Asymptotic Homogenization & Machine Learning

Conor Rowan

Spring 2023 Research Summary

Contents

1 Introduction

This semester of research focused on learning techniques used in multiscale analysis of solids, and exploring how data-driven methods might be incorporated into these multiscale analyses. More specifically, I studied the theory of asymptotic homogenization for the governing equations of linear elasticity,

implemented the microscale problem in MATLAB with finite elements, and used data generated from this solver to train/test different neural networks in Python's machine learning library "PyTorch." Because multiscale analyses of structures are typically very expensive, the use of machine learning would be to construct pre-trained surrogate models that streamline computations by eliminating finite element solves on the microstructure. Note that if the microstructure of a solid exhibits no spatial variation, then in the context of linear elasticity there is no use for machine learning. This is because the effective (macroscopic) properties of the material are independent of the applied strains by linearity of the microscale constitutive law, and thus can be computed once and applied everywhere in the solid. Essentially, this amounts to one additional step before turning to the macroscopic finite element solution. However, when the microstructure varies spatially, effective material properties must be computed for each realization of the microstructure which can be costly. A surrogate model which mapped the microstructure to the effective material properties would be beneficial in this setting. Additionally, we might be interested in the stress state of the microstructure in order to understand how small-scale material variations contribute to stress concentration. This could be important for understanding the influence of the microstructure on damage or fracture, for example. Analogous to the above, when the material is homogeneous linearity makes this analysis inexpensive, yet there is the need for efficiency when the microstructure is different at each integration point in the finite element mesh. The details of these observations will be discussed later–the point is that even in the framework linear elasticity, structures with spatially varying microstructures can be expensive to analyze. This is especially true in an optimization problem where the multiscale finite element problems needs to be solved repeatedly. The microstructure could itself be a design variable, in which case a huge number of microstructural finite element analyses would be required! Note that even for homogeneous material, the effective material properties depend on the macroscopic strains being applied in the context of non-linear elasticity. In other words, a surrogate model which mapped the applied strain to the effective material properties for a non-linear problem would be very useful. Thus, the goal of this report is to investigate how machine learning can be used to streamline multiscale finite element analysis of linear elastic structures. Comparisons to non-linear problems will be made, but the non-linear multiscale finite element problem will not be investigated here. To these ends, the theory of asymptotic homogenization will be developed and implemented in MATLAB. Next, two relevant neural network architectures will be discussed and implemented in PyTorch. Finally, a number of problems will be formulated, solved, and discussed.

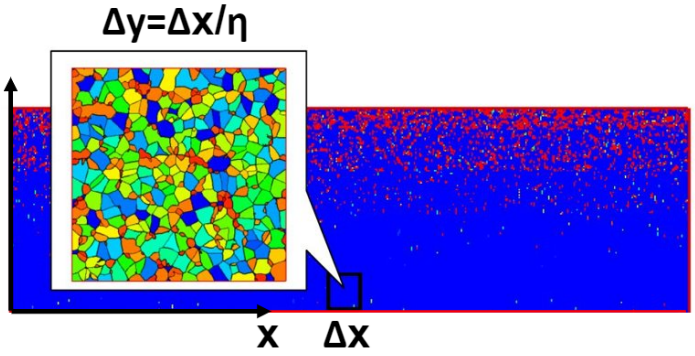

Figure 1: In the asymptotic expansion, the coordinates y and x are treated as independent, despite being defined in terms of one another. Through the scale parameter η, y resolves the microstructure by changing rapidly with the macroscopic coordinate x, which does not "see" the microstructure. Though theoretically infinite, η introduces an inherent length scale into the coordinate system. Note that by assumption, the microstructural coordinate y is periodic with period η.

2 Asymptotic Homogenization

2.1 Introduction

We wish to analyze a structure composed of heterogeneous material on a very small scale. This could be a metal which appears homogeneous and isotropic on the length scale of meters or centimeters, but is actually made up of crystals, pores, and grains on the micrometer scale. Asymptotic homogenization provides a framework to understand the contribution of the small-scale heterogeneities to the mechanics of the structure. For a typical coordinate x which represents position in space, we introduce a second coordinate y

$$x = \eta y \tag{1}$$

where $\eta << 1$ is a small parameter which controls the extent of separation between the macro- and micro-scales. See Figure 1 for a schematic. Small changes in the macroscopic coordinate x result in very large changes in y, which indicates that the fine details of the material in the structure can be better resolved by the "fast" variable y than the "slow" one x. With this, we conceptualize the problem as having two scales: the macroscopic scale defined by coordinate x, and the microscopic scale y whose separation from x is controlled by the parameter η. Thus, we write the domain Ω^η which is interpreted as the all the macroscopic points plus their accompanying microstructures, obtained by magnifying the material by a factor of η. For other quantities, a superscript η will indicate the quantity before being split into its microscopic and macroscopic parts. For all perturbation methods, it is assumed that the slow and fast

variables are independent, and that the quantities of interest can be expanded in powers of η. This seemingly bizarre assumption can be justified by noting that at each x, there will be a corresponding microstructure defined with y, and that changes in y have little influence on the macroscopic position. It is as if quantities defined in the macroscopic coordinate system change so slowly compared to the microscale that they appear constant. Furthermore, when we assume that the variations in the microstructure are periodic with period η, this is called "periodic homogenization." In this case, $\Delta x = \eta$ corresponds to $\Delta y = 1$ and a full traversal of the heterogeneous microstructure. We imagine the structure as being built up of a tessellation of these periodic microstructures. We will refer some instance of the repeating periodic microstructure as a random volume element (RVE). See Figure 2. Now, expand the displacement field in powers of η

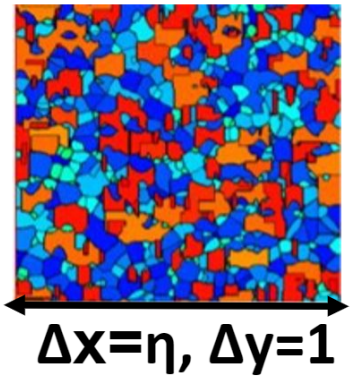

Figure 2: The dimensions of the RVE are controlled by the scale parameter η. The assumption of periodicity imposes certain boundary conditions on the deformation of RVE's which ensure that no gaps form in the structure. *Note that even when the microstructure is assumed to vary in space, the periodic homogenization framework is used. This is a contradiction!*

$$u^\eta(x,y) = u^0(x) + \eta u^1(x,y) + \eta^2 u^2(x,y) + \dots \tag{2}$$

Vector notation for the coordinates is not being used for convenience. This derivation applies to any number of spatial dimensions. In some sense, this step should be thought of as an ansatz of perturbation theory–it is something we basically take for granted. By assumption, the first term in the expansion does not depend on the microscopic coordinate[1]. See Figure 3 for an example of a function which is periodic on the fine scale. All perturbation analyses begin with this assumption. Our goal is to obtain some kind of governing equations

[1] If this assumption is not made, it adds additional steps the derivation but it can eventually be proven

for the multiscale problem. To this end, we look at the (infinitesimal) strain-displacement relations

$$\epsilon_{ij}^{\eta} = \frac{1}{2}\left(\frac{du_i}{dx_j^{\eta}} + \frac{du_j}{dx_i^{\eta}}\right) \tag{3}$$

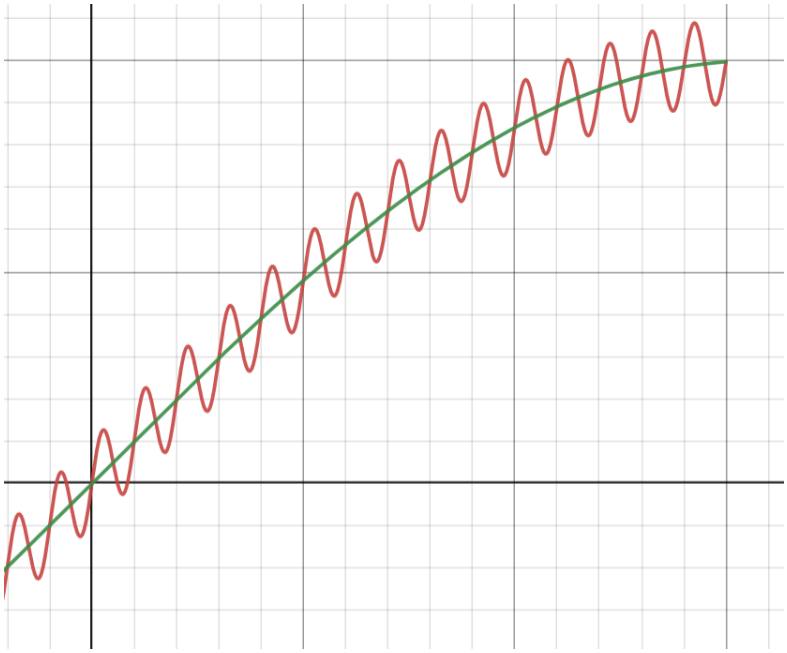

Figure 3: Plot of $f(x, x/\eta) = \sin(x) + \eta \sin(x/\eta)$ indicating how the perturbatively small parameter η controls the frequency and magnitude of the first correction term in an asymptotic expansion.

