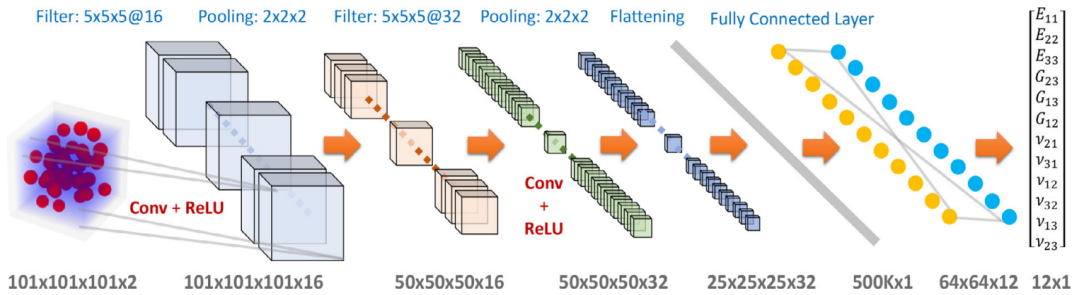


Figure 3.4.1: Proposed 3D-CNN architecture for effective properties prediction of heterogeneous materials.[19]

Each filter's weights and biases are trained to extract the most important characteristics from the input data. This process is called "training". Convolutional operations may often

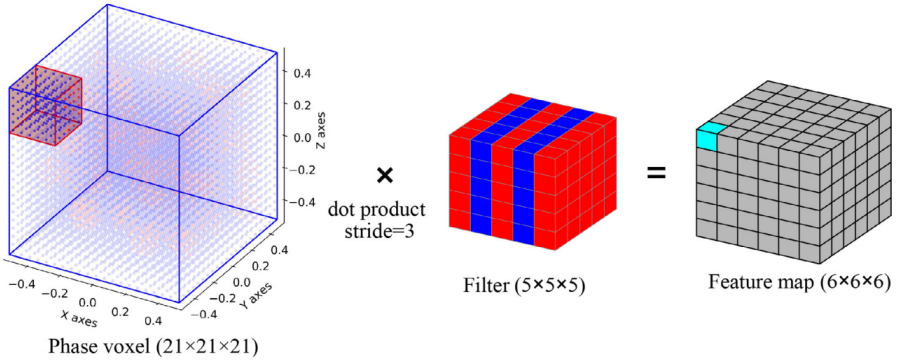


Figure 3.4.2: Convolution operation in the 3D-CNN model.[19]

be defined by a number of hyperparameters, including stride, padding, and filter size. The term "stride" refers to the amount of each successive step that filters make. For example, if the stride length is set to 1, this indicates that the filters will scan the volume one voxel at a time. It is advantageous to pad the input with voxels that have a value of zero since this helps to maintain the spatial size of the output. If the convolution operations are carried out with a stride of 1 and 2-layer zero padding. When building a CNN, pooling layers are often included in between the subsequent convolutional layers 3.4.2. By continuously downsampling the voxel value, it brings the spatial size of the data to a smaller and smaller size. The greatest value or the average value of a volume may be determined by pooling processes 3.4.3. The max-pooling operation's functionality with a volume size of 2x2x2. The activation layers are used to provide the CNN some nonlinearity, which is achieved by their application. The activation function will conduct an elementwise

