

Recurrent localization networks applied to the Lippmann-Schwinger equation

Conlain Kelly ^a, Surya R. Kalidindi ^{a,*}

^a Georgia Institute of Technology, Atlanta, Georgia 30332, USA



ARTICLE INFO

Keywords:

Machine learning
Learned optimization
Localization
Convolutional neural networks

ABSTRACT

The bulk of computational approaches for modeling physical systems in materials science derive from either analytical (i.e., physics based) or data-driven (i.e., machine-learning based) origins. In order to combine the strengths of these two approaches, we advance a novel machine learning approach for solving equations of the generalized Lippmann-Schwinger (L-S) type. In this paradigm, a given problem is converted into an equivalent L-S equation and solved as an optimization problem, where the optimization procedure is calibrated to the problem at hand. As part of a learning-based loop unrolling, we use a recurrent convolutional neural network to iteratively solve the governing equations for a field of interest. This architecture leverages the generalizability and computational efficiency of machine learning approaches, but also permits a physics-based interpretation. We demonstrate our learning approach on the two-phase elastic localization problem, where it achieves excellent accuracy on the predictions of the local (i.e., voxel-level) elastic strains. Since numerous governing equations can be converted into an equivalent L-S form, the proposed architecture has potential applications across a range of multiscale materials phenomena.

1. Introduction

Most problems in materials science and engineering require the exploration of linkages between materials processing history, materials structure, and materials properties. Generally referred as Process-Structure-Property linkages [1], they constitute the core materials knowledge needed to drive materials innovation supporting advances in technology [2]. Traditional physics-based numerical methods have long been the standard for solving the governing field equations underpinning these linkages. For mechanical problems, these have included ubiquitous finite element methods [3–6] as well as FFT-based spectral methods [7–9]. However, standard solvers can constitute a major performance bottleneck in problems which require repeated solution over varied inputs, such as inverse problems [10–12] and multi-scale materials design [13–15].

As an alternative, machine learning (ML) provides tools to approximate unknown linkages in a parametrized fashion, with great success in many domains [16]. One of the most successful classes of ML models is neural networks [17], which have been applied with excellent results both in general applications [16,18–20], and within materials science [15,21–23]. Unfortunately, ML models tend to act as “black boxes”

whose inner workings do not permit the depth of analysis provided by purely physics-based models [24]. There is a clear demand for approaches that leverage the advantages of both methodologies in order to build reliable, scalable, and interpretable reduced-order models.

One example of such an effort is the Materials Knowledge Systems (MKS) framework [25,26]. Aimed at multiscale materials design [27,28], MKS formulates the governing field equations for heterogeneous materials in a Lippmann-Schwinger (L-S) form [7,29]. Using regression techniques to calibrate the first-order terms of a series expansion to the L-S equation, MKS presents a generalized approach for solving a broad class of scale-bridging problems in materials design and optimization [30,31]. However, improving the accuracy of these models requires higher-order L-S terms, which rapidly become more computationally expensive.

As an alternative, we propose an approach inspired by the intersections between iterative spectral methods [32] and recent advances in inverse imaging [19,33]; we cast the recurrent L-S equation as an optimization problem. Rather than employing a predefined optimization strategy, such as gradient descent or conjugate gradients [34], the optimizer is posed as a recurrent collection of convolutional neural networks. After being calibrated to available curated data (e.g., results

* Corresponding author.

E-mail addresses: ckelly84@gatech.edu (C. Kelly), surya.kalidindi@me.gatech.edu (S.R. Kalidindi).

of FEA simulations, phase field models), these networks act as proximal (or “update”) operators which take in a candidate solution and output an improved version. This iterative methodology emphasizes the underlying physics to permit greater model robustness and deeper analysis.

In this paper, we begin with a brief analysis of the L-S equation and its application to the linear elasticity problem, followed by discussion on the general L-S equation. Using this analysis, we then demonstrate how the L-S equation can be naturally posed as a machine learning problem, and how a neural network can learn proximal operations which minimize a physical quantity (e.g., stress field divergence) within a solution field. By exploring the interplay between the physical and computational interpretations of the L-S equation, we provide insight into a new class of ML models for materials science. We then analyze which aspects of our approach provide the greatest gains by exploring various model configurations, reinforcing the value of an iterative (rather than feed-forward) approach. Finally, we evaluate our methodology on the problem of elastic localization and compare it to a previous machine learning model.

2. Background

2.1. Linear elasticity and L-S

Originally developed in the context of quantum mechanics [29], the L-S equation – or class of equations – is an implicit integral form that can represent a fairly general space of physical phenomena. The L-S equation is especially useful in the context of physics of heterogeneous media with spatially-varying physical parameters: stiffness, conductivity, density, etc. We defer discussion of the general Lippmann-Schwinger form until Section 2.2.

As a case study, we consider the problem of computing the internal elastic strain field of a composite material undergoing bulk stresses or strains [7,32,35]. The composite microstructure is assumed to be composed of two or more distinct phases (i.e., thermodynamic material constituents), each exhibiting its own constitutive laws. This problem is herein referred to as elastic localization. An example two-phase microstructure and corresponding elastic strain field are presented in Fig. 1.

Physically, elastic localization is governed by a generalized Hooke’s law relating the variation in stress $\sigma(x)$, strain $\epsilon(x)$, and stiffness $C(x)$ over some volume V , along with the demand that the equilibrium stress field be divergence-free:

$$\sigma = C\epsilon, \quad (1)$$

$$\nabla \cdot \sigma = 0. \quad (2)$$

We consider constant periodic boundary conditions that correspond to the imposed volume-averaged strain, $\bar{\epsilon}$. With these choices, one can model the internal mechanical response of a representative volume

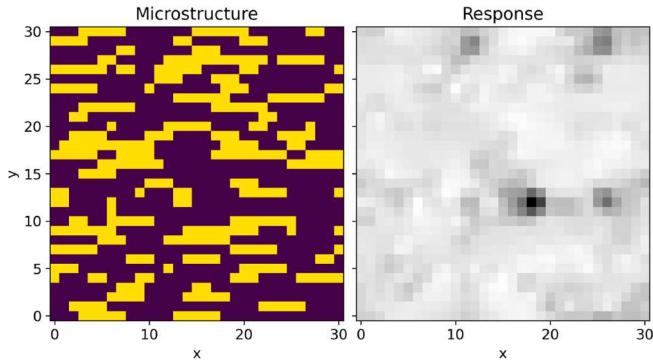


Fig. 1. Example microstructure-strain field pair for two-phase elastic localization. Yellow is high-stiffness phase; purple is low-stiffness phase; contrast between elastic moduli is $CR = 50$.

element (RVE) of the material. In these models, the RVE often serves as a statistical unit cell of a larger material structure. We note that the problem statement expressed here serves as a simple illustration of the L-S methodology, which has been successfully applied to more complex material systems and/or boundary conditions [32,30].

Following the work of Kröner [36], this system is converted into an equivalent form (the elastic L-S equation) by splitting the local elastic stiffness C into a selected (constant) reference value C^R and a perturbation value C' . Substituting these stiffness tensors into Eqs. (1) and (2) provides a partitioned version of the governing field equations:

$$\nabla \cdot (C^R \epsilon) = -\nabla \cdot (C' \epsilon) \quad (3)$$

Observe that the left-hand side of this equation is a linear operator, denoted as A , which acts on ϵ :

$$A\epsilon \equiv \nabla \cdot (C^R \epsilon) = C^R \nabla \cdot \epsilon \quad (4)$$

Clearly $A\epsilon$ is linear in ϵ ; therefore, it will have a corresponding Green’s function $G(x, s)$ [37]. Since divergence is uniform in space, we make the simplification $G(x, s) = G(x - s)$. This function represents the system’s response to an impulse inhomogeneity

$$AG(x - s) = \delta(x - s) \quad (5)$$

where δ denotes the Dirac-delta function.

We also partition the strain field as $\epsilon = \epsilon^R + \epsilon'$, where ϵ^R represents a (constant) reference strain determined by boundary conditions (this is also equal to the internal average strain $\bar{\epsilon}$), and ϵ' is the corresponding perturbation strain. Both C^R and ϵ^R are constant, so $\nabla \cdot (C^R \epsilon^R) = 0$ and thus ϵ^R is the homogeneous solution to A . For a given inhomogeneity $b(x)$, we can use the Green’s function to solve for the particular solution $\epsilon'(x)$:

$$A\epsilon(x) = b(x) \quad (6)$$

$$\Rightarrow \epsilon(x) = \epsilon^R + \int_V G(x - s) b(s) ds \quad (7)$$

Now we formally treat the right-hand side of Eq. (3) as the inhomogeneity b to obtain the elastic Lippmann-Schwinger Equation:

$$\epsilon'(x) = - \int_V G(x - s) (\nabla \cdot [C'(s) \epsilon(s)]) ds. \quad (8)$$

or

$$\epsilon(x) = \bar{\epsilon} - \int_V G(x - s) (\nabla \cdot [C'(s) \epsilon(s)]) ds \quad (9)$$

Since many governing equations can be converted into a similar structure, we refer to their transformed version as an “L-S form”; that is, Eq. (9) is the Lippmann-Schwinger equation corresponding to the elastic governing equations. A Lippmann-Schwinger derivation corresponding to a general governing equation is presented in Section 2.2. The reader is referred to Refs. [38,39] for more background on Green’s functions.