Derivatives are taken with respect to the two-scale coordinate x^{η}. Treating the two scales as independent coordinates, we expand this derivative with the chain rule

$$\frac{d}{dx_i^{\eta}} = \frac{\partial}{\partial x_i} + \frac{1}{\eta}\frac{\partial}{\partial y_i} \tag{4}$$

In a first-order perturbation theory, only the first two terms are retained in the asymptotic expansion of Eq. 2. Plugging in this truncated expansion to Eq. 3 and using Eq. 4, we have

$$\epsilon_{ij}^{\eta} = \frac{1}{2}\left[\left(\frac{\partial}{\partial x_j} + \frac{1}{\eta}\frac{\partial}{\partial y_j}\right)\left(u_i^0(x) + \eta u_i^1(x,y)\right) + \left(\frac{\partial}{\partial x_i} + \frac{1}{\eta}\frac{\partial}{\partial y_i}\right)\left(u_j^0(x) + \eta u_j^1(x,y)\right)\right]$$

$$= \frac{1}{2}\left[\frac{\partial u_i^0}{\partial x_j} + \eta\frac{\partial u_i^1}{\partial x_j} + \frac{\partial u_i^1}{\partial y_j} + \frac{\partial u_j^0}{\partial x_i} + \eta\frac{\partial u_j^1}{\partial x_i} + \frac{\partial u_j^1}{\partial y_i}\right]$$

Analogous to Eq. 2, we group terms in powers of η such that $\epsilon(x,y) = \eta^0\epsilon^0(x,y) + \eta\epsilon^1(x,y) + \ldots$ To simplify writing out these equations, use the notation $\epsilon_{k\ell}^z(f) := \frac{\partial f_k}{\partial z_\ell} + \frac{\partial f_\ell}{\partial z_k}$. We then have

$$\eta^0: \quad \epsilon_{ij}^0(x,y) = \epsilon_{ij}^x(u^0) + \epsilon_{ij}^y(u^1) \tag{5a}$$

$$\eta^1: \quad \epsilon_{ij}^1(x,y) = \epsilon_{ij}^x(u^1) \tag{5b}$$

Unlike the displacement, the first term in the asymptotic expansion of the strain field depends on both coordinates. This calculation illustrates many of the ingredients of the concepts and mathematics of the perturbation analysis: introduction of two coordinates, asymptotic expansion of displacement, expanding derivatives with the chain rule, and grouping quantities by powers of η. There are now two routes to obtaining governing equations–pursue the weak form directly from stress equilibrium, or derive the Navier equation in multiscale setting then weaken it. The former will be sketched, but passing through the Navier equation will prove to be much simpler conceptually.

2.2 Weak Form

Stress equilibrium in a multiscale body can be written as

$$\frac{d\sigma_{ij}^{\eta}}{dx_j^{\eta}} + b_i = 0 \tag{6}$$

As before, first order perturbation analyses are restricted to keep two terms in asymptotic expansions. Like the strain, the stress is written as

$$\sigma_{ij}^{\eta}(x,y) = \sigma_{ij}^0(x,y) + \eta\sigma_{ij}^1(x,y) + \ldots$$

Expanding the derivatives with chain rule per Eq. 4 and plugging in the expansion of the stress, we group the resulting equations in powers of η:

$$\eta^{-1}: \quad \frac{\partial \sigma_{ij}^0}{\partial y_j} = 0 \tag{7a}$$

$$\eta^0: \quad \frac{\partial \sigma_{ij}^0}{\partial x_j} + \frac{\partial \sigma_{ij}^1}{\partial y_j} + b_i = 0 \tag{7b}$$

These two equations could be weakened and used to obtain finite element formulations along with a constitutive relation defined in terms of the multiscale strains per Eqs. 5. This approach would be much more convenient for non-linear constitutive relations. However, one gets into tricky terrain arguing how to deal with boundary conditions and test functions in the multiscale setting. What functions do we test against? What coordinates are they defined in terms of, and how do we know this? The alternative approach of going through the Navier equation then weakening the solution is more intuitive and requires fewer leaps of logic.

2.3 Navier Equation

The Navier equation writes stress equilibrium in terms of displacements. Stress equilbrium in multiscale setting is given by Eq. 7, and derivatives are computed with Eq. 4. Strain is related to the displacement through Eq. 3, so we only require a constitutive relation to proceed. We will assume that the constitutive tensor only depends on the microscale coordinate, otherwise the math becomes extremely cumbersome. Thus, we have

$$\sigma_{ij}^\eta(x,y) = C_{ijk\ell}(y)\epsilon_{k\ell}^\eta(x,y) \tag{8}$$

Substituting this into stress equilibrium, we can use symmetries of the material tensor to simplify the resulting expression. Note that the body force only depends on the macroscale coordinate:

$$\frac{d}{dx_j^\eta} C_{ijk\ell}(y) \frac{du_k^\eta}{dx_\ell^\eta} = -b_i(x)$$

Substituting the definition of the multiscale derivative and the two-term aymptotic expansion for the displacement, this becomes

$$\left(\frac{\partial}{\partial x_j} + \frac{1}{\eta}\frac{\partial}{\partial y_j}\right) C_{ijk\ell}(y) \left(\frac{\partial}{\partial x_\ell} + \frac{1}{\eta}\frac{\partial}{\partial y_\ell}\right)\Big(u_k^0(x) + \eta u_k^1(x,y)\Big) = -b_i(x)$$

This expression can be expanded, and the terms grouped in powers of η. It is a length calculation to show that, when keeping the two lowest powers of η, the Navier equation is

$$\frac{1}{\eta}\left(\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial u_k^0}{\partial x_\ell}\right)+\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial u_k^1}{\partial y_\ell}\right)\right)+C_{ijk\ell}\frac{\partial^2 u_k^0}{\partial x_j\partial x_\ell} + C_{ijk\ell}\frac{\partial^2 u_k^1}{\partial x_j\partial y_\ell}+\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial u_k^1}{\partial x_\ell}\right)=-b_i \quad (9)$$

We obtain two governing equations by claiming that terms of equal powers of η must be equal individually. This results in

$$\eta^{-1}: \quad \frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial u_k^1}{\partial y_\ell}\right)=-\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial u_k^0}{\partial x_\ell}\right) \quad (10a)$$

$$\eta^{0}: \quad C_{ijk\ell}\frac{\partial^2 u_k^0}{\partial x_j\partial x_\ell}+C_{ijk\ell}\frac{\partial^2 u_k^1}{\partial x_j\partial y_\ell}+\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial u_k^1}{\partial x_\ell}\right)=-b_i \quad (10b)$$

The first equation only involves y derivatives. Because the macroscopic coordinate is assumed to not change with the microscale coordinate (scale separation assumption), the term $\partial u_k^0/\partial x_\ell$ is a constant with respect to y. Because this equation is linear, we can write

$$u_i^1(x,y)=\chi(y)_{imn}\frac{\partial u_m^0}{\partial x_n}(x) \quad (11)$$

The 3-index tensor function χ is interpreted as giving the displacement components of the RVE under the action of applied macroscopic unit strains. Because $u^1(x,y)$ is periodic in y, the process of solving for unit response will enforce periodicity in $\chi_{imn}(y)$. In order to solve for the RVE's response to unit strains, use Eq. 11 with $\partial u_k^0/\partial x_\ell=\delta_{ka}\delta_{\ell b}$ and the first governing equation:

$$\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial \chi_{kab}}{\partial y_\ell}\right)=-\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\delta_{ka}\delta_{\ell b}\right) \quad (12)$$

This is the governing equation for $\chi(y)$. A given value of indices a and b, yields an equation for the displacement components. Because the strain tensor is symmetry, not every combination of indices needs to be computed. Note that this equation has the form of stress equilibrium in the RVE for a body force proportional to spatial variations in the constitutive tensor. Turning now to the second of Eqs. 10, we can substitute Eq. 11 in order to remove any $u^1(x,y)$ dependence. Because the RVE unit response function χ only dependence on the microscale coordinate y and the first term in the displacement expansion u^0 only depends on x, we have

$$C_{ijk\ell}\frac{\partial^2 u_k^0}{\partial x_j\partial x_\ell}+C_{ijk\ell}\frac{\partial \chi_{kab}}{\partial y_\ell}\frac{\partial^2 u_a^0}{\partial x_j\partial x_b}+\frac{\partial}{\partial y_j}\left(C_{ijk\ell}\chi_{kab}\frac{\partial^2 u_a^0}{\partial x_b\partial x_\ell}\right)=-b_i \quad (13)$$

Notice how the assumption that $C_{ijk\ell}=C_{ijk\ell}(y)$ is used throughout–if the microstructure varied macroscopically, then the unit response χ would as well.