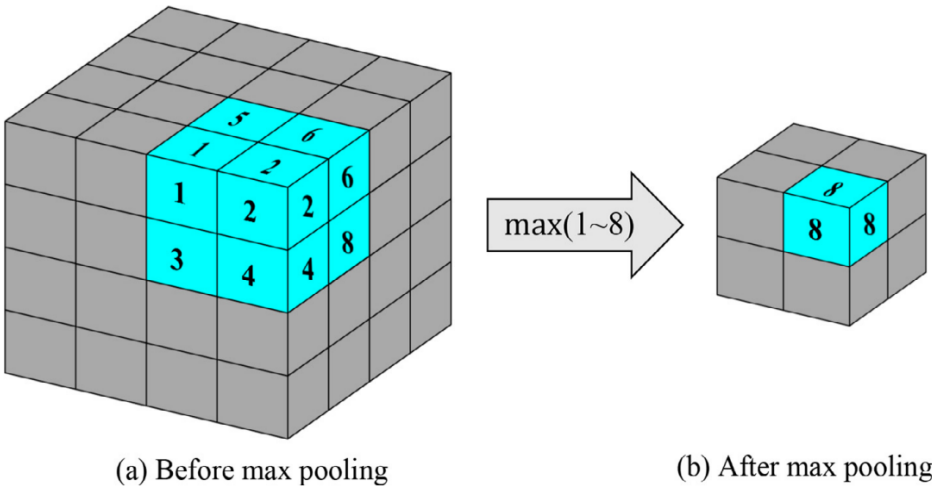


Figure 3.4.3: Max pooling operation in the 3D-CNN model.

nonlinear modification on the input tensor before returning the result. Rectified Linear Unit (ReLU) $f(x) = max(0,x)$, Sigmoid function $f(x) = 1/(1+e^{-x})$, and tanh function $f(x) = tanh(x)$ are some examples of typical activation functions.

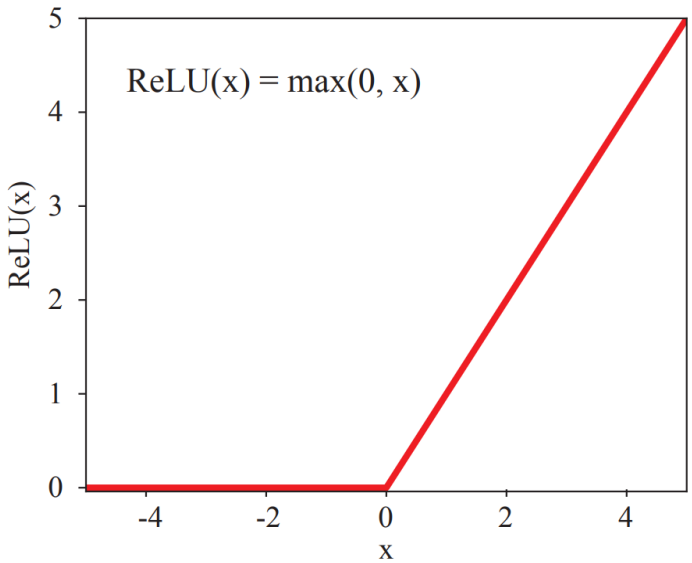


Figure 3.4.4: Rectified linear unit (ReLU) function as the activation function[19]

In comparison to the Sigmoid or tanh functions, the ReLU 3.4.4 function is favoured among these non-linear functions due to the fact that it has inexpensive arithmetic operations and excellent convergence properties on the stochastic gradient descent (SGD) algorithm. This is what leads to its selection as the function of choice. Below shown is the mathematical statement that may be stated to describe the output value at location (x,

y, z) on the jth feature map in the ith 3D convolutional layer.

$$\gamma_{j,xyz}^{(i)} = \mathrm{ReLU}\left(b_j^{(i)} + \sum_{m=1}^{M^{(i-1)}} \sum_{p=0}^{p^{(i)}-1} \sum_{q=0}^{p^{(i)}-1} \sum_{r=0}^{K^{(i)}-1} w_{jm,pqr}^{(i)} \gamma_{m,(x+p)(y+q)(z+r)}^{(i-1)}\right) \tag{3.14}$$

where ReLU(•) denotes elementwise ReLU function; $b_j^{(i)}$ is the common bias for jth feature map; $w_{jm,pqr}^{(i)}$ is the (p, q, r)th value of the 3D filter for jth feature map at ith layer associated with the mth feature map in the $(i-1)^{th}$ layer;$M^{(i-1)}$ is the number of feature maps at $(i-1)^{th}$ layer; $P^(i)$, $Q^(i)$ and $R^(i)$ denotes the size of the 3D filter at i^{th} layer.