Using insights from previous work [22,27,35], we modify this equation to make it amenable to a learning paradigm. First, we integrate-by-parts to shift the derivative onto G and absorb it into a new operator, \hat{G} . Using $*$ to concisely denote a convolution, we obtain

$$\epsilon(x) = \bar{\epsilon} - \int_V \hat{G}(x - s) C'(s) \epsilon(s) ds = \bar{\epsilon} - \hat{G} * (C' \epsilon) \quad (10)$$

Next, we define a binary microstructure representation $m^h(x)$ that equals 1 if the material at point x is of phase (or material type) h , and 0 otherwise. Since each phase has its own stiffness, we can project the stiffness tensor C onto each phase: $C = \sum_h C^h m^h$ and likewise $C' = \sum_h C'^h m^h$. Finally, we combine the Green’s function terms with the C' expression to obtain yet another operator $\Gamma(x - s)^h \equiv \hat{G}(x - s) C^h$.

Applying all of these modifications, the elastic L-S form becomes

$$\boldsymbol{\epsilon}(\mathbf{x}) = \bar{\boldsymbol{\epsilon}} - \sum_h \boldsymbol{\Gamma}^h * (\mathbf{m}^h \boldsymbol{\epsilon}) \quad (11)$$

The problem of elastic localization has thus been reduced to a single convolutional integral containing the microstructure \mathbf{m}^h , candidate strain field $\boldsymbol{\epsilon}$, and a physics-determined stencil $\boldsymbol{\Gamma}^h$. Curiously, the first two terms appear solely as an element-wise product between \mathbf{m}^h and $\boldsymbol{\epsilon}$. This is due to the fact that the strain field is constrained indirectly by the divergence-free condition on $\boldsymbol{\sigma}$. One also observes that all effects of \mathbf{C} have been absorbed into $\boldsymbol{\Gamma}^h$. This corresponds to the fact that $\boldsymbol{\Gamma}^h$ is not unique: infinitely many choices of \mathbf{C}^R could result in this equation. Although mathematically equivalent to the original physics (and visually more complicated), Eq. (11) provides significant advantages for solution over large heterogeneous volumes. Several solution strategies have been explored to address elastic localization in heterogeneous material systems, resulting from different interpretations of the L-S equation.

Mathematically, $\boldsymbol{\Gamma}^h$ is a set of convolutional kernels – one for each phase – encoding the underlying physics of the problem. Given a strain field, it computes the corresponding stresses and peels off the strain perturbation field required to minimize the stress divergence. Using this perspective, many existing models view the L-S equation as a fixed-point equation and solve it via root-finding [7]. The rate of convergence of these methods tends to depend heavily on the variation in material properties and the choice of \mathbf{C}^R [32]. All of these physics-based approaches require a quantitative knowledge of the Green's function \mathbf{G} .

From a computer science perspective, the term $\mathbf{m}^h \boldsymbol{\epsilon}$ simply represents the strain field segmented by phase (since \mathbf{m}^h is a binary indicator function). Given a collection of true structure-strain pairs, one could either learn $\boldsymbol{\Gamma}^h$, or some approximation, to best conserve the equality. Following this view, several ML-based elastic localization models [22,35] have been applied to learn (non-iterative) linkages between \mathbf{m}^h and $\boldsymbol{\epsilon}$ by either approximating a series expansion of $\boldsymbol{\Gamma}^h$, or using a neural network to map \mathbf{m}^h to $\boldsymbol{\epsilon}$ directly, bypassing $\boldsymbol{\Gamma}^h$ completely. The disadvantage of these models is that they either truncate or ignore the underlying physics, trying to re-learn it from data. The tension between these two perspectives leaves room for a hybrid method which retains the iterative L-S structure, but uses ML to deduce the internal details of the $\boldsymbol{\Gamma}^h$ operator.

2.2. General L-S equation

This section presents a derivation for the general L-S equation, from a generic governing equation, drawing on the work of Moulinec [7] and Kröner [36]. This is provided for two reasons: to provide greater intuition and background for the L-S equation, and to motivate its compatibility with ML solvers. First, we write (in conservation form) a governing differential equation controlling a field \mathbf{y} which varies over space \mathbf{x} spanning some volume V :

$$\mathcal{H}(\mathbf{y}(\mathbf{x}); \mathbf{x}) = 0. \quad (12)$$

Observe that any governing equation can be partitioned into two coupled subequations, each governing their own subsystems. First define an operator \mathbf{A} which captures all linear (and spatially homogeneous) components of some general operator \mathcal{H} . Now define a second (possibly nonlinear) operator \mathbf{B} containing the rest of \mathcal{H} . One obtains the earlier example of elastic localization with the substitutions $\mathbf{y} \equiv \boldsymbol{\epsilon}$, $\mathbf{A}\boldsymbol{\epsilon} \equiv \nabla \cdot (\mathbf{C}^R \boldsymbol{\epsilon})$, and $\mathbf{B}(\boldsymbol{\epsilon}) \equiv \nabla \cdot (\mathbf{C}' \boldsymbol{\epsilon})$. Although not explicitly denoted, both \mathbf{A} and \mathbf{B} may contain implicit information about the solution domain's structure (terms such as \mathbf{C} or \mathbf{m}).

Using these operators we can rewrite the original equation as:

$$\mathcal{H}(\mathbf{y}; \mathbf{x}) = \mathbf{A}\mathbf{y} + \mathbf{B}(\mathbf{y}; \mathbf{x}) = 0 \quad (13a)$$

or

$$\mathbf{A}\mathbf{y} = -\mathbf{B}(\mathbf{y}; \mathbf{x}). \quad (13b)$$

This partitions the governing equation into two coupled systems: a linear homogeneous system permitting only “simple” solutions, and a nonlinear, heterogeneous system where the solution is more complicated. Before solving the complete equation, we consider the auxiliary system

$$\mathbf{A}\mathbf{y}(\mathbf{x}) = \mathbf{b}(\mathbf{x}) \quad (14)$$

for some inhomogeneity \mathbf{b} . We define the homogeneous solution to this system as \mathbf{y}^R , so that $\mathbf{A}\mathbf{y}^R = 0$. Note that in general, \mathbf{y}^R is determined by both \mathbf{A} and the relevant boundary conditions, and for some problems there may be more than one suitable \mathbf{y}^R . For problems with a constant solution field on the boundary, one finds that $\mathbf{y}^R = \bar{\mathbf{y}}$, i.e., the reference field is the average solution everywhere.

The choice of \mathbf{y}^R induces a corresponding perturbation (or “particular solution”) $\mathbf{y}' = \mathbf{y} - \mathbf{y}^R$. Because \mathbf{y}^R is annihilated by \mathbf{A} , note that $\mathbf{A}\mathbf{y}' = \mathbf{A}\mathbf{y}$. Since \mathbf{A} is linear, it will have a Green's function $\mathbf{G}(\mathbf{x}, \mathbf{s})$, which captures the system's impulse response [38]. Using this we write the particular solution to the auxiliary equation as a convolution between \mathbf{G} and \mathbf{b} and reconstruct the complete solution:

$$\mathbf{A}\mathbf{G}(\mathbf{x}, \mathbf{s}) = \delta(\mathbf{x} - \mathbf{s}) \Rightarrow \mathbf{y}'(\mathbf{x}) = \int_V \mathbf{G}(\mathbf{x}, \mathbf{s}) \mathbf{b}(\mathbf{s}) d\mathbf{s} \quad (15)$$

or

$$\mathbf{y}(\mathbf{x}) = \mathbf{y}^R + \int_V \mathbf{G}(\mathbf{x}, \mathbf{s}) \mathbf{b}(\mathbf{s}) d\mathbf{s}. \quad (16)$$

Now we return to Eq. (13b) and apply the auxiliary approach, this time treating the entire \mathbf{B} term as our homogeneity \mathbf{b} (and noting the attached minus sign). Plugging this into the perturbation expression for \mathbf{y} gives us:

$$\mathbf{y} = \mathbf{y}^R - \int_V \mathbf{G}(\mathbf{x}, \mathbf{s}) \mathbf{B}(\mathbf{y}(\mathbf{s}); \mathbf{s}) d\mathbf{s}. \quad (17)$$

This is the general Lippmann-Schwinger equation; since this derivation holds for any operator \mathcal{H} , we use the term “L-S form” for a given \mathcal{H} to describe the result of partitioning and substituting that \mathcal{H} into Eq. (17). Referring back to the example of elastic localization, Eq. (9) is the equivalent L-S form for Hooke's law (Eq. (2)).