Eq. 13 cannot be satisfied pointwise in x and y. The body force and displacement gradients are purely macroscopic, whereas the constitutive tensor and unit response χ are purely microscopic. The macroscale coordinate x is essentially naive to the small scale variations in the structure's material and mechanics captured by y. Returning to the analogy of zooming into a complex microstructure at each x point, we argue that the microscale interacts with the macroscopic mechanics through an average. Thus, we average the influence of the microstructure and say the averaged multiscale equation is satisfied pointwise in x:

$$\left(\frac{1}{|\Omega^y|}\int_{\Omega^y} C_{ijk\ell}dy\right)\frac{\partial^2 u_k^0}{\partial x_j \partial x_\ell} + \left(\frac{1}{|\Omega^y|}\int_{\Omega^y} C_{ijk\ell}\frac{\partial \chi_{kab}}{\partial y_\ell}dy\right)\frac{\partial^2 u_a^0}{\partial x_j \partial x_b} + \left(\frac{1}{|\Omega^y|}\int_{\Omega^y} \frac{\partial}{\partial y_j}(C_{ijk\ell}\chi_{kab})dy\right)\frac{\partial^2 u_a^0}{\partial x_b \partial x_\ell} = -b_i \quad (14)$$

The third term on the left-hand side of this equation is the divergence of a periodic function (both the material tensor and RVE unit response are periodic). The divergence theorem can be used to show the integral of the divergence of a periodic function is zero. Introducing delta functions to manage indices, we obtain the following expression:

$$\left(\frac{1}{|\Omega^y|}\int_{\Omega^y} C_{ijk\ell}\left(\delta_{ka}\delta_{\ell b} + \frac{\partial \chi_{kab}}{\partial y_\ell}\right)dy\right)\frac{\partial^2 u_a^0}{\partial x_j \partial x_b} = -b_i \quad (15)$$

The integral is a four-index object with no x or y dependence. Eq. 15 is the Navier equation in terms of the macroscopic coordinate x for a constant constitutive tensor. Thus, we recognize the integral in parentheses as the homogenized material tensor:

$$C^H_{ijab} := \frac{1}{|\Omega^y|}\int_{\Omega^y} C_{ijk\ell}\left(\delta_{ka}\delta_{\ell b} + \frac{\partial \chi_{kab}}{\partial y_\ell}\right)dy \quad (16)$$

The homogenized tensor furnishes the effective material properties of a material which exhibits periodic heterogeneities on a small scale. It can be computed once the unit response of the RVE is known.

2.4 2D Finite Element Formulation

Eq. 12 is the governing equation for the displacement response of the RVE to applied unit strains. When this is known, the homogenized tensor can be computed per Eq. 16. Thus, we fish to formulate a finite element problem to compute $\chi_{kab}(y)$. Specifically, we fill focus on 2D isotropic materials with heterogeneous microstructures. The microstructure will vary spatially only through the Young's Modulus $E(y)$. Weaken the strong form of the microscale governing equation by integrating over the RVE against a test function δu_i:

$$\int_{\Omega^y} \frac{\partial}{\partial y_j}\left(C_{ijk\ell}\frac{\partial \chi_{kab}}{\partial y_\ell}\right)\delta u_i d\Omega = -\int_{\Omega^y} \frac{\partial}{\partial y_j}\left(C_{ijk\ell}\delta_{ka}\delta_{\ell b}\right)\delta u_i d\Omega \tag{17}$$

Per Figure 2, the microscale domain has dimensions of unity in the y coordinate system. Integrate by parts and note the periodic boundary conditions of the RVE imply there is no traction, thus the boundary term vanishes:

$$\int_{\Omega^y} C_{ijk\ell}\frac{\partial \chi_{kab}}{\partial y_\ell}\frac{\partial \delta u_i}{\partial y_j} d\Omega = -\int_{\Omega^y} C_{ijk\ell}\delta_{ka}\delta_{\ell b}\frac{\partial \delta u_i}{\partial y_j} d\Omega \tag{18}$$

In 2D, there are three independent components of stress and strain. This expression can be written in matrix form as

$$\int_{\Omega^y} \begin{bmatrix} \delta u_{1,1} \\ \delta u_{2,2} \\ \delta u_{1,2} + \delta u_{2,1} \end{bmatrix} \cdot \boldsymbol{D}(y) \begin{bmatrix} \chi^{\tilde{a}}_{1,1} \\ \chi^{\tilde{a}}_{2,2} \\ \chi^{\tilde{a}}_{1,2} + \chi^{\tilde{a}}_{2,1} \end{bmatrix} d\Omega$$
$$= -\int_{\Omega^y} \begin{bmatrix} \delta u_{1,1} \\ \delta u_{2,2} \\ \delta u_{1,2} + \delta u_{2,1} \end{bmatrix} \cdot \boldsymbol{D}(y)\hat{\boldsymbol{e}}_{\tilde{a}} d\Omega$$

In this expression, comma notation is used for derivatives, $\boldsymbol{D}(y)$ is the 2D constitutive relation, and the index $\tilde{a}$ corresponds to ab through $\tilde{a} = 1 \rightarrow (a,b) = (1,1)$, $\tilde{a} = 2 \rightarrow (a,b) = (2,2)$, and $\tilde{a} = 3 \rightarrow (a,b) = (1,2)$. The unit vector $\hat{\boldsymbol{e}}_{\tilde{a}}$ is used to apply the unit strains. Because only the Young's Modulus varies over the RVE, this can be written as

$$\int_{\Omega^y} E(y) \begin{bmatrix} \delta u_{1,1} \\ \delta u_{2,2} \\ \delta u_{1,2} + \delta u_{2,1} \end{bmatrix} \cdot \boldsymbol{D}^{unit} \begin{bmatrix} \chi^{\tilde{a}}_{1,1} \\ \chi^{\tilde{a}}_{2,2} \\ \chi^{\tilde{a}}_{1,2} + \chi^{\tilde{a}}_{2,1} \end{bmatrix} d\Omega$$
$$= -\int_{\Omega^y} E(y) \begin{bmatrix} \delta u_{1,1} \\ \delta u_{2,2} \\ \delta u_{1,2} + \delta u_{2,1} \end{bmatrix} \cdot \boldsymbol{D}^{unit}\hat{\boldsymbol{e}}_{\tilde{a}} d\Omega \tag{19}$$

Per the usual 2D finite element procedure, the elemental stiffness matrix can be written in terms of a 3×8 matrix of derivatives $\boldsymbol{B}$ which takes the elemental degrees of freedom to the finite element approximation of the strain field in the element:

$$\boldsymbol{\epsilon^h}(\delta u) = \begin{bmatrix} \delta u^h_{1,1} \\ \delta u^h_{2,2} \\ \delta u^h_{1,2} + \delta u^h_{2,1} \end{bmatrix} = \boldsymbol{B} \begin{bmatrix} \vdots \\ dofs \\ \vdots \end{bmatrix}$$

The elemental stiffness matrix is then

$$\boldsymbol{K}^e = \int_{\Omega^e} E(y)\boldsymbol{B}^T\boldsymbol{D}^{unit}\boldsymbol{B}d\Omega$$

Now, the forcing term is specific to the governing equation for the microscale finite element problem. The microstructure is loaded with a constant volumetric strain. The elemental force vector for this problem is

$$\boldsymbol{f}^e = -\int_{\Omega^e} E(y)\boldsymbol{B}^T\boldsymbol{D}^{unit}\hat{\boldsymbol{e}}_{\tilde{a}}d\Omega \tag{20}$$