Fully connected (FC) layers are employed at the end of the 3D-CNN where neurons between two neighboring layers are interconnected. FC layers take the flattened tensor from the previous hidden layer as the input and map them to desired output which are exactly the vector of effective material properties with length of 12. The connection between two adjacent layers, here from $(i-1)^{th}$ to i^{th}, can be expressed concisely in the form of tensor operations, given by

$$\gamma^{(i)} = \sigma\left(\mathrm{W}^{(i)}\gamma^{(i-1)} + \mathrm{b}^{(i)}\right) \tag{3.15}$$

where $\gamma^{(i-1)}$ and $\gamma^{(i)}$ represent the ith layer's input and output, respectively, and $\sigma(\cdot)$ stands for the activation function functioning elementwise;The weight matrix and bias vector between the ith and the $(i-1)^{\mathrm{th}}$ FC layers are denoted by the symbols $\mathrm{W}^{(i)}$ and $\mathrm{b}^{(i)}$, respectively. The 3D-CNN's trainable parameters are the weights and biases in the FC layers. The loss function is adopted as the mean square error (MSE) between the 3D-CNN's prediction and the training dataset's ground truth, which is provided by

$$L(\mathbf{W}, \mathbf{b} \mid \mathbf{D}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1}^{12}\left(\mathrm{y}_{kl}^{\mathrm{truth}} - \mathrm{y}_{kl}^{\mathrm{pred}}\right)^2 \tag{3.16}$$

where D stands for the training data set, $(\mathrm{x_k}, \mathrm{x_k}), n$ stands for the total number of samples, and l stands for the index of components for the effective properties vector. By reducing

the loss function, one may derive the ideal parameters W^*, b^*, namely,

$$\{\mathbf{W}^*, \mathbf{b}^*\} = \text{argmin}_{\{\mathbf{W},\mathbf{b}\}}\{L(\mathbf{W}, \mathbf{b} \mid \mathbf{D})\} \tag{3.17}$$

A common issue is overfitting, which is brought on by DNN's excellent approximation capabilities. First, there may be difficulties with optimization if the elastic (or shear) modulus and Poisson's ratio have different scales. A wide output variable range might result in very high error gradient values, which would make learning unstable by causing weight values to change quickly. Label rescaling is used to resolve this problem. Poisson's ratios and elastic (or shear) moduli are min-max scaled from 0 to 1. For instance

$$\hat{y} = \frac{y - min(y)}{max(y) - min(y)} \tag{3.18}$$

where y denotes the output component vector while $\hat{y}$ is the corresponding scaled output. In addition to label rescaling, early stopping and sample shuffling during training are adopted as the regularize to alleviate overfitting.

In this study, the convolutional layers are set up with a filter size of 5 x 5 x 5, a stride length of 1, and no padding. The pooling layer is specified to allow a maximum pooling of size 2 x 2 x 2. Due to the aforementioned benefits, the ReLU function is chosen as the activation function. Parametric tests are used to choose other hyperparameters, like the number of filters, the depth of convolutional layers, and FC layers. The training of 3D-CNN models uses the Adam algorithm, which optimizes adaptive learning rates.

Chapter 4

RESULTS AND DISCUSSION

First section explain the architectural design of the 3-D CNN model. Second section mentions how effective properties were predicted using the model.

4.1 Design of 3-D CNN Architecture

The assessment of a 3D-CNN for homogenizing heterogeneous materials is covered in this portion of the article. A number of parametric tests were run on the network's hyperparameters, such as filter size, depth, and width, to find the best design for the job. Then, using a dataset of 60 Representative Volume Elements (RVEs), the 3D-CNN was trained to forecast the effective qualities of these elements. By comparing the findings of the 3D-CNN with those produced using conventional FEA, the performance of the 3D-CNN was assessed. Additionally, uncertainty quantification was also out on a separate dataset that replicated uncertain inputs in order to account for the unpredictability of inclusion distribution in natural heterogeneous materials. A fresh dataset of RVEs with various inclusion shapes was used to test the transferability of the trained 3D-CNN model. The high-level neural networks API, Keras, and Python 3.7 were used to create the suggested 3D-CNN architecture. Such hyperparameters as filter size, the filter number, learning rate, number of hidden layers, and batch size all impact how well CNNs learn. Due to the extensive sample space, determining the ideal combination of these hyperparameters might be difficult. As a result, they are often searched in a limited sample space through trial and error. The number of filters in the convolutional layer may be chosen dependent on the complexity and quantity of samples of the input, for example. The selection of hyperparameters should be made based on the performance on the validation dataset since the number of fully connected (FC) layers and neurons directly influences the network's parameters and representational power. Various hidden layer and filter counts were exam-

ined for this work, and the performance of each was assessed using Mean Squared Error (MSE) on the validation dataset. A 0.05 mm-radius inclusion may be captured thanks to the input Cartesian grid's size of 101 x 101 x 10. During training, a batch size of 25 was employed, with a fixed filter size of 5 in each of the three dimensions. Using early stopping, the trained model with the lowest MSE for each architecture after four epochs was kept. The 3D-CNN architecture with certain hyperparameters was chosen for the remainder of the research based on accuracy and efficiency.