Note that the linear system \mathbf{A} only enters the L-S equation through the definitions of \mathbf{y}^R and \mathbf{G} . For example, if one used the trivial choice of the identity for \mathbf{A} , the corresponding Green's function would just be the Dirac delta function, and Eq. (17) would simplify to the original governing equation.

For the L-S form to be advantageous over \mathcal{H} , \mathbf{A} must capture a non-trivial amount of the underlying equation. There are three primary factors which make the L-S equation useful. First, it is *partitioned*: the original system is broken into two coupled physical systems. This makes it similar to a traditional “splitting method” where the linear and nonlinear components are separated, allowing the solution of the nonlinear components to be informed by the homogeneous, linear solution. The coupling between systems means that the L-S equation is also *recursive*: the presence of \mathbf{y} in the inhomogeneity term leads to its appearance on both sides of the equation. If one desires to solve the problem analytically, the L-S form is likely no more useful than the original governing equation. However, the implicit structure is very suitable for iterative and optimization-based solvers [9,40]. Finally, the L-S equation is *convolutional*: the integral is actually a convolution between the \mathbf{B} term and a (possibly-unknown) Green's function \mathbf{G} . Roughly speaking, Eq. (17) presents the solution field $\mathbf{y}(\mathbf{x})$ as a balance between the global homogeneous “pull” (\mathbf{y}^R) and the localized “tug” ($\mathbf{y}'(\mathbf{x})$) of the solution values in a neighborhood near \mathbf{x} . In situations where \mathbf{B} is a

purely differential operator (such as elastic localization), and with appropriate boundary conditions, Eq. (17) can be integrated-by-parts to shift part of \mathbf{B} onto \mathbf{G} . This can simplify the integral term so that all of the physics is contained in a single convolutional stencil.

2.3. Neural networks background

As one of the most popular ML tools in use, neural networks are a class of parametrized function approximators that can be calibrated to curated data [17]. At a high level, an artificial neural network (ANN) operates as an alternating sequence of tunable linear transforms and nonlinear activation functions – mimicking the operation of physical neurons in the brain [41,42]. Under certain conditions, a sufficiently large neural network can be shown to act as a universal function approximator [43,44], motivating their use in a myriad of disciplines.

Two relevant specializations are the convolutional neural network (CNN), which uses convolution with a fixed-width stencil as its transform operation [45,18], and the recurrent neural network (RNN), which operates on sequential data and considers latent information carried across input iterations [46]. These tools can be combined to model the underlying structure of various problems. A well-designed ML model trained on sufficient data can be significantly faster than an analytical equivalent and still provide reasonable accuracy [47]. This comes at the expense of interpretability – they return a “black-box” model which is difficult to understand and analyze [24]. Additionally, the topology of these networks (e.g., number of layers, nodes per layer, activation functions) strongly determines their success [48], and the “best” configuration is problem-dependent and often constructed ad hoc.

Recently there has been tremendous interest in the application of neural networks to mathematical problems [16]. Specifically, variations of recurrent CNNs have been explored [49] to learn Bayesian priors for image denoising [33] or proximal operators for medical imaging [19]. These image analysis methods pose the problem such that the desired output is obtained via a learned optimization procedure, where the optimizer itself is formulated as a neural network. Surprisingly, these methods often employ very simple network designs, especially compared to deeper and more elaborate structures found in mainstream ML [20,50].

3. Methodology

3.1. L-S as learned optimization

We now explore how the perturbation expansion and L-S form allow a governing equation to be interpreted naturally as a machine learning problem. We first define a new operator Φ representing the entire right-hand side of Eq. (17). We also use \mathbf{m} to represent a problem domain’s underlying microstructure, which influences the inhomogeneity \mathbf{B} . Given a sample microstructure \mathbf{m}^* , we obtain the corresponding strain \mathbf{y}^* by minimizing the error (or loss) \mathcal{L} between \mathbf{y} and $\Phi(\mathbf{y}, \mathbf{m}^*)$ over all \mathbf{y} .

$$\Phi(\mathbf{y}, \mathbf{m}) \equiv \mathbf{y}^R - \int_V \mathbf{G}(\mathbf{x}, \mathbf{s}) \mathbf{B}(\mathbf{y}(\mathbf{s}); \mathbf{s}) \mathbf{ds} \quad (18)$$

$$\mathbf{y}^* = \Phi\left(\mathbf{y}^*; \mathbf{m}^*\right) = \underset{\mathbf{y}}{\operatorname{argmin}} \mathcal{L}(\mathbf{y}, \Phi(\mathbf{y}, \mathbf{m}^*)) \quad (19)$$

Although Φ may not be linear itself, linear analysis methods provide a useful interpretation: for a given microstructure \mathbf{m}^* , Φ has a (possibly non-unique) generalized eigenfunction \mathbf{y}^* with unit eigenvalue. Issues regarding the solution’s uniqueness and existence can arise from the original governing equation’s nonlinearity. In the case of the elasticity problem, the governing equation is linear, so a unique solution will exist.

Now the original problem of solving the governing equation \mathcal{W} has been reduced to that of minimizing the scalar loss \mathcal{L} given a particular microstructure via a learned optimization strategy. To do this, we define

a parametrized learner \mathcal{F}_θ that performs a sequence of proximal operations which progressively improve the solution field to match the governing physics. Here θ represents all possible parameters of this learner. The optimal set of parameters θ_{opt} is obtained by minimizing the expected error produced by \mathcal{F}_θ w.r.t. θ ; in the case of a CNN this represents the optimal network weights, and can be obtained via standard backpropagation [51]. Given a microstructure and initial guess \mathbf{y}_0 , we want \mathcal{F}_θ to provide a solution field which is approximately the true solution field \mathbf{y}^* :

$$\mathcal{F}_\theta(\mathbf{y}_0, \mathbf{m}^*) = \hat{\mathbf{y}} \approx \mathbf{y}^* \quad (20)$$

This is accomplished by a sequence of updates

$$\mathbf{y}_i = \mathbf{y}^R + f_i(\mathbf{y}_{i-1}, \mathbf{m}^h) \quad (21)$$

where f_i represents the perturbation proximal operator used at iteration i ; given a microstructure and candidate strain field, it outputs the perturbation component corresponding to an improved estimate of the true strain field. A pseudocode and visual representation of the approach developed in this work are presented in Fig. 2. Ideally, after the model is trained, \mathcal{F}_θ and Φ have the same eigenfunctions (i.e., $\mathcal{F}_\theta(\mathbf{y}^*, \mathbf{m}^*) = \mathbf{y}^*$).

The optimization strategy employed by the \mathcal{F}_θ model is directly determined by the choice of f_i . To explore what this might look like for the elastic problem, consider an L2 loss function and plug in the elastic LS formula (Eq. (11)):

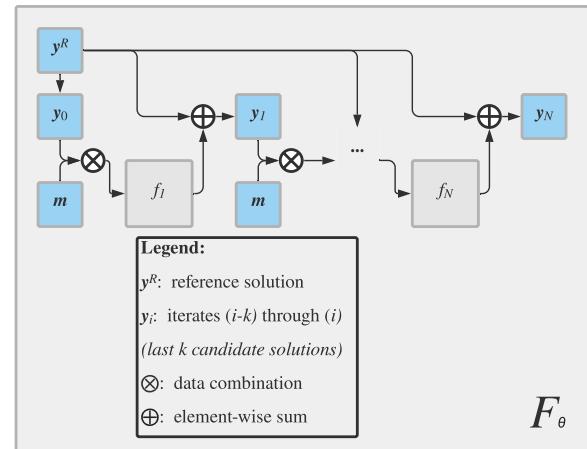
$$\Phi\left(\mathbf{y}; \mathbf{m}^h\right) = \mathbf{y}^R - \sum_h \mathbf{\Gamma}^{h*}(\mathbf{m}^h \mathbf{y}) \quad (22)$$

$$\mathcal{L}(\mathbf{y}, \Phi(\mathbf{y}; \mathbf{m}^h)) \equiv \frac{1}{2}(\mathbf{y} - \Phi(\mathbf{y}; \mathbf{m}^h))^2 = \frac{1}{2}\left(\mathbf{y} - \mathbf{y}^R + \sum_h \mathbf{\Gamma}^{h*}(\mathbf{m}^h \mathbf{y})\right)^2 \quad (23)$$

The original fixed-point approach of Moulinec [7] corresponds to the choice

```
def  $\mathcal{F}_\theta(\mathbf{y}^R, \mathbf{m})$ :
     $\mathbf{y}_0 = \mathbf{y}^R$ 
    for  $i = 1$  to  $N$ :
         $\mathbf{y}_i = \mathbf{y}^R + f_i(\mathbf{y}_{i-1}, \mathbf{m})$ 
    return  $\mathbf{y}_N$ 
```

(a) Pseudocode for optimization procedure



(b) Visualization of data flow through optimization procedure

Fig. 2. Pseudocode and visualization for optimization-based solution of the L-S equation.