These quantities can be inserted into their global counterparts with a typical finite element assembly procedure. Note that the 2D finite element problem is solved on a square of side length 1 and with periodic boundary conditions. *The periodic boundary conditions can be enforced by enforcing zero displacement at the four corners, and equating displacement vectors of nodes on opposite sides of the domain.* Now that we can solve for the RVE's response to unit strains, we can compute the homogenized tensor with Eq. 16. Collapsing indices to cast this into matrix/vector notation, we have

$$D^H_{ij} = \frac{1}{|\Omega^y|}\int D_{ij} + D_{ik}\epsilon_k(\chi^j)d\Omega = \frac{1}{|\Omega^y|}\int D_{ij} + \sigma^j_i(y)d\Omega \tag{21}$$

where $\sigma^j_i(y)$ is defined as the component i of the stress tensor at position y arising from applied unit strain j. The constitutive relation is represented by matrix D_{ij} to distinguish from the four index tensor $C_{ijk\ell}$. Numerically, we can store the Young's Modulus as constant over each element (but varying between elements). In this case, the homogenized tensor can be computed with

$$\boldsymbol{D}^H = \frac{1}{|\Omega^y|}\sum_{n=1}^{n_{el}}\Delta A^e\left(E(y_n)\boldsymbol{D}^{unit} + \begin{bmatrix}\sigma^1_1 & \sigma^2_1 & \sigma^3_1\\ \sigma^1_2 & \sigma^2_2 & \sigma^3_2\\ \sigma^1_3 & \sigma^2_3 & \sigma^3_3\end{bmatrix}_{y_n}\right) \tag{22}$$

For plane stress, the constitutive relation is given by

$$\begin{bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\end{bmatrix} = \frac{E}{1-v^2}\begin{bmatrix}1 & v & 0\\ v & 1 & 0\\ 0 & 0 & (1-v)/2\end{bmatrix}\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} \tag{23}$$

The microscale displacements $\chi^j_i(y)$ (technically interpreted as the components of the displacement field arising from applied unit strain j) are necessary to compute the effective material properties. However, they also provide information about the stress state of the microstructure itself. It is natural to wonder how the stress state of the microstructure differs from the macroscopic stress response of the RVE. If we are interested in damage or fracture, probably we care primarily about extreme stress states within the microstructure. To this end, we can define a "microscale stress concentration" with

$$\gamma\Big(E(y), \boldsymbol{\epsilon}\Big) := \frac{\max\Big(\sigma_v\Big(\sigma^j_i(y)\epsilon_j\Big)\Big)}{\sigma_v\big(D^H_{ij}\epsilon_j\big)} \tag{24}$$

Here, $\boldsymbol{\epsilon}$ is a generic applied macroscopic strain, and $\sigma_v(\cdot)$ corresponds to a function taking three stress components to the von Mises stress. We use linearity in the numerator to compute the generic stress state from the unit strain response, and take the maximum von Mises stress in the microstructure. As a reminder, $\sigma_i^j(y)$ is stress component i at position y in the RVE from applied strain j. This ratio gives a sense of how variations in the microstructure amplify stresses compared to the effective properties.

3 Data-driven Surrogate Modeling

When the material microstructure varies spatially and/or the material exhibits non-linear constitutive behavior, multiscale analyses of structures require a very large number of finite element solves. Consequently, we are interested in finding surrogate models for some or all of the microscale finite element problems. Example multiscale problems of interest include:

- A surrogate model to compute the homogenized tensor from the RVE microstructure (linear elasticity). This would be useful when the material varies in space or the microstructure is an optimization variable.
- A surrogate model to compute the homogenized tensor from the RVE microstructure and the applied macroscopic strains (non-linear elasticity). Even when the material is constant, this would be useful to replace microscale finite element solves at each iteration in Newton's method.
- A surrogate model to predict the stress or displacement field in the RVE as a function of the microstructure and the applied strains (linear or non-linear elasticity). This is only useful if the mechanics of the microstructure are of interest, otherwise homogenized tensor should be predicted directly. Something like "stress concentration" effects within RVE could be of interest if designing against damage/fracture.

3.1 General Comments on Implementation

One of the tasks of this semester was to become familiar with Python's powerful machine learning library "PyTorch." All neural networks were constructed in PyTorch to allow for experimentation and customization. Python is used to read in text file outputs from custom MATLAB microscale implementation or MORIS homogenization code. In each problem, hyperparameters can be tuned to optimize the performance of the network. These include activation function, network width/depth, and learning rate. Training can be run for as many epochs are required to obtain saturation/overfitting. The Adam optimizer and mean-squared error loss are used for all examples. Finally, two different network architectures are investigated–DeepOnet and Fully-Connected Network.

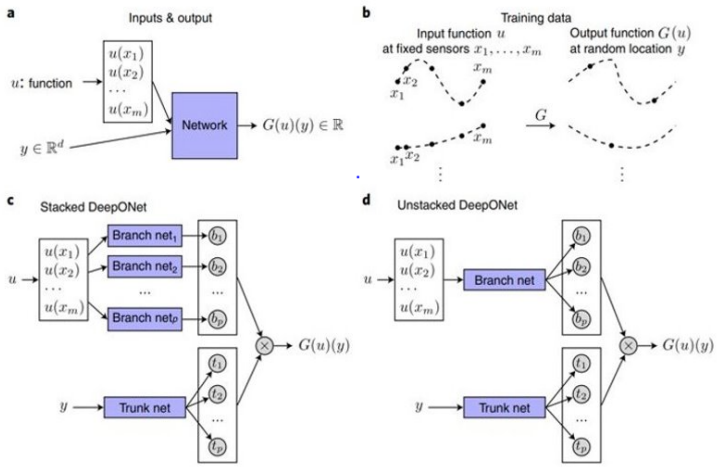

Figure 4: Schematic of DeepOnet from original Karniadakis paper: a) the network takes in an input function and a point in the output domain and models the action of operator G on the input function u at point y. b) the network is trained by sampling input functions at specified "sensor" points along with y points and the corresponding solution. Note that the same input function is used multiple times if different y points are sampled. c) & d) stacked and unstacked DeepOnet architecture. The unstacked network is a more frugal approach.

3.2 Deep Operator Network (DeepOnet)

DeepOnet originates in a 2020 Lu & Karniadakis paper titled "Learning nonlinear operators for identifying differential equations based on universal approximation theorem of operators." An operator is a map from a space of functions to another space of functions. A simple example operator is an antiderative:

$$G(u)(y) = \int_0^y u(x)dx$$

The operator G takes in a function u and outputs the function $G(u)(y)$. DeepOnet is a neural network architecture designed to match the structure of this problem: it takes in a function and a point in the output domain, and predicts the value of the operator acting on the input function at that point. See Figure 4 for a schematic of DeepOnet, taken from the original paper. The operator could be of a more complicated form, such as

$$a(x)\frac{\partial^2 f}{\partial x^2} + b(x)\frac{\partial f}{\partial x} + c(x)f(x) = u(x)$$

Conceptually, there is an operator which takes the input function $u(x)$ to the solution of the ODE $f(x)$. This might be written as $f(x) = G(u)(x)$. DeepOnet

has a "branch" and "trunk" network which treat the input function and point in output domain separately. The input function $u(x)$ passes through the branch network and point y through the trunk, then the two recombined with a dot product. Conceptually, this could be thought of as the network learning shape functions in the y domain within the trunk net which are independent of the applied forcing. Then, the branch network learns coefficients on these shape functions which depend on the input function. The solution is approximated as a linear combination of shape functions and weights.

A training data point in DeepOnet consists of samples of a particular input function $u^i(x)$, a point in the output domain y^j(potentially vector-valued), and a target $G(u^i)(y^j)$ which is the action of the operator we are trying to learn on $u(x)$ at point y. This has the following form:

$$\left[u^i(x_1), \ldots, u^i(x_n), y_1^j, \ldots, y_d^j, G(u^i)(y^j)\right] \tag{25}$$

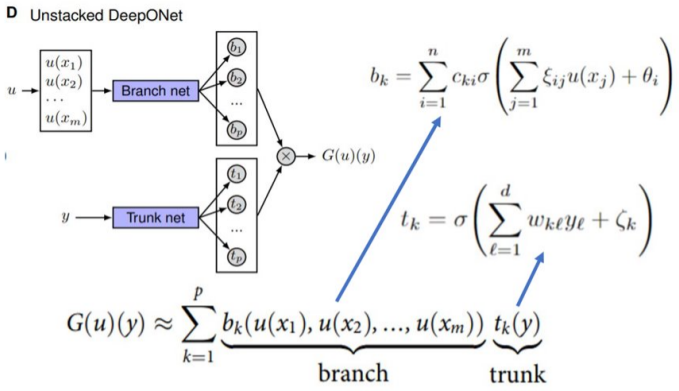

Figure 5: The unstacked DeepOnet is used throughout this report. The input function is passed through a single hidden-layer fully connected network in the branch net to obtain the weights b_k. The trunk net encodes the point y with a single layer network. The weights and biases in these transformations are parameters which are fit during training. Note that $\sigma(\cdot)$ represents an activation function.