4.2 Prediction of effective properties

We assess the efficacy of their 3D-CNN model for homogenizing heterogeneous materials in this section. To establish an appropriate architecture for the present application, they performed parametric tests on network hyperparameters including filter size, depth, and width. The authors analyse the model's performance by contrasting the outcomes with conventional Finite Element Analysis (FEA) by using the trained 3D-CNN to predict the effective characteristics on a testing dataset composed of 60 Representative Volume Elements (RVEs). The authors perform uncertainty quantification on a separate dataset that simulates the input with uncertainty in order to account for the unpredictability of the inclusion distribution in naturally occurring heterogeneous materials. They also look at how well the trained 3D-CNN model can be applied to a fresh dataset of RVEs with various inclusion shapes. The authors used Python 3.7 and Keras, a high-level neural networks API, to implement the suggested 3D-CNN architecture. 60 RVEs with Volume Fraction (VF) values ranging from 2 to 28 make up the validation dataset. The trained model can predict the 12 elements of Young's modulus, shear modulus, and Poisson's ratio with high accuracy. On RVE samples with high and low VFs, the model performs equally well, while the latter show more unpredictability. To quantitatively assess prediction ability, the authors compute the mean absolute relative error (MARE) for each component. The MAREs for all 12 components are lower than 0.1. Since the binary point cloud is utilized to describe the RVE, the trained model encodes the topological information of the microstructure rather than the material characteristics. However, using the representation

of a point cloud that conveys the material attributes is an easy method to generalise the model.

	E_{11}	E_{22}	E_{33}	G_{23}	G_{13}	G_{12}	ν_{21}	ν_{31}	ν_{12}	ν_{32}	ν_{13}	ν_{23}
MARE	0.058	0.037	0.038	0.057	0.052	0.085	0.012	0.013	0.012	0.012	0.014	0.016

Table 4.2.1: Mean Absolute Relative Error on testing data set

The process of training the 3D-CNN model requires significant computational resources, taking around 3 hours on a GPU to achieve satisfactory results for the 200 RVEs in the training dataset. However, this computational expense is a one-time event, as once the model is trained, it can be used for inference on any new RVEs that are included in the dataset. Additionally, future work on the transferability of the trained 3D-CNN could reduce the need to retrain models for new datasets, improving the model's generalizability while reducing time and computational expenses. The predicted distributions of effective properties for three VF cases are shown in 4.2.2 and 4.2.3. It is obvious that the modulus components are positively correlated to the VF while the Poisson's ratio components are on the contrary, which are in accordance with the Voigt/Reuss models. In a word, the 3D-CNN's ability to reproduce the probabilistic distribution of the effective properties, together with its high computational efficiency, will make it a promising approach for probabilistic design of engineering composites.