$$f_i^M(y_{i-1}, \mathbf{m}^h) = - \sum_h \Gamma^{h*} \left(\mathbf{m}^h y_{i-1} \right) \quad \forall i \quad (24)$$

As an alternative, we can obtain a “steepest descent” formula by taking the gradient [34] of Eq. (23):

$$f_i^G(y_{i-1}, \mathbf{m}^h) = y_{i-1} - y^R - \gamma_i \frac{\partial}{\partial y_{i-1}} \mathcal{L} \left(y_{i-1}, \Phi(y_{i-1}; \mathbf{m}^h) \right) \quad (25)$$

or

$$\begin{aligned} f_i^G(y_{i-1}, \mathbf{m}^h) &= y_{i-1} - y^R - \gamma_i \left(\mathbf{I} + \frac{\partial}{\partial y_{i-1}} \sum_h \Gamma^{h*} (\mathbf{m}^h y_{i-1}) \right) \\ &\quad \left(y_{i-1} - y^R + \sum_h \Gamma^{h*} (\mathbf{m}^h y_{i-1}) \right) \end{aligned} \quad (26)$$

where γ_i denotes the step size at iteration i and \mathbf{I} represents the identity operator. By flattening everything into vectors and representing the convolution with an equivalent Toeplitz matrix [52], one can convert the product term in Eq. (26) into a single linear operator \mathbf{H} acting on y_{i-1} . The linearity of \mathbf{H} comes from the fact that the $\left(\mathbf{I} + \frac{\partial}{\partial y_{i-1}} \dots \right)$ term is actually independent of y_{i-1} . Using a constant λ_i to collect remaining terms of y^R , the steepest descent rule becomes:

$$f_i^G(y_{i-1}, \mathbf{m}^h) = (\mathbf{I} - \gamma_i) (y_{i-1} - y^R) + \gamma_i \mathbf{H} y_{i-1} + \lambda_i y^R \quad (27)$$

Effectively, the gradient descent rule says that the new perturbation field is obtained by correcting the previous perturbation field ($y_{i-1}' = y_{i-1} - y^R$) using a set of convolutional operations involving Γ^h and \mathbf{m}^h , then subtracting off a factor of y^R such that the output perturbation field is zero-mean.

A variety of more complicated update rules have been proposed to accelerate the solution of different forms of the L-S equation [32]. From the examples above one sees that any update rule will involve various terms of Γ^h , which itself contains both physical properties (such as C^h) as well as derivatives of the Green’s function (such as \hat{G}); more complicated update rules will simply require higher-order combinations. Therefore, if we hope to learn f_i , it must be parametrized in a way such that it can capture global convolutional stencils, as well as various derivatives thereof. Finally, we note that although most analytical approaches employ the same operator for each iteration, the \mathcal{F}_θ formulation permits varying f_i across iterations.

3.2. CNNs for Lippmann-Schwinger

Most analytical minimization procedures are (by design) problem-agnostic; this means that they can be expected to work reasonably well for many problems, but may not be optimal for the problem at hand. Rather than using a predefined optimization strategy, we formulate each f_i as a CNN that learns a proximal operator mapping a given microstructure and candidate solution field to an improved perturbation field. The central motivation behind using a CNN proximal operator is that given sufficient parameterization, it can emulate almost any optimization strategy; furthermore, that strategy will be customized to the problem at hand during training. Of course, this comes with the immense caveat that, absent any advances in theoretical ML, a learned optimizer will not have any provable convergence guarantees. This means that even as $N \rightarrow \infty$, our learned model may not converge to the true solution field for a given microstructure. In practice, however, the model can be tuned and trained until it consistently produces solutions within acceptable error tolerances.

We define the \mathcal{F}_θ optimization model imbued with a CNN f_i as a recurrent localization network (RLN), which performs the following operations during both training and evaluation phases: given a microstructure \mathbf{m} and initial guess y^R , estimate the true solution field by

refining it over N iterations. At iteration i , the microstructure is combined with candidate solutions from previous iterations and passed through CNN f_i . Note that f_i outputs a *perturbation* field y'_i ; the reference solution y^R is already known, so there is no need to learn that. In order to simulate a multi-step solver [3,53] and estimate higher-order derivative terms, f_i considers the last k solutions via multiple input channels (rather than just y_{i-1}). One could potentially achieve this property, and perhaps obtain better results, by using GRU or LSTM modules [33] which learn a “latent” state to pass between iterations; however, 3D convolutional implementations for these operations were not part of major ML libraries at the time of writing.

Specifically for elastic localization, \mathbf{m} and y are combined via element-wise multiplication following Eq. (11). To enforce computational stability, all strains are normalized by the average strain $\bar{\epsilon}$. Moreover, the output of each f_i network has its average subtracted to enforce the constraint that the perturbation strain field is always zero-mean.

Following prior work [19], we define a full RLN as using a different f_i for each iteration (although we use the same CNN structure for each), for a total of N distinct networks. The idea behind this is to allow the network capture different properties at each iteration, akin to terms in a series expansion. Having significantly more tunable parameters, this configuration provides the most expressive model. By allowing different operators to be employed at different iterations, this approach also deviates the most from standard analytical optimization procedures.

Alternatively, one may wish to reuse the same intermediate network across iterations ($f_i = f \forall i$). This approach is denoted as RLN-t since the weights are *tied* between iterations. This means that the same operator will be employed at each iteration; however, since a time series is fed into each f_i (rather than a single data point), the RLN-t is still able to learn higher-order derivatives. It has the primary advantage of simplicity and efficiency, since it uses a factor of N fewer parameters than the full RLN.

Finally, we test the importance of the iterative and recurrent nature by considering a single f_i network, i.e., choosing $N = 1$. We call this a feed-forward localization network (FLN) and use it as a control to quantify the benefits of iteration vs. network design. Although the RLN-t and the FLN have the same number of parameters, the RLN-t uses each parameter N times, effectively simulating a deeper network.

For $N > 1$, the proximal CNNs are all calibrated simultaneously via backpropagation. During training a batch of true structure-strain pairs are fed through \mathcal{F}_θ in its entirety, and all proximal networks are updated simultaneously to minimize a calibration loss function $\mathcal{L}^{(cal)}$ (which is not necessarily the same as the solution field loss \mathcal{L} above). Rather than only consider the loss of the final output, we use a weighted sum of the loss of individual iterates: $\mathcal{L}^{(cal)} = \sum_i w_i \mathcal{L}_i^{(cal)}$ for some weights w_i . The goal is to encourage the network to progress between iterations, while also finding the best possible solution.

Following the analysis of Andrychowicz et al. [49] we interpret the use of $\mathcal{L}^{(cal)}$ as a variant of Backpropagation Through Time [54]. The choice of weights w_i could theoretically act as a form of stabilization or even regularization. By requiring that each iteration output a reasonable candidate solution, each proximal operator is constrained to behave somewhat physically, which might help prevent overfitting. However, this means that the network is encouraged to make larger changes in early iterations, potentially reducing its final-iterate accuracy. Conversely, if only the final result is important, then intermediate iterations could explore the loss curve more. However, only updating based on the last iteration will slow the model’s training, and possibly increase the chances of the network weights converging to a poor local minimum. Clearly further experiments are required to explore these hypotheses.

Following similar works [33] we chose the uniform weighting $w_i = 1 \forall i$. This induces the network to make larger corrections in early iterations (to avoid carrying costly errors through several iterations) and relatively smaller corrections in later iterations. However, our numerical

experiments (Section 4) indicate that perhaps a different weighting might help the network capture fine-scale microstructure features. The appropriate number of iterations depends on the specific problem; for elasticity $N = 5$ proved sufficient in both the RLN and the RLN-t, and more iterations yielded little benefit. For the number of previous iterations to track, the value $k = 2$ was chosen. The above choices of hyperparameter were largely heuristic and made for simplicity, but they worked well for elasticity; for a more intensive problem a cross-validation procedure would be a better method for their selection.

3.3. Proximal operator design

Various network topologies for f_i [19,33,55] were tested with one goal in mind: use convolutional kernels to effectively capture local interactions. The architecture that proved most successful for elasticity was based roughly on a V-Net [55] and is presented in Fig. 3. By combining information across length scales through the use of up- and down-sampling operations, this architecture is able to encode and combine both fine- and coarse-grained features. Notably, even a simple sequence of 3 convolutional layers (similar to that of Adler [19]) worked reasonably well. As with the hyperparameters above, a cross-validation procedure could aid in picking a superior network topology for a given problem; for this work, we intentionally avoided hyperparameter optimization to emphasize the mathematical analysis and iterative approach.