The same function $u^i(x)$ is re-used for many samples in the output domain y^j. The parameters of the network are updated in training to minimize the mean-squared error between the true solution $G(u^i)(y^j)$ and the network prediction $\tilde{G}(u^i)(y^j)$. Turning now to the use of DeepOnet in the context of multiscale solid mechanics, we wish to replace the microscale finite element problems with surrogate models. Given the structure of DeepOnet, it is necessary that the output lives in the space of functions, otherwise are not learning an operator. Thus, it is not feasible to predict the homogenized tensor with DeepOnet,

because this is a matrix with all spatial variation integrated away per Eq. 16. The problem of interest is then Eq. 19–DeepOnet can be used to eliminate the finite element problem on the RVE. We are probably interested in either the displacements $\chi_i^j(y)$ or the microscale stresses arising from these displacements. This problems fits the structure of DeepOnet much better, as a forcing function (applied strains) pass through a complex operator to produce an output field. If we look at displacements, conceptually we have $\chi_i(y) = G_i(\boldsymbol{\epsilon})(y)$ where the operator is now vector-valued. DeepOnet can be modified to account for this by adding a separate branch network for each displacement component. Analogous to finite elements where the same shape functions are given different degrees of freedom for components of the displacement at a given node, the trunk network is not modified to account for vector-valued outputs–the different branch networks combine via a dot product with a trunk network that is only dependent on the position y.

There are some caveats to the use of DeepOnet in this particular setting. To see this, let's replace the unit strains in Eq. 19 with an arbitrary applied strain vector:

$$\int_{\Omega^y} E(y) \begin{bmatrix} \delta u_{1,1} \\ \delta u_{2,2} \\ \delta u_{1,2} + \delta u_{2,1} \end{bmatrix} \cdot \boldsymbol{D}^{unit} \begin{bmatrix} \chi_{1,1} \\ \chi_{2,2} \\ \chi_{1,2} + \chi_{2,1} \end{bmatrix} d\Omega$$
$$= - \int_{\Omega^y} E(y) \begin{bmatrix} \delta u_{1,1} \\ \delta u_{2,2} \\ \delta u_{1,2} + \delta u_{2,1} \end{bmatrix} \cdot \boldsymbol{D}^{unit} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} d\Omega \quad (26)$$

If we want to find the displacement field $\chi_i(y)$ for applied strain $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^T$, the unit strain response along with linearity can be used to argue

$$\chi_i(y) = \epsilon_1 \chi_i^1(y) + \epsilon_2 \chi_i^2(y) + \epsilon_3 \chi_i^3(y) \quad (27)$$

In other words, linearity allows us to compute the displacement field of any applied strain once the unit responses are known. This result calls into question the utility of a surrogate model which maps $[\boldsymbol{\epsilon}, y] \rightarrow \chi(y)$. Furthermore, the applied strain is not a function in the sense of the earlier DeepOnet input functions $u(x)$ as it is simply a constant vector. In the homogenization framework outlined here, the applied strain forcing is constant over the entire RVE, thus there is no continuous function to sample at fixed sensor points as DeepOnet suggests.

Though the applied strain is constant in space, the essence of the problem still reflects the structure of DeepOnet–an input forcing is acted on a by an operator to produce a function-valued output. It is as if we want to learn the displacement field in a beam bending problem as a function of the magnitude of a constant distributed force. But in the case of the RVE (as well as a linear beam

bending problem), linearity makes the dependence on that variable magnitude trivial...strictly speaking, what would make sense is to learn the RVE's three unit responses for a variety of materials with some neural network. Through linearity, this would allow efficient computation of the the homogenized tensor and the mechanics of the microstructure for any applied strains and material distributions within the training set. In the framework of DeepOnet, however, this suggests one network for each unit strain response where the input is the microstructure via $E(y)$. See Figure 6 for a schematic. The problem with this approach is that the Young's Modulus $E(y)$ is not a function acted on by an operator as it is a parameter of the operator itself! In the case of linear elasticity, *some* neural network architecture similar to this seems to be the most efficient and logical approach–because for a given material, all microstructural mechanics can be constructed from three unit responses, we should target the influence of the material on the unit response and rather than focus on the applied strains.

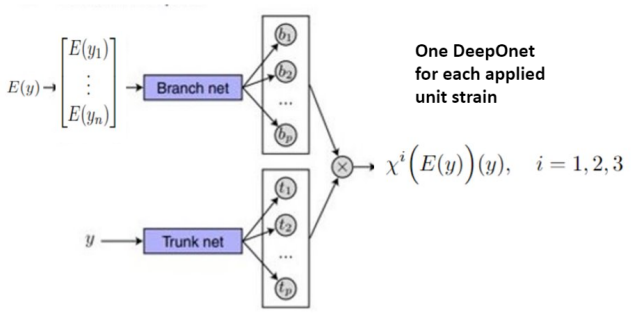

Figure 6: The ideal surrogate model for the case of linear elasticity–three networks are trained to predict the three unit responses of the RVE for a variety of materials. The material could be parameterized to avoid a large vector input to the branch net. Once the networks are trained, three finite element solves for the homogenized tensor and the mechanics of the microstructure are replaced by three forward passes through the networks. Perhaps a different network architecture is better suited for this problem?

All of this being said, DeepOnet does make sense in the world of non-linear elasticity. There is no decomposition of the mechanics of the RVE into unit responses, thus there is no simpler representation of the displacement field than the arbitrary expression $\chi(\boldsymbol{\epsilon}, E(y), y)$. Though the applied strain is still a constant vector, the difference between samples of a function and a vector seems immaterial. In summary, we have the following takeaways:

- DeepOnet is useful in situations where the displacement/stress response of the microstructure are of interest. It should not be used to predict the homogenized tensor because the output is not a function

- DeepOnet does not take advantage of linearity in linear elasticity, thus its use/implementation is more of a proof of concept. It does make sense in non-linear settings.
- A surrogate model for the problem outlined in Figure 6 would be speed up multiscale computations in linear problems. This could be a different network architecture.

3.3 Fully-Connected Network (FNN)

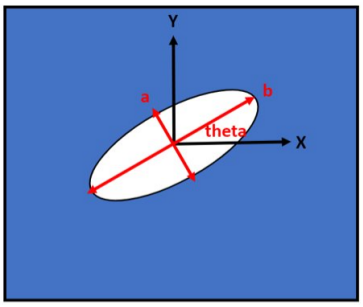

Figure 7: An example of how the Young's Modulus field $E(y)$ within the RVE can be parameterized. An elliptical hole with dimensions/orientation controlled by three parameters is punched in the center of the domain. The Young's Modulus takes on a constant value E_0 outside the ellipse, and an arbitrary small value approximating void inside.

A fully-connected network is probably the simplest neural network architecture. It represents a very general map between input and output vectors. The FNN is a logical choice of architecture to learn the homogenized tensor as a function of the material (spatially varying Young's Modulus) in the RVE. Essentially, we want to learn the relationship outlined by Eq. 16. Conceptually, we have

$$\boldsymbol{D}^H = \boldsymbol{D}^H\left(\sigma\Big(\chi\big(E(y)\big)\Big)\right)$$

The homogenized tensor is a function of the stresses, which are a function of displacements, which are a function of the material through $E(y)$. In a finite element setting, the modulus is stored at a discrete set of points, so the FNN should learn the relation

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} - \text{hidden layers} \rightarrow \begin{bmatrix} C_1^H \\ C_2^H \\ \vdots \\ C_6^H \end{bmatrix}$$

But in a fine mesh, the input vector will be intractably large. Thus, per Figure 7, we will restrict our attention to materials which have lower-dimensional representations. The modulus field $E(y)$ is a constructed from a set of parameters θ so that $E(y) = E(y;\theta)$. Instead of the modulus at every point in the finite element mesh, the input to the FNN can be the parameters governing the distribution of material. Thus, we learn the map between material parameters and effective material properties of the RVE:

$$\begin{bmatrix}\theta_1\\ \vdots \\ \theta_p\end{bmatrix} - \text{hidden layers} \rightarrow \begin{bmatrix}C_1^H\\ C_2^H\\ \vdots \\ C_6^H\end{bmatrix} \tag{28}$$

This is represented schematically in Figure 8. Note that this relation is valid because the homogenized tensor does not depend on the magnitude of the applied strain. This is an artefact of the linear constitutive relation. In non-linear problems, we could still use an FNN to learn the independent components of the constitutive tensor; however, they will depend on the strains applied to the RVE. In a non-linear setting, the map to learn is

$$\begin{bmatrix}\theta_1\\ \vdots \\ \theta_p\\ \epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} - \text{hidden layers} \rightarrow \begin{bmatrix}C_1^H\\ C_2^H\\ \vdots \\ C_6^H\end{bmatrix} \tag{29}$$

Having a surrogate model of this sort would greatly expedite Newton iterations in a non-linear solver. Instead of solving a non-linear problem on RVE's at every integration point in the finite element mesh to obtain effective material properties, only a forward pass through the surrogate model would be required.