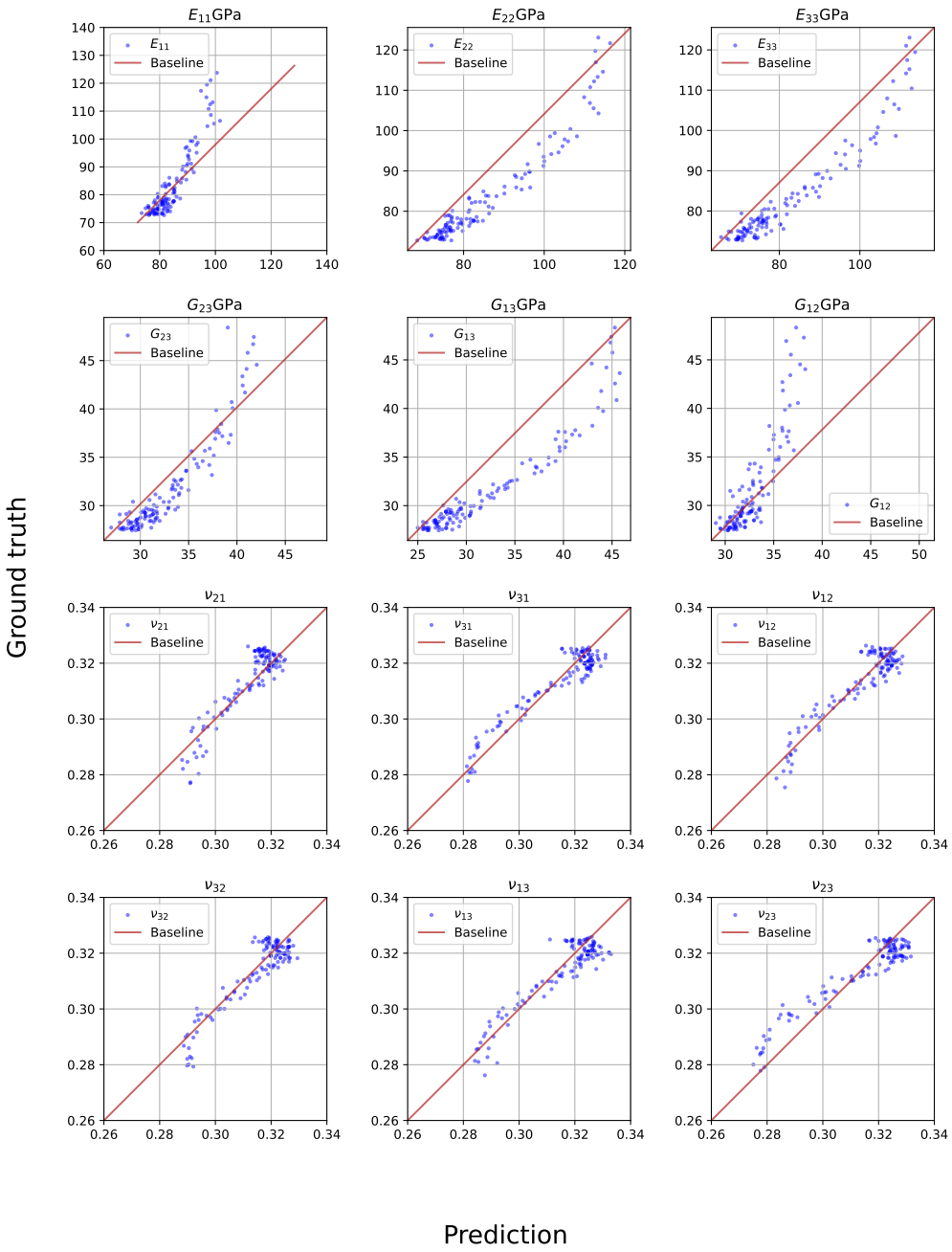


Figure 4.2.1: Comparison between the 3D-CNN prediction and ground truth (FEA)