For this study, most convolutional kernels are $3 \times 3 \times 3$, with 1×1 kernels used for channel reduction and $2 \times 2 \times 2$ for up/downsampling, summing to $\approx 160,000$ learnable parameters for each proximal network f_i . A parametric rectified linear unit (PReLU) activation function [56] was used after the $3 \times 3 \times 3$ convolutions. This activation is similar to the regular ReLU, but has a non-zero slope α for negative inputs; α is trained with all the other network weights during back-propagation. In our experiments this improved the model's stability during training and accuracy during testing. Effectively, it allows the network to be much more expressive with only a few extra parameters – changing one α scales the entire output of a channel, rather than $3 \times 3 \times 3 = 27$ weights representing each voxel in that channel's filter.

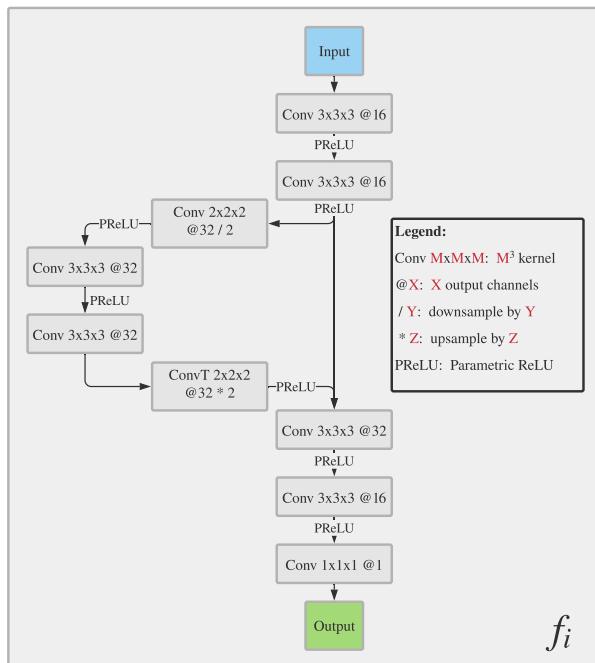


Fig. 3. Architecture for internal f_i networks.

4. Linear elasticity experiments

We now present results from the application of RLN-type models to the elastic localization problem in two-phase composites. A synthetic dataset of 20,480 $31 \times 31 \times 31$ microstructure/strain field pairs was generated and randomly split 40% / 20% / 40% into disjoint train/validation/test sets, respectively. The train set was used to calibrate \mathcal{F}_0 , the validation set served to measure its quality while training, and the test set was used to compute error metrics.

The microstructures were generated via PyMKS [25] by creating a uniform random field, applying a set of Gaussian filters, and thresholding the result. In this microstructure generation process, the filter's width in each direction controls the characteristic size and shape of the material grains, and the threshold controls the relative volume fraction of each phase.

Combinations of these four parameters were generated via a Latin hypercube sampling procedure to generate 4096 design points for microstructure generation. For each design point, 5 random microstructures were generated in order to produce statistically similar inputs, so that the datasets had some statistical redundancy. A random sample of 8 training microstructures is displayed in Fig. 4. By varying these design parameters, one can cover a vast subspace of all possible microstructures, although we note that the space of all microstructures is intractably large – ignoring circular symmetries there are $O(2^S)$ possible different microstructures with S voxels. By selecting our training and testing sets using the same procedure, we demonstrate the RLN's ability to interpolate between “familiar” microstructure samples, but not to extrapolate to microstructures which lie outside the closure of the training set.

It is important to note that these microstructures do not necessarily correspond to thermodynamically favorable structures (i.e., they might appear in nature). However, this is a limitation only in data and not methodology – by using a fully-convolutional structure [57] the RLN can handle any voxelized input, including experimental data.

The elastic strain fields in each microstructure were obtained from previously-built finite element models [22]. One major simplification is that even though the imposed 3D strain field is actually a second-order tensor (and the elastic stiffness field a fourth-order tensor), the linear nature of the problem allows us to solve for each component individually and superimpose their solutions as needed [35]. As such we apply an overall displacement to a material RVE along only the x -axis and we focus solely on the ϵ_{xx} term. For these simulations, a total strain of 0.1% was applied via periodic boundary conditions.

The relevant material parameters to be prescribed are thus the elastic moduli E_1, E_2 and Poisson ratios ν_1, ν_2 of the two phases. To roughly match most common metals, we chose $\nu_1 = \nu_2 = 0.3$. With these selections, the contrast ratio $CR \equiv \frac{E_2}{E_1}$ has the most dominant role on the final strain field. With the choice $E_2 \geq E_1$, one observes that $CR \geq 1$. In general, as the contrast in stiffnesses increases, the problem becomes harder to solve with both iterative and data-driven methods [32]. Following prior work [22], we tested the RLN on contrast ratios of 10 and 50.

Each RLN model was implemented in PyTorch [58] and trained independently for 60 epochs on an NVIDIA V100 GPU [59], which took approximately 8 hours. The RLN models were all calibrated using the Adam optimizer, a Mean Square Error loss, and a Cosine annealing learning rate decay [60]. The last choice proved especially important since the models demonstrated great sensitivity to learning rate; introducing the cosine decay helped the model converge smoothly and quickly. After training, the epoch with the lowest validation loss was chosen as the “optimal”. In reality, the training and validation losses were tightly coupled as demonstrated in Fig. 5. Note that training losses are aggregated during the epoch, whereas validation losses are computed after the epoch is complete, which causes the validation loss to occasionally appear lower.

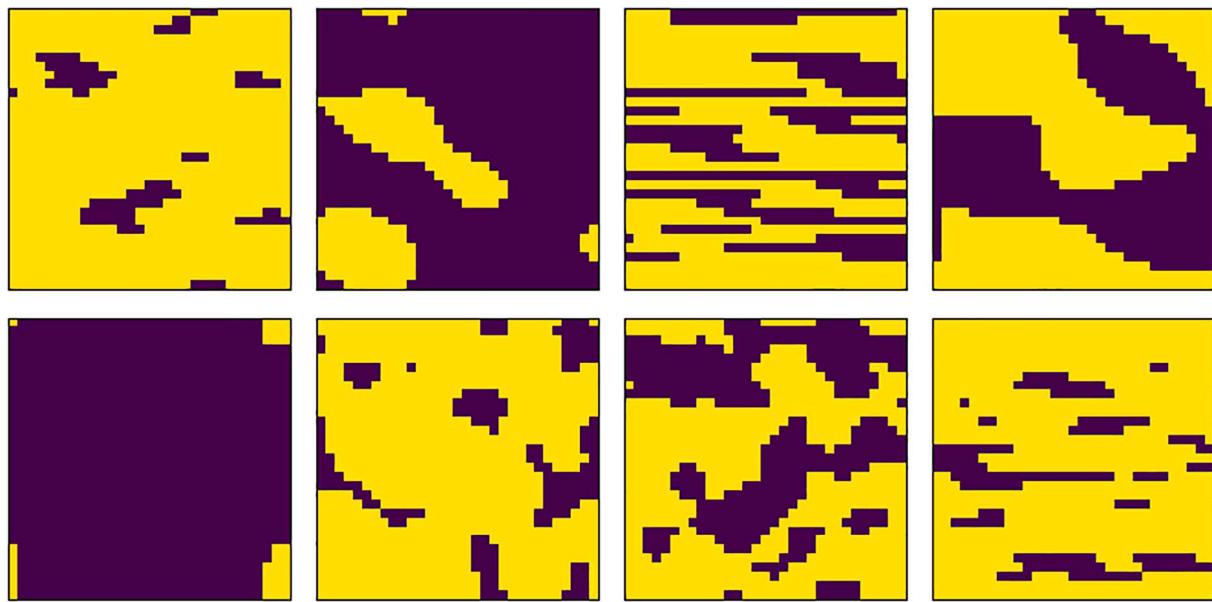


Fig. 4. Random sample of 8 training microstructures sliced along (y, z) axis. Yellow is high-stiffness phase, purple is low-stiffness.

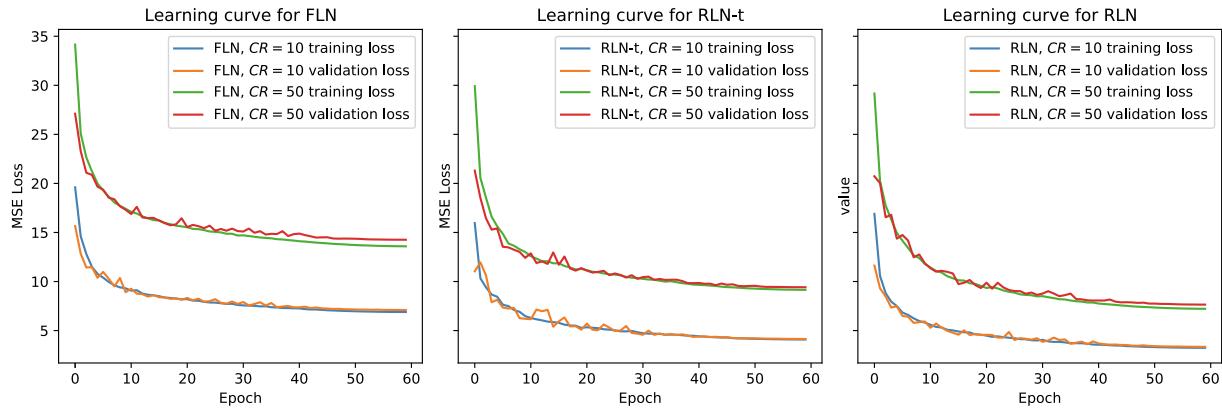


Fig. 5. Learning curve containing calibration losses for each model and contrast ratio.