4 Results

4.1 DeepOnet–Stress

Given that stress is most likely to be the quantity of interest in a multiscale analysis (whether to predict damage or effective material properties), it is logical to use DeepOnet as a surrogate model for stresses as a function of material governing the microstructure and the applied strains. See Figure 9 for a schematic of the Young's Modulus field $E(x, y)$. Inside the unit domain $\Omega^y = [0, 1] \times [0, 1]$, the microstructure is parameterized with

$$E(x, y) = E_0\Big(1 + a\sin(2\pi x)\sin(2\pi y)\Big) \tag{30}$$

where a is a parameter sampled from uniformly from the interval $[-0.5, 0.5]$. DeepOnet takes in the parameter a along with three applied strain components

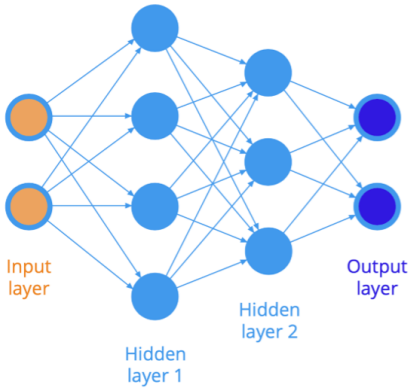

Figure 8: An example fully connected neural network. In the multiscale problem, the microstructure of the RVE is governing by the spatially varying Young's Modulus $E(y;\theta)$ where θ is a set of parameters. The parameters θ are passed into the input layer, and the network is trained to predict the components of the homogenized tensor in the output layer.

and a position in space and outputs a prediction of the 2D stress tensor. Applied strain components are sampled uniformly and independently from the interval $[-1, 1]$. One problem with using stress as the quantity of interest in DeepOnet is that stress fields are not necessarily continuous. This violates assumptions behind the universal approximation theorem of operators which motivated the development of the DeepOnet framework in the first place. This is something to keep in mind, despite the fact that the particular parametric microstructures here have smoothly varying material properties thus ensuring continuity of the stress. Note the material parameter a enters into both the branch and trunk network. See Table 4.1 for a list of parameters governing the construction and training of the DeepOnet to predicted stresses in the RVE. Note that one "sample" in the training data is the stress at the center of every element in a finite element simulation (ie one combination of applied strain and material for all spatial positions within RVE).

Training and test performance are shown in Figures 10-11. Though not shown here, the sigmoid activation function performs very poorly compared to Leaky ReLu in this problem. The basic architecture of DeepOnet can be "deepened" by adding additional hidden layers in the branch and/or trunk networks. We call this an "augmented" DeepOnet structure.

4.2 DeepOnet–Displacement

The same parametric microstructure as above is used to predict displacements within the RVE as a function of the applied strains, material, and spatial po-

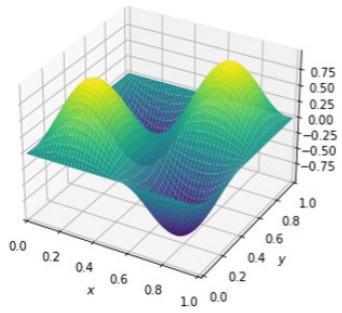

Figure 9: The form of the perturbation to a nominal Young's Modulus value within the RVE used in all DeepOnet surrogate models.

Hyperparameter	Value
Learning Rate	0.05
Nominal Young's Modulus (E_0)	1E7
Poisson Ratio (v)	0.3
Training Batch Size	10000
Samples in Training Data	10000
Samples in Test Data	500

Table 1: The finite element mesh which generated the data for the stress-based DeepOnet consists of 25×25 nodes. Other hyperparameters are varied to fine tune performance of the network.

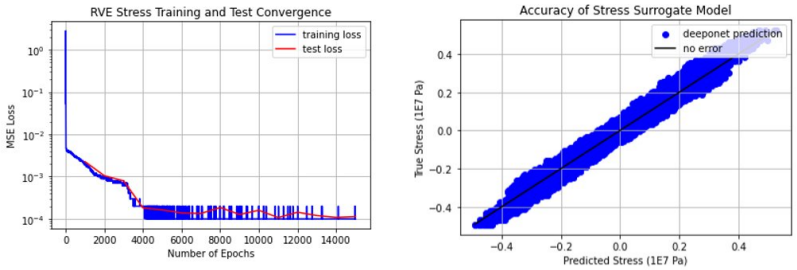

Figure 10: Training and validation for traditional DeepOnet structure with hidden layer width $n = 35$ and dot product size $p = 35$, and LeakyReLu activation function.

sition. The displacement field is continuous, so the assumptions of the universal operator approximation theorem are respected. Whereas above, stresses at element centers were predicted, we now train the network on nodal displace-

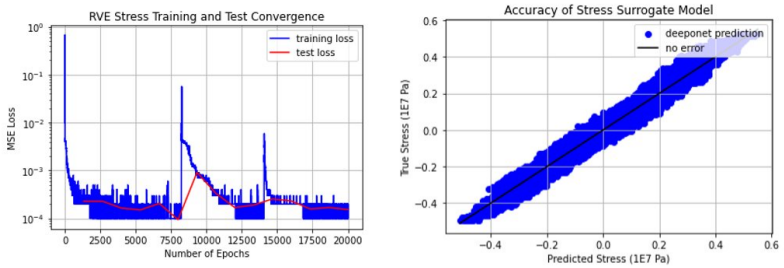

Figure 11: Training and validation for augmented DeepOnet structure where a second hidden layer is added to the branch network and a hidden layer added to the trunk network. Despite the vast increase in size of this network, the validation accuracy does not improve significantly.

Hyperparameter	Value
Learning Rate	0.1
Nominal Young's Modulus (E_0)	1E7
Poisson Ratio (v)	0.3
Training Batch Size	10000
Samples in Training Data	10000
Samples in Test Data	500

Table 2: The finite element mesh which generated the data for the displacement-based DeepOnet consists of 25×25 nodes. Other hyperparameters are varied to fine tune performance of the network.

ments. See Table 4.2 for a list of fixed hyperparameters. It was observed that a larger learning rate was advantageous for training the DeepOnet on displacement data, and that the sigmoid activation function performed much better than LeakyReLu (opposite of above). See Figures 12-14 for results of training and validation of the DeepOnet focused on predicting displacements. In a nutshell, these models perform much better than the stress predictions and have the benefit of generalizing to more complex microstructures with stress discontinuities.

4.3 Deeponet–Stresses and Homogenized Tensor From Displacement

Now that we have models trained to predict displacements, we can use PyTorch's gradient computations to obtain stresses from the displacement fields fit by the network. We take gradients with respect to inputs into the trunk network (two spatial coordinates), use the strain-displacement relation to obtain the strain vector, then the constitutive model for stresses. MATLAB stress

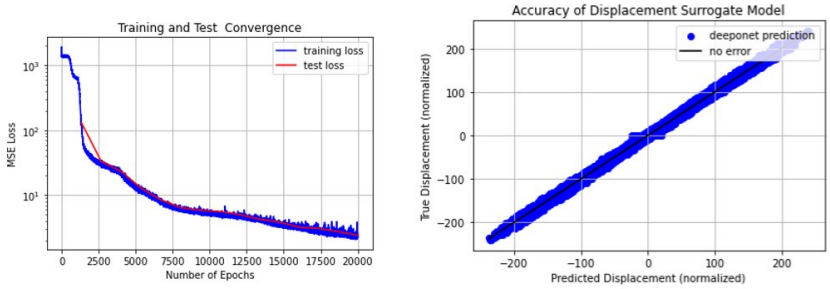

Figure 12: Traditional DeepOnet structure tested on 500 unseen microstructural displacement fields with branch hidden layer width $n = 30$ and dot product size $p = 30$. The displacements are multiplied by $\sqrt{E_0}$ for normalization.