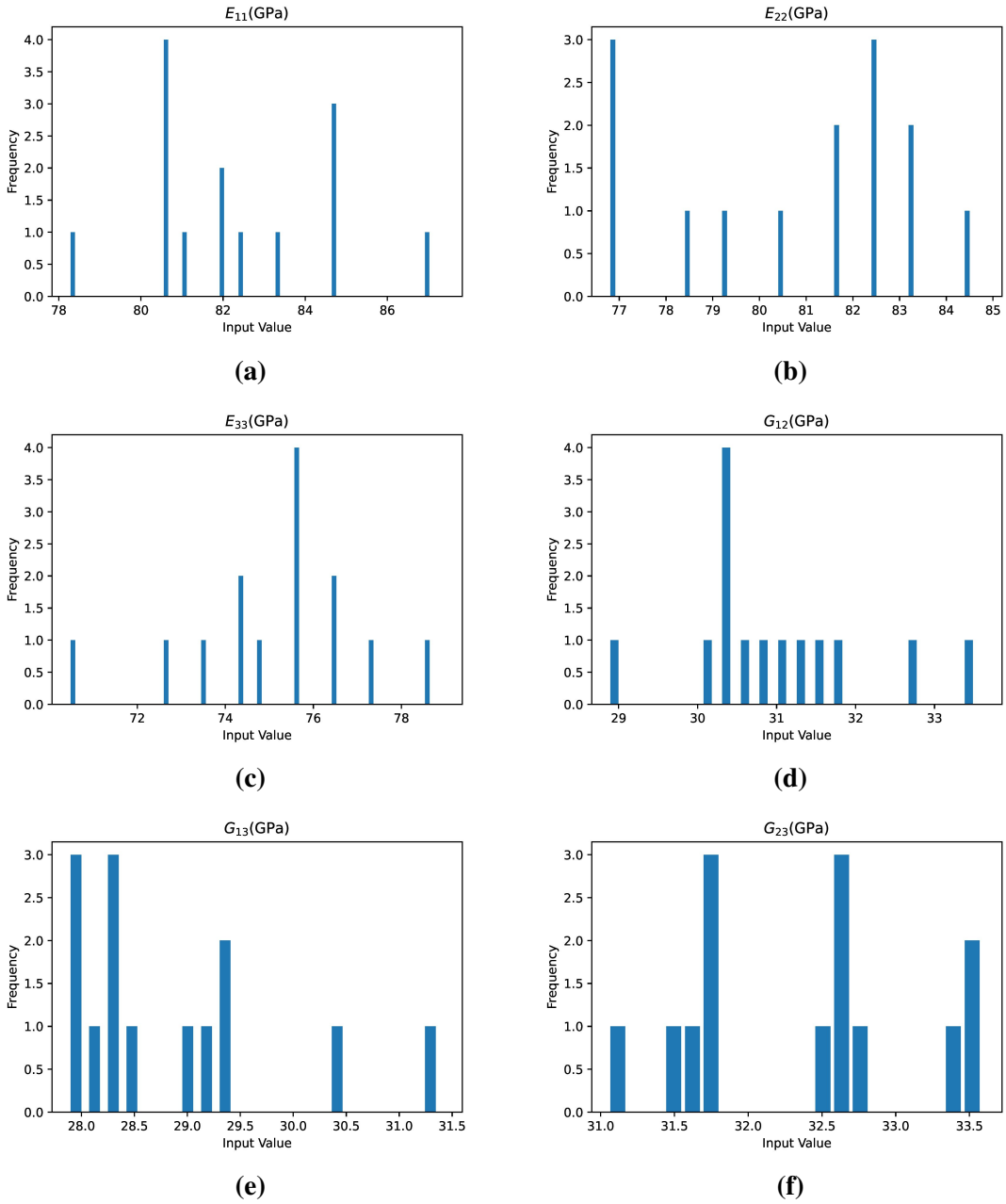


Figure 4.2.2: Histogram distribution of predicted effective properties for RVEs with mean VF of 0.07: (a) E_{11} (b) E_{22} (c) E_{33} (d) G_{23} (e) G_{13} (f) G_{12}