4.1. Results

Following Yang et al. [22], the accuracy of the model was evaluated for each instance using the mean absolute strain error (MASE) over all S voxels:

$$MASE\left(\mathbf{y}_{pred}, \mathbf{y}_{true}\right) = 100 \times \frac{1}{S} \sum_{i=1}^S \left| \frac{\mathbf{y}_{pred}[i] - \mathbf{y}_{true}[i]}{\mathbf{y}^R} \right|. \quad (28)$$

Fig. 6 presents the MASE distribution across each microstructure/strain pair in the test set. Note that the MASE is an aggregate measure which measures RVE-wide error; the pointwise error variation within a microstructure is explored below. The mean and standard deviation of the MASE distribution are collected in Table 1 for the RLN-type models. For comparison we also present results from a recent study [22] using a feed-forward deep learning (DL) model to predict the strain fields; note that the DL model was trained and tested on its own dataset prior to this effort.

It is important to note that the DL model had a fundamentally different architecture: it was designed to work on 21^3 voxel structures (whereas ours was tested on 31^3), and it predicted the strain one voxel at a time (whereas ours predicts strain across the entire microstructure simultaneously). The dataset for the DL model employed large amounts of data augmentation using circular permutations, and contained

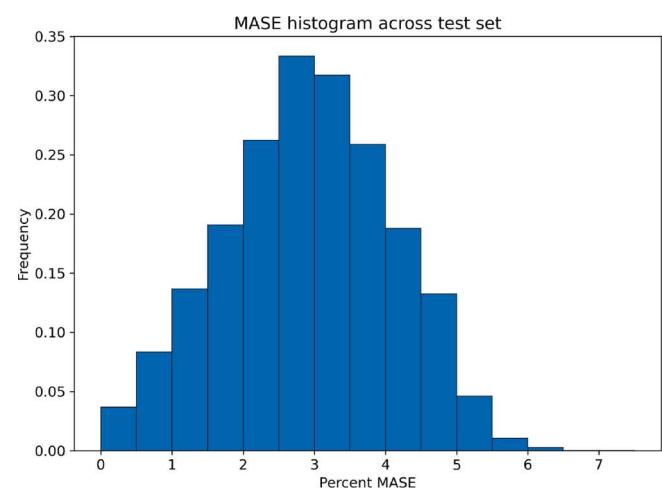


Fig. 6. MASE distribution for RLN, CR = 50.

significantly more input-output pairs. Finally, the DL model employed linear layers (to collapse to a single output); the size of these layers implies that the DL model used substantially more network weights than

Table 1

Test-set MASE metrics for RLN and control models, for both $CR = 10$ and $CR = 50$

Model	MASE (mean \pm std. dev.)
<i>Contrast-10</i>	
Comparison DL model [22]	3.07% \pm 1.22%
FLN	4.98% \pm 1.49%
RLN-t	1.81% \pm 0.58%
RLN	1.21% \pm 0.37%
<i>Contrast-50</i>	
Comparison DL model	5.71% \pm 2.46%
FLN	9.23% \pm 3.29%
RLN-t	4.26% \pm 1.65%
RLN	2.92% \pm 1.17%

the RLN. As a result, it is difficult to compare the DL dataset and results with ours. We emphasize that the DL model represents a strong comparison model for ML elastic localization, but that objective ranking of relative performance is difficult. Nevertheless, the RLN architecture is able to produce significantly more accurate strain field estimates on the RLN dataset than the DL architecture produces on the DL dataset. Keeping these caveats in mind, the following analysis explores the difference between RLN configurations.

Looking at the aggregate statistics, the FLN performs worst of all models analyzed; it is vastly outperformed by the RLN-t even though they have the same number of parameters. This is also reflected in the learning curve in Fig. 5: the RLN-t trained faster than the FLN and converged to a more accurate model simply by applying the proximal operator repeatedly across multiple iterations. Intriguingly, the FLN results are somewhat worse than the DL results. This implies that our simple network topology and relative lack of hyperparameter tuning produced a less-powerful model than the DL. However, the RLN-t did much better, producing less error on average (and significantly less variance) than the DL control for both contrast ratios. The disparity in performance indicates that dataset and network topology alone cannot explain the improved MASE relative to the DL model – the iterative methodology produces a more powerful model.

The full RLN is the most accurate model, producing roughly half as much error and variability as the DL model for both contrast ratios. The improvement over the RLN-t has a number of possible origins: the RLN uses a factor of N more parameters, and in turn it uses a different operator at each iteration. We note that after training, the full RLN increases the memory overhead but not the prediction time: up to GPU memory details, each iteration has the same computational costs regardless of weight tying. Excluding data I/O overhead, the RLN-t and the full RLN take only ≈ 87 seconds to predict strain fields for the entire testing set (8,192 microstructures), or roughly 11 ms per microstructure (c.f. ≈ 5 s per microstructure for FEA).

Fig. 7 presents the worst (by voxel) test-set slice for the RLN model

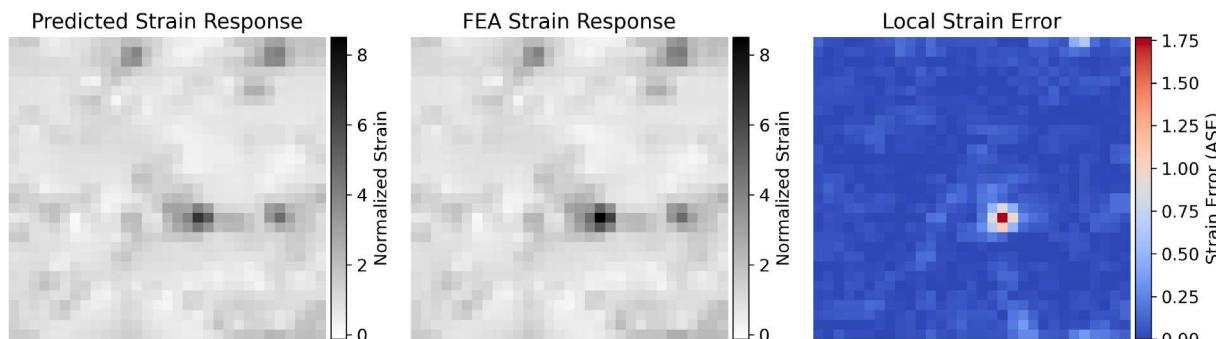


Fig. 7. Slices of RLN predicted strain, true (FEA) strain, and ASE for worst test-set instance, $CR = 50$.

compared to the FEA-generated fields. The RLN appears to perform poorest near extreme strain peaks, especially in a 3-voxel wide cube around these peaks. This is caused by two issues. First, the microstructure is effectively undersampled in these areas. Referring back to Fig. 4, one sees that many microstructures have spatial features that are only one or two voxels wide. Furthermore, the output of the RLN is effectively ‘smeared’ near these peaks due to the usage of a $3 \times 3 \times 3$ filter. A deeper network, or one using multiple filter sizes, might be able to better capture these features.

Finally, we explore how the predicted strain field y_i evolves across iterations, as well as its difference $\Delta_i \equiv y_i - y_{i-1}$. This is presented in Fig. 8 for the FLN, Fig. 9 for the RLN-t, and Fig. 10 for the full RLN. The greyscale coloring is the same scale as Fig. 7 and has its maximum at the true strain response’s maximum. The redscale coloring represents a negative strain update (y decreasing between iterations).

The FLN appears to handle low-strain areas fairly well, but misses the strain peaks in the center. Note that it does not apply the same operation as the first iteration of the RLN-t; the FLN is trained to make the best possible answer within a single jump. In comparison, the RLN-t does a better job of capturing strain peaks, likely since it can build them up over several iterations. What is less clear is that it fails to capture the central strain peak as well as the full RLN – the differences between the two are evidently rather fine-scale, so we focus our analysis on the full RLN.

After the first iteration the RLN has picked up most of the relative troughs and peaks, and finer tweaks are handled by the later iterations. For example, most of the high-strain regions (greyscale deltas) appear to be captured in the first two iterations, whereas low-strain regions

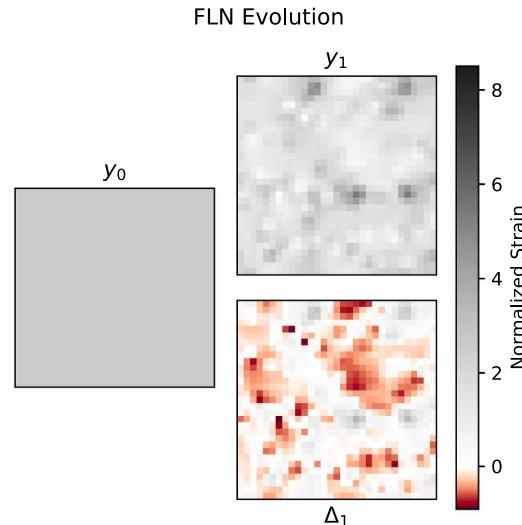


Fig. 8. FLN predictions and differences between iterations for selected test-set instance, $CR = 50$.