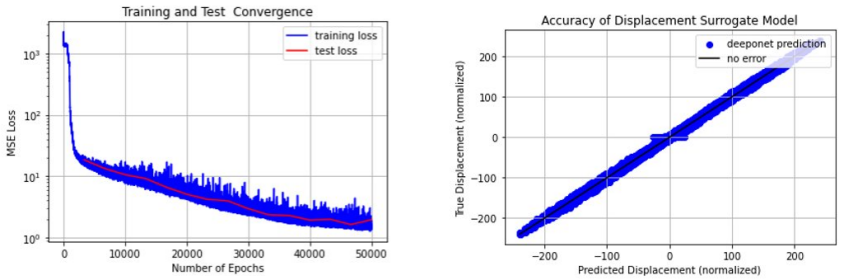

Figure 13: Traditional DeepOnet structure tested on 500 unseen microstructural displacement fields with branch hidden layer width $n = 35$ and dot product size $p = 35$. The displacements are multiplied by $\sqrt{E_0}$ for normalization. The average magnitude of the error between stress components is 3%.

data is generated, whereby the spatial position, applied strain, and material parameter are stored along with corresponding stress tensor values. This is an interesting test of the network in two ways–first, the network is trained on nodal displacements, whereas stresses are computed at element centers. Second, and probably more important, is that we have not enforced any constraints on gradients of the network! We are, in a sense, hoping that the interpolation of the training data is sufficiently smooth to produce reasonable gradients. Thus, we predict a displacement for a given set of inputs, then compute the stress corresponding stress through the neural network, and compare against the MATLAB solution. Once again, sigmoid activation shows the best performance so results from other activation functions will not be shown. See Figure 15 for validating the accuracy of the stress predictions for two different models. As outlined in the figure, the gradient operation only incurs a slight increase in the percent error of the predictions compared to displacements. A simple linear regression

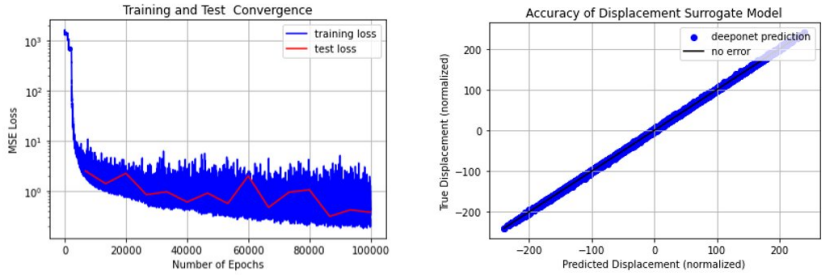

Figure 14: Augmented DeepOnet structure tested on 500 unseen microstructural displacement fields. A second hidden layer is added to the branch network and a hidden layer added to the trunk. All hidden layers widths are $n = 25$, while the size of the dot product combining the branch and trunk is $p = 35$. The displacements are multiplied by $\sqrt{E_0}$ for normalization. The average magnitude of the error between stress components is 1%.

analysis is used to determine whether the applied strain components, material parameter, or spatial position can be used to predict the error in the stresses. In addition to these predictors, a quadratic basis function $(x(1-x))$ is added to gauge whether error is concentrated in the center of the RVE, which would not be picked up by a traditional linear regression. It was found that only a small percent of the variation in the stress error is explained by any of these predictor variables, thus the error appears more like noise than a systematic trend.

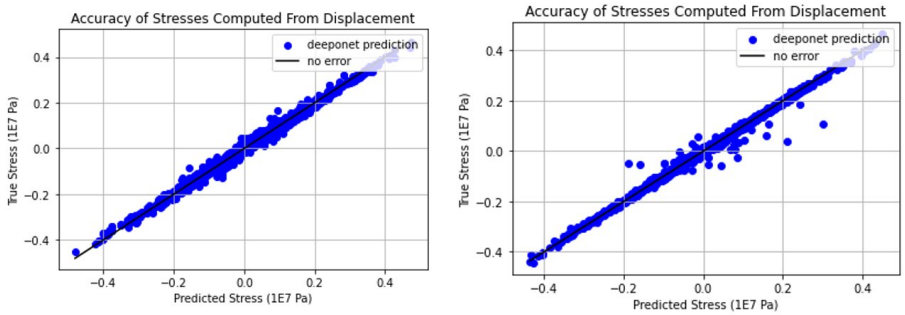

Figure 15: The model on the left is traditional DeepOnet structure with $n = p = 35$ and on the right, the augmented DeepOnet structure with additional hidden layers. Both models are tested against 5000 input/stress tensor pairs from MATLAB. The traditional DeepOnet went from 3% error in the displacements to 8% error in stresses, whereas the augmented DeepOnet went from 1% error in displacements to 4% in stresses.

As a final test of the DeepOnet trained on displacements, we can compute the homogenized tensor from stress predictions generated by computing displacement gradients. This is similar to testing the accuracy of the stress predictions, except that the homogenized tensor depends only on the accuracy of stress predictions from the three unit strains. If the model does not accurately predict the stress fields from the unit strains specifically, it will not produce accurate effective material properties of the RVE (regardless of how accurate stress predictions from other applied strains may be). Using the definition of the homogenized tensor from Eq. **??**, we loop through the positions of element centers, compute the stresses from the three unit strains at this position with DeepOnet (going first through displacements and using a given material parameter a), and add them to the microstructural constitutive matrix. A single homogenized tensor is produced for each material parameter a, and we can compare the eigenvalues of the tensor predicted by DeepOnet against a true value from MATLAB. See Figure 16 for the results of this analysis.

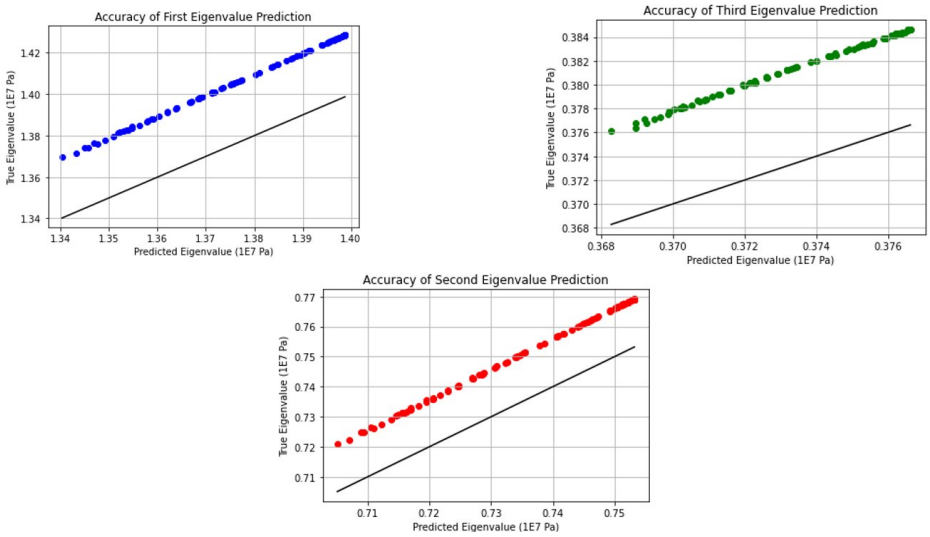

Figure 16: Black line indicates no error in eigenvalues, whereas scatter plots are results of DeepOnet homogenization compared against true values. The trend is matched very well but with a constant shift–it seems that DeepOnet consistently overpredicts eigenvalues of the homogenized tensor. The error might arise from the fact that the unit strains are "extreme" inputs given that components of applied strains are sampled from $[-1, 1]$ to train the network. Despite the shift, the magnitude of the error is still quite small.

Hyperparameter	Value
Learning Rate	0.01
Activation Function	Sigmoid
Width of Layer 1	30
Width of Layer 2	30
Samples in Training Data	10000
Samples in Test Data	500

Table 3: Parameters governing the construction, training, and testing of two-layer fully-connected network. The finite element mesh which generated the data consisted of 25×25 nodes.

4.4 FNN–Homogenized Tensor

The accuracy of the predictions from the surrogate model are assessed by comparing the three eigenvalues of the true and predicted homogenized tensors. See Figure 17 for the results of the FNN's performance and the above table showing problem parameters. The training is done in batches of 3000 data points. No over-fitting is observed in the data, but the error saturates after about 15000 epochs. In a basis of eigenvectors, the homogenized tensor is diagonalized with the three eigenvalues, thus they characterize the behavior of the transformation and are a reasonable choice for comparing two tensors. The surrogate model performs very well except for a handful of outliers, which are especially prominent in the first and second eigenvalues (ranked greatest to least). The FNN replaces three finite element solves for the unit strains with a single forward pass. If the few outliers are dealt with in some way, this seems to be a promising approach to expedite multiscale linear elastic analysis and optimization problems.

5 Code Roadmap

This is a list of MATLAB and Python codes used in this semester of research for future reference.