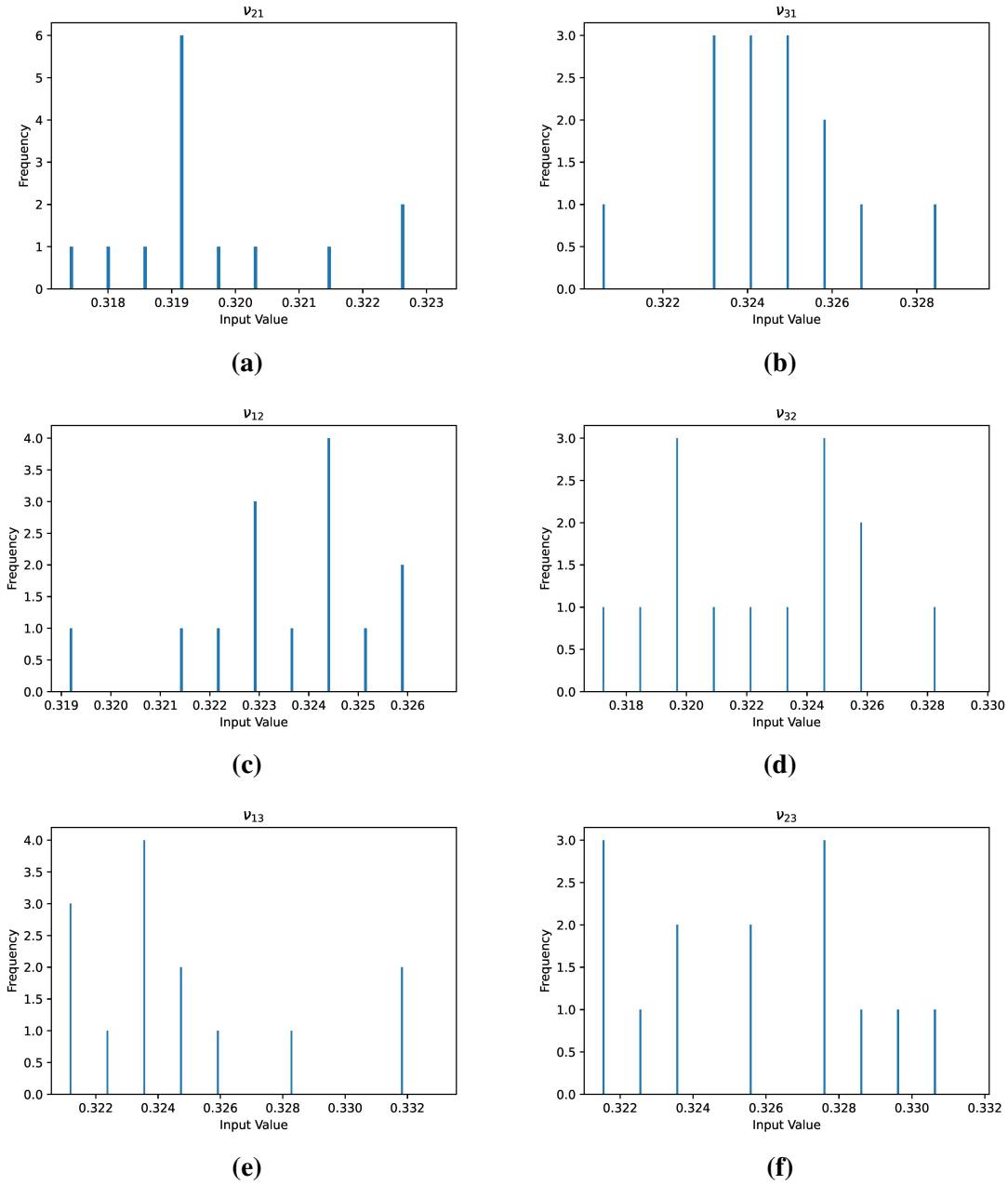


Figure 4.2.3: Histogram distribution of predicted effective properties for RVEs with mean VF of 0.07:(a) ν_{21} (b) ν_{31} (c) ν_{12} (d) ν_{32} (e) ν_{13} (f) ν_{23}.

Chapter 5

CONCLUSION AND FUTURE SCOPE

The article presents a novel method that uses a 3D-CNN to determine the effective properties of heterogeneous materials. This method is particularly focused on RVEs containing distributed particle inclusion. The design of the 3D-CNN includes various layers such as 3D convolution layers, pooling, flattening, and FC layers. In order to optimize the network architecture, a parametric study is conducted. The 3D-CNN is then trained with FEA-based linear homogenization. Through a test of accuracy, the proposed approach is shown to have the potential for efficient design and analysis of composite materials.

The 3D-CNN model developed in this study is shown to accurately and efficiently reproduce effective material properties when compared to traditional FEA. In addition to its accuracy and efficiency, the proposed method also offers benefits such as providing an end-to-end solution, preserving probabilistic distribution, and transfer learning for different types of composites. The proposed 3D-CNN model provides an end-to-end solution for effective property prediction that can be applied in different scenarios. The preservation of probabilistic distribution allows the model to be applied to different types of composites, which is particularly useful in cases where the distribution of inclusions or particles is different from those in the training dataset.

Future research for this method includes exploring transfer learning for other heterogeneous materials, extending the current 3D-CNN for nonlinear material properties, and applying the trained model for microstructure generation. Transfer learning could be a useful tool in expanding the applicability of the proposed method to other heterogeneous materials with different types of inclusions or particles. Extending the current 3D-CNN for nonlinear material properties would further enhance the capabilities of the model. Finally, applying the trained model for microstructure generation would be useful for designing materials with specific microstructures that have desired effective properties.

Bibliography

[1] Jacob Aboudi, Steven M Arnold, and Brett A Bednarcyk. *Micromechanics of composite materials: a generalized multiscale analysis approach*. Butterworth-Heinemann, 2013.