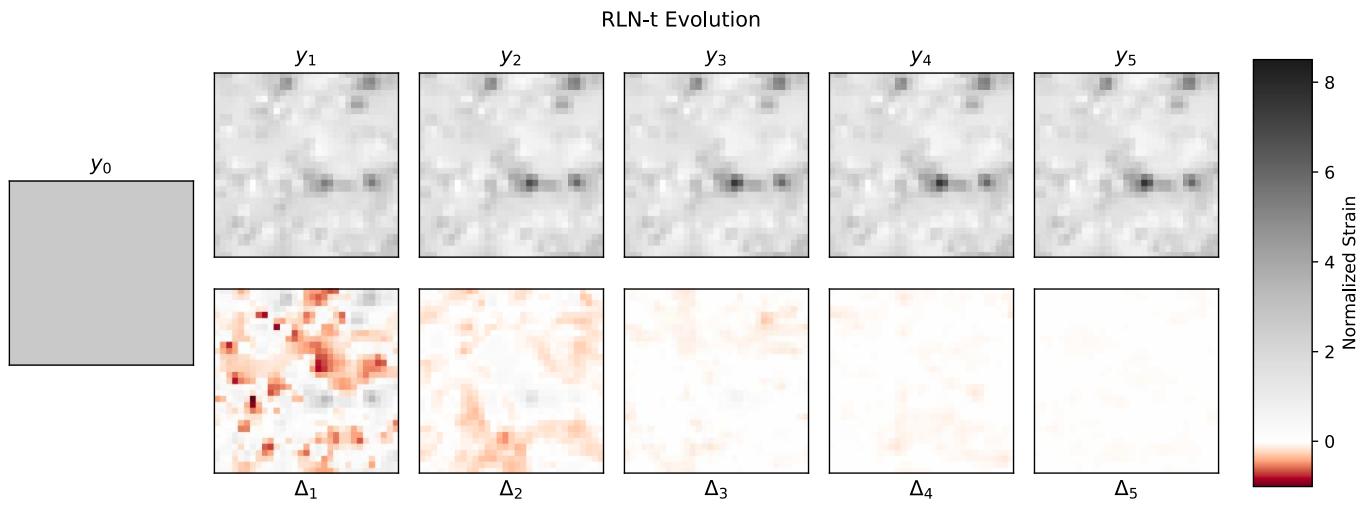


Fig. 9. RLN-t predictions and differences between iterations for selected test-set instance, $CR = 50$.

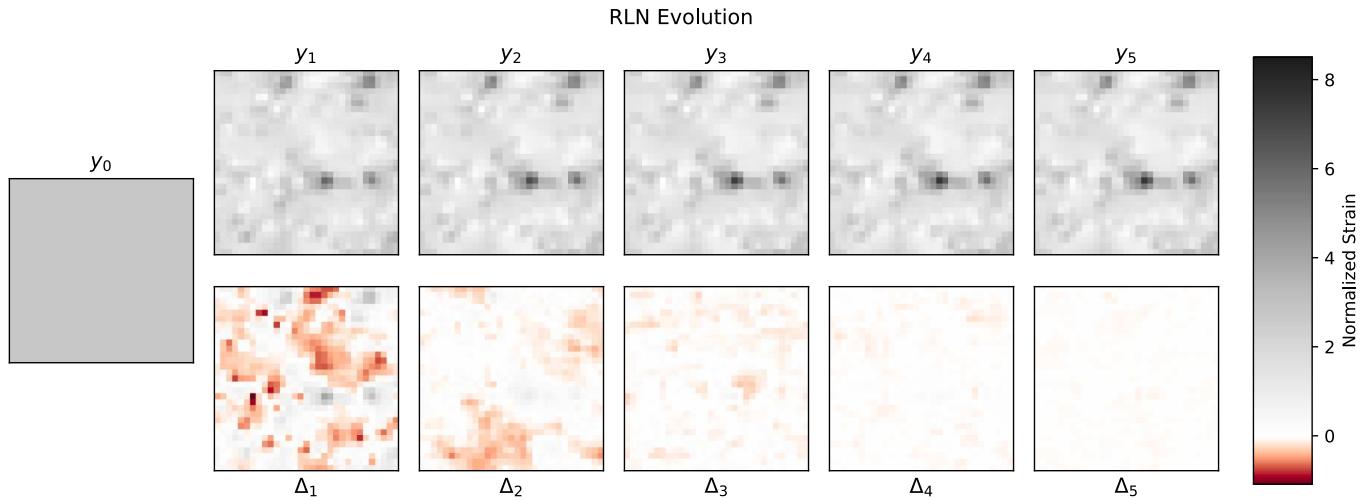


Fig. 10. RLN predictions and differences between iterations for selected test-set instance, $CR = 50$.

(redscale deltas) continue to be refined in later iterations.

This is due to the RLN's ability to learn different, and nonlinear, operators at each iteration – the MSE loss function tends to magnify high-magnitude errors, even if they are very localized (i.e., strain peaks and troughs). The model is therefore encouraged to estimate these outlier areas first (corresponding to Δ_1 having much more magnitude). Of course, this puts a lot of weight on the first proximal network to capture the strain outliers correctly. One possible solution would be to adjust the loss function weighting so that early iterations are encouraged to converge gradually towards a good solution, rather than quickly towards a decent one. In other words, by allowing earlier iterations to mispredict strain peaks somewhat, the model may be more likely to escape local minima and actually obtain higher final-iteration accuracy.

This approach differs from traditional finite-element based approaches in that it seeks a solution by learning Green's functions (and derivatives thereof) of the governing equation, rather than solving the governing equation numerically. This provides an advantage in prediction speed by requiring a rather costly (but one-time) training overhead. The computational complexity of convolving a 3D field containing S voxels with a 3D stencil containing k voxels is $O(Sk)$ (since each voxel in the first field must be multiplied by each voxel in the stencil). For a fixed network topology, k is a constant; therefore the prediction runtime will increase linearly with the number of microstructure voxels. A spectral method using the Fast Fourier Transform will cost at least $O(S \log S)$, and

any numerical methods (finite element or otherwise) employing linear solvers will likely be even more expensive. We reiterate that using GPU parallelization, the RLN requires on average 11 ms to predict the strain field for a given microstructure, compared to several seconds for the finite element solver. This makes it very valuable for inverse problems, where the localization problem must be solved thousands or even millions of times in order to solve a higher-level metaproblem [15]. Once trained for a specific governing equation (e.g. linear elasticity) and set of material properties (e.g., contrast ratio), the RLN methodology can be applied to any voxelized microstructure. Although we only tested a 31^3 structure, in principle a model could be trained on one structure size and used on another (possibly with reduced accuracy); this is the subject of ongoing work. Note that the model must be trained anew to predict strains for a different contrast ratio.

5. Conclusions

In this paper, we describe a new learning-based methodology for addressing Lippmann-Schwinger type physics problems. Embedding recurrent CNNs into a learned optimization procedure provides for a flexible, but interpretable, ML model. The design of proximal networks is informed by problem-specific domain knowledge; that knowledge also provides a physical interpretation of what the model is learning. Furthermore, the partitioned and convolutional structure of this

approach acts as a regularizer by enforcing underlying physical properties such as mean field values and spatial invariance. The iterative scheme allows for emulation of a deeper network, vastly increasing model robustness without increasing the parameterization space. If space allows, using a different network for each iteration further improves the model's expressiveness and accuracy.

When applied to the elasticity localization problem and using our dataset, our model produced much more accurate and interpretable results than previous deep learning models produced on similar datasets, while being much faster than analytical approaches. The CNN architecture used here was designed for simplicity, but could be improved with more advanced techniques such as inception modules [61], perceptual losses [62], Fourier layers [63], or variational layers [64]. Moreover, many hyperparameters, such as number of iterations and loss function weighting, can be tuned on a problem-specific basis.

CRediT authorship contribution statement

Conlain Kelly: Conceptualization, Methodology, Software, Data curation, Investigation, Writing - original draft, Writing - review & editing. **Surya R. Kalidindi:** Supervision, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by NSF Graduate Research Fellowship DGE-1650044, and SK acknowledges support from NSF 2027105. We used the Hive cluster supported by NSF 1828187 and managed by PACE at Georgia Institute of Technology, USA. The authors thank Anne Hanna, Andreas Robertson, Nic Olsen, and Mady Veith for insightful discussions that helped shape this work. Data and Software Availability The raw and processed data required to reproduce these findings are available to download from [dataset]<https://doi.org/10.17632/v6dt8dwrh8.2> [65]. The implementation and training code for each RLN configuration, as well as trained models, are available under an open source license on Github (see Ref. [66]).