- "deeponet_stress_parameterized_material.m" – MATLAB driver script for generating RVE stress data for parametric microstructure
- "deeponet_displacement_parameterized_material.m" – MATLAB driver script for generating RVE displacement data for parametric microstructure (also creates stress and homogenized tensor to compare DeepOnet predictions against)
- "FNN_homogenized_tensor_wild_ellipse.m" – MATLAB driver script for generating homogenized tensor for RVE with elliptical hole controlled by three parameters

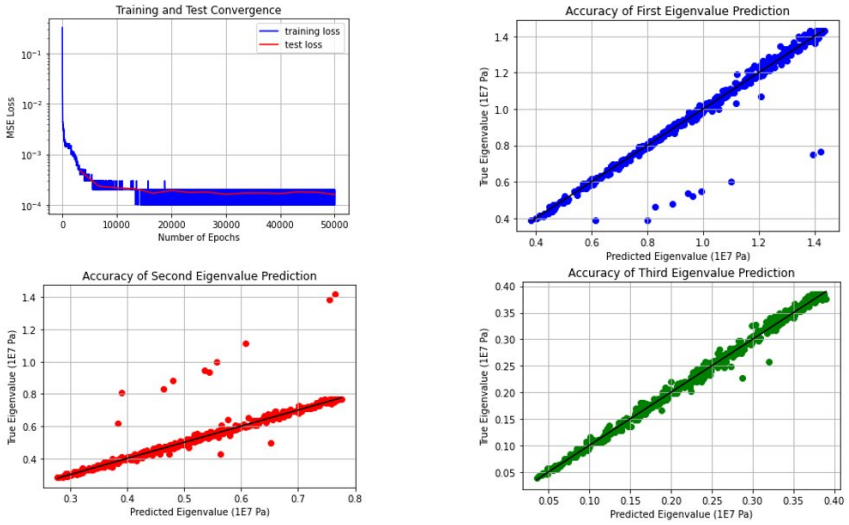

Figure 17: Results of using fully-connected network to predict homogenized tensor for RVE material controlled by three parameters. The predicted eigenvalues match the true eigenvalues very well apart from some outliers. The black line indicates perfect agreement between the two models.

- "homogenized_tensor.m" – MATLAB script that calls other functions to compute homogenized tensor for given Young's Modulus field
- "RVE_displacement.m" – computes stress and displacement in RVE for given mesh size, applied strain, and Young's Modulus field
- "calfem_data_structures.m" – Called in beginning of driver scripts to produce necessary data structures for calfem finite element analysis on tensor mesh with equal elements
- "DEEPONET_RVE_STRESS.py" – Python driver script reading in data from text files, building neural network, training, and validating for stress predictions
- "DEEPONET_RVE_DISPLACEMENT.py" – Python driver script building neural network to predict displacement, computing stresses from these displacements, and building homogenized tensor from stresses (longest script of three python files)
- "homogenized_tensor_FNN_wild_ellipse.py" – Python driver script building fully connected network to predict six components of homogenized tensor from three parameters governing RVE microstructure

Appendices

A Stress Equilibrium Convolution Filter

In two dimensions, stress equilibrium in the absence of body forces is

$$\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{12}}{\partial x_2} = 0$$
$$\frac{\partial \sigma_{12}}{\partial x_1} + \frac{\partial \sigma_{22}}{\partial x_2} = 0$$

Imagine we want to approximate stress equilibrium at a point P inside a square mesh where the stress tensor is known at surrounding points. Call the element below P south (S), to the right east (E), above north (N), and to the left west (W). Using central differencing and Voigt notation for 2D stresses, this relation is approximated as

$$\begin{bmatrix} R_1 \\ R_2 \end{bmatrix} = \frac{1}{2\Delta x} \begin{bmatrix} (\sigma_1^E - \sigma_1^W) + (\sigma_3^N - \sigma_3^S) \\ (\sigma_3^E - \sigma_3^W) + (\sigma_2^N - \sigma_2^S) \end{bmatrix}$$

where R_1 and R_2 represent residuals which will be small if equilibrium is approximately satisfied. When the stress tensor is known at the four surrounding elements, this can be written in matrix form as

$$R_i = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} = F_{ij} s_j = \frac{1}{2\Delta x} \begin{bmatrix} 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \sigma_1^S \\ \sigma_2^S \\ \sigma_3^S \\ \sigma_1^E \\ \sigma_2^E \\ \sigma_3^E \\ \sigma_1^N \\ \sigma_2^N \\ \sigma_3^N \\ \sigma_1^W \\ \sigma_2^W \\ \sigma_3^W \end{bmatrix}$$

Note that the stress tensor will be a prediction from the neural network. If the network is trained to predict displacements, a scheme of this sort could be used to apply some constraints on the displacement gradients. The benefit of this approach is that it has a similar structure to convolution filters, which could simplify implementation. However, if constraints are put on displacement gradients, we might as well train on stress and displacement data. *It is important to note that in the microscale finite element problem, there are non-zero body forces in the form of volumetric strains.* Thus we expect non-zero residuals in the stress equilibrium shown above.

B Material Reconstruction

Two important facts about engineering materials are known from experimental evidence: empirical constitutive laws for a given material differ between specimens under identical test conditions, and the microstructure varies spatially within/between specimens in an apparently random manner. As we know from homogenization, the microstructure influences the effective material properties and thus the first point may be explained by the second: random spatial variations in the microstructure introduce uncertainty into the effective constitutive properties of a material. Thus, the field of Material Characterization & Reconstruction (MCR) is required in order to, well, characterize the variations of the microstrucure and reconstruct novel but statistically equivalent macroscopic structures from a limited number of experimental samples of the microstructure. Defining "statistically equivalent" is the essence of the field.

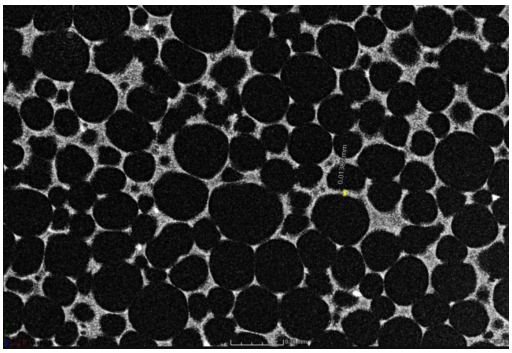

Figure 18: Microstructural CT scan. This could be a porous material or two-phase composite. Statistical properties of the two phases can be estimated from images of this sort and then used to reconstruct novel materials by sampling.

A common approach is to treat the microstructure as an image, and ensure that statistical properties of the different phases (colors) within image are preserved for a reconstructed material. A popular technique for two-phase composite materials is the two-point correlation, which measures spatial autocorrelations of the microstructure in different directions. For a given vector r, the two-point f_r^{ij} correlation measures the probability of finding phase j at position $x+r$ when the phase is i and position x. Though this technique has had success in reconstructing representative microstructures, it offers no control over the effective material properties of the RVE.

Generative models such as Variational autoenconders (VAE) and generative adversarial networks (GAN) have been widely used content which is novel but statistically equivalent to a training data set. These models can be trained on images of a certain object, and then sampled to produce a novel image of that

object. They use the language of statistics and neural networks to fit complex distributions to vectorized image data, and require that samples from the fit distributions correspond to images which are similar to the training set. Neural networks are convenient here because additional terms are simple to add to the loss function which trains the network–in addition to training the model to produce microstructures (in the form of images) which visually and statistically resemble training data, we could compute homogenized properties of the generated images and penalize microstructures which differ significantly from the effect properties of training data. In fact, we might be able to fit the distribution of experimental constitutive laws so that the variation in reconstructed microstructures corresponds to that of experiments!

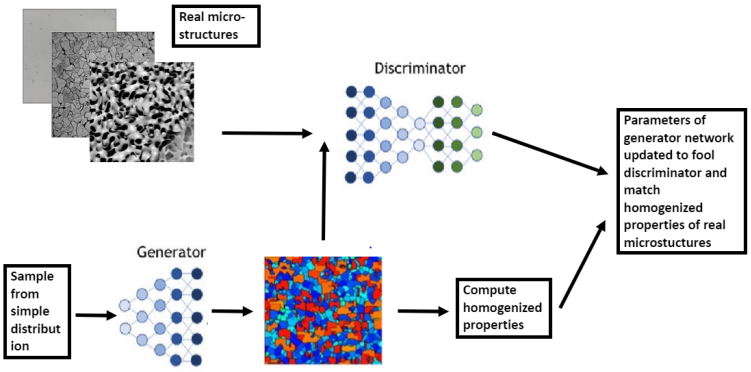

Figure 19: Generative Adversarial Network framework for material reconstruction problem. Inputs to the generator network are sampled from simple distributions (noise), and the network is trained to produce images of microstructures which maximize false classifications from the discriminator. This ensures that the discriminator, which is trained alongside the generator, cannot tell the difference between real microstructural images and the generated ones. Simultaneously, outputs from the generator are homogenized to obtain effective material properties, and microstructures which deviate from the empirical distribution of homogenized properties are penalized.