[2] Erik Andreassen and Casper Schousboe Andreasen. How to determine composite material properties using numerical homogenization. *Computational Materials Science*, 83:488–495, 2014.

[3] Anurag Bhardwaj, Wei Di, and Jianing Wei. *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018.

[4] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, 2017.

[5] C. Chen, D. Duhamel, and C. Soize. Probabilistic approach for model and data uncertainties and its experimental identification in structural dynamics: Case of composite sandwich panels. *Journal of Sound and Vibration*, 294(1):64–81, 2006.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[7] Xiaoping Du and Wei Chen. Efficient uncertainty analysis methods for multidisciplinary robust design. *AIAA journal*, 40(3):545–552, 2002.

[8] Frédéric Feyel. A multilevel finite element method (fe2) to describe the response of highly non-linear structures using generalized continua. *Computer Methods in Applied Mechanics and Engineering*, 192(28):3233–3244, 2003. Multiscale Computational Mechanics for Materials and Structures.

[9] Felix Fritzen and Oliver Kunc. Two-stage data-driven homogenization for nonlinear solids using a reduced order model. *European Journal of Mechanics - A/Solids*, 69:201–220, 2018.

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

[11] Akash Gupta, Ahmet Cecen, Sharad Goyal, Amarendra K. Singh, and Surya R. Kalidindi. Structure–property linkages using a data science approach: Application to a non-metallic inclusion/steel composite system. *Acta Materialia*, 91:239–254, 2015.

[12] Surya R Kalidindi. *Hierarchical materials informatics: novel analytics for materials data*. Elsevier, 2015.

[13] M. Kamiński. Boundary element method homogenization of the periodic linear elastic fiber composites. *Engineering Analysis with Boundary Elements*, 23(10):815–823, 1999.

[14] Konstantinos Kamnitsas, Christian Ledig, Virginia F.J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Daniel Rueckert, and Ben Glocker. Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, 2017.

[15] Nikhil Ketkar. *Introduction to PyTorch*, pages 195–208. Apress, Berkeley, CA, 2017.

[16] Giacomo Landi, Stephen R. Niezgoda, and Surya R. Kalidindi. Multi-scale modeling of elastic response of three-dimensional voxel-based microstructure datasets using novel dft-based knowledge systems. *Acta Materialia*, 58(7):2716–2725, 2010.

[17] C. Miehe, J. Schotte, and M. Lambrecht. Homogenization of inelastic solid materials at finite strains based on incremental minimization principles. application to the texture analysis of polycrystals. *Journal of the Mechanics and Physics of Solids*, 50(10):2123–2167, 2002.

[18] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[19] Chengping Rao and Yang Liu. Three-dimensional convolutional neural network (3d-cnn) for heterogeneous material homogenization. *Computational Materials Science*, 184:109850, 2020.

[20] András Reuß. Berechnung der fließgrenze von mischkristallen auf grund der plastizitätsbedingung für einkristalle. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):49–58, 1929.

[21] Haitham H. Saeed. Properties of polymer impregnated concrete spacers. *Case Studies in Construction Materials*, 15:e00772, 2021.

[22] R.J.M. Smit, W.A.M. Brekelmans, and H.E.H. Meijer. Prediction of the mechanical behavior of nonlinear heterogeneous systems by multi-level finite element modeling. *Computer Methods in Applied Mechanics and Engineering*, 155(1):181–192, 1998.

[23] A. Stiénon, A. Fazekas, J.-Y. Buffière, A. Vincent, P. Daguier, and F. Merchi. A new methodology based on x-ray micro-tomography to estimate stress concentrations around inclusions in high strength steels. *Materials Science and Engineering: A*, 513-514:376–383, 2009.

[24] Zijiang Yang, Yuksel C. Yabansu, Reda Al-Bahrani, Wei keng Liao, Alok N. Choudhary, Surya R. Kalidindi, and Ankit Agrawal. Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets. *Computational Materials Science*, 151:278–287, 2018.

[25] Gérard-Philippe Zéhil, Chady Ghnatios, and Rawad Himo. Soft computing approaches to homogenized properties of inclusion-modified concrete mixtures: Application to xlpe-modified concrete. *Journal of Building Engineering*, 29:101161, 2020.