References

- [1] S. Kalidindi, Hierarchical Materials Informatics: Novel Analytics for Materials Data, Butterworth-Heinemann, Boston, 2015.<https://doi.org/10.1016/B978-0-12-410394-8.00001-1>. URL: <http://www.sciencedirect.com/science/article/pii/B9780124103948000011>.
- [2] NIST, Materials Genome Initiative Strategic Plan, 2014. URL: https://www.nist.gov/system/files/documents/2018/06/26/mgi_strategic_plan_-dec_2014.pdf.
- [3] K.W. Morton, D.F. Mayers, Numerical Solution of Partial Differential Equations: An Introduction, second ed., Cambridge University Press, 2005, 10.1017/CBO9780511812248.
- [4] F. Roters, P. Eisenlohr, L. Hantcherli, D. Tjahjanto, T. Bieler, D. Raabe, Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: theory, experiments, applications, *Acta Mater.* 58 (4) (2010) 1152–1211, <https://doi.org/10.1016/j.actamat.2009.10.058>. URL: <http://www.sciencedirect.com/science/article/pii/S1359645409007617>.
- [5] M. Cheng, J. Warren, Controlling the accuracy of unconditionally stable algorithms in the cahn-hilliard equation, *Phys. Rev. E* 75 (2007), 017702, <https://doi.org/10.1103/PhysRevE.75.017702>.
- [6] D. McDowell, J. Panchal, H.-J. Choi, C. Seepersad, J. Allen, F. Mistree, Integrated Design of Multiscale, Multifunctional Materials and Products, 2009.<https://doi.org/10.1016/C2009-0-20058-4>.
- [7] H. Moulinec, P. Suquet, A numerical method for computing the overall response of nonlinear composites with complex microstructure, *Comput. Methods Appl. Mech. Eng.* 157 (1) (1998) 69–94, [https://doi.org/10.1016/S0045-7825\(97\)00218-1](https://doi.org/10.1016/S0045-7825(97)00218-1). URL: <http://www.sciencedirect.com/science/article/pii/S0045782597002181>.
- [8] T. de Geus, J. Vondréjc, J. Zeman, R. Peerlings, M. Geers, Finite strain fft-based non-linear solvers made simple, *Comput. Methods Appl. Mech. Eng.* 318 (2017) 412–430, <https://doi.org/10.1016/j.cma.2016.12.032>.
- [9] J.C. Michel, H. Moulinec, P. Suquet, A computational scheme for linear and non-linear composites with arbitrary phase contrast, *Int. J. Numer. Methods Eng.* 52 (12) (2001) 139–160, <https://doi.org/10.1002/nme.275>.
- [10] R. Snieder, Inverse problems in geophysics, in: *Signal Recovery and Synthesis*, Optical Society of America, 2001, p. SMA2. doi:10.1364/SRS.2001.SMA2. URL: <http://www.osapublishing.org/abstract.cfm?URI=SRS-2001-SMA2>.
- [11] X. Wu, G. Proust, M. Knezevic, S. Kalidindi, Elastic-plastic property closures for hexagonal close-packed polycrystalline metals using first-order bounding theories, *Acta Mater.* 55 (8) (2007) 2729–2737, <https://doi.org/10.1016/j.actamat.2006.12.010>. URL: <http://www.sciencedirect.com/science/article/pii/S1359645407000171>.
- [12] A. Jain, J. Bollinger, T. Truskett, Inverse methods for material design, *AIChE J.* 60. <https://doi.org/10.1002/aic.14491>.
- [13] M. Parno, T. Moselhy, Y. Marzouk, A multiscale strategy for bayesian inference using transport maps, *SIAM/ASA J. Uncertainty Quantification* 4 (1) (2016) 1160–1190, <https://doi.org/10.1137/15M1032478>.
- [14] M. Horstemeyer, Multiscale Modeling: A Review, 2009, pp. 87–135.https://doi.org/10.1007/978-90-481-2687-3_4.
- [15] C.-T. Chen, G.X. Gu, Generative deep neural networks for inverse materials design using backpropagation and active learning, *Adv. Sci.* 7 (5) (2020) 1902607, <https://doi.org/10.1002/advs.201902607>.
- [16] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, *Rev. Mod. Phys.* 91 (2019), 045002, <https://doi.org/10.1103/RevModPhys.91.045002>.
- [17] J. Schmidhuber, Deep learning in neural networks: an overview, *Neural Netw.* 61 (2015) 85–117, <https://doi.org/10.1016/j.neunet.2014.09.003>.
- [18] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, *Neural Inf. Process. Syst.* 25.<https://doi.org/10.1145/3065386>.
- [19] J. Adler, O. Oktem, Learned primal-dual reconstruction, *IEEE Trans. Med. Imaging* 37 (6) (2018) 1322–1332, <https://doi.org/10.1109/TMI.2018.2799231>.
- [20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, *CoRR abs/1512.0385*. arXiv:1512.0385. URL: <http://arxiv.org/abs/1512.0385>.
- [21] Z. Yang, Y.C. Yabansu, R. Al-Bahra尼, W. keng Liao, A.N. Choudhary, S.R. Kalidindi, A. Agrawal, Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets, *Comput. Mater. Sci.* 151 (2018) 278–287, <https://doi.org/10.1016/j.commatsci.2018.05.014>. URL: <http://www.sciencedirect.com/science/article/pii/S0927025618303215>.
- [22] Z. Yang, Y.C. Yabansu, D. Jha, W. keng Liao, A.N. Choudhary, S.R. Kalidindi, A. Agrawal, Establishing structure-property localization linkages for elastic deformation of three-dimensional high contrast composites using deep learning approaches, *Acta Mater.* 166 (2019) 335–345, <https://doi.org/10.1016/j.actamat.2018.12.045>.
- [23] J. Schmidt, M.R.G. Marques, S. Botti, M.A.L. Marques, Recent advances and applications of machine learning in solid-state materials science, *NPJ Comput. Mater.* 5 (2019) 1–36.
- [24] V. Buhrmester, D. Münch, M. Arens, Analysis of explainers of black box deep neural networks for computer vision: a survey (2019). arXiv:1911.12116.
- [25] D. Brough, D. Wheeler, S. Kalidindi, Materials knowledge systems in python – a data science framework for accelerated development of hierarchical materials, *Integr. Mater. Manuf. Innov.* 6 (2016) 1–18, <https://doi.org/10.1007/s40192-017-0089-0>.
- [26] M.I. Latypov, L.S. Toth, S.R. Kalidindi, Materials knowledge system for nonlinear composites, *Comput. Methods Appl. Mech. Eng.* 346 (2018) 180–196.
- [27] S.R. Kalidindi, A bayesian framework for materials knowledge systems, *MRS Commun.* 9 (2) (2019) 518–531, <https://doi.org/10.1557/mrc.2019.56>.
- [28] S. Kalidindi, A. Khosrovani, B. Yucel, A. Shanker, A. Blekh, Data infrastructure elements in support of accelerated materials innovation: Ela, pymks, and matin, *Integr. Mater. Manuf. Innov.* (2019) 1–14, <https://doi.org/10.1007/s40192-019-00156-1>.
- [29] B.A. Lippmann, J. Schwinger, Variational principles for scattering processes. i, *Phys. Rev.* 79 (1950) 469–480, <https://doi.org/10.1103/PhysRev.79.469>.
- [30] D.B. Brough, D. Wheeler, J.A. Warren, S.R. Kalidindi, Microstructure-based knowledge systems for capturing process-structure evolution linkages, *Curr. Opin. Solid State Mater. Sci.* 21(3) (2017) 129–140, materials Informatics: Insights, Infrastructure, and Methods.<https://doi.org/10.1016/j.cossms.2016.05.002>. URL: <http://www.sciencedirect.com/science/article/pii/S1359028616300298>.
- [31] M.W. Priddy, N.H. Paulson, S.R. Kalidindi, D.L. McDowell, Strategies for rapid parametric assessment of microstructure-sensitive fatigue for hcp polycrystals, *Int. J. Fatigue* 104 (2017) 231–242, <https://doi.org/10.1016/j.ijfatigue.2017.07.015>. URL: <http://www.sciencedirect.com/science/article/pii/S0142112317303055>.
- [32] R.A. Lebensohn, A.D. Rollett, Spectral methods for full-field micromechanical modelling of polycrystalline materials, *Comput. Mater. Sci.* 173 (2020), 109336, <https://doi.org/10.1016/j.commatsci.2019.109336>. URL: <http://www.sciencedirect.com/science/article/pii/S0927025619306354>.
- [33] P. Putzky, M. Welling, Recurrent inference machines for solving inverse problems (2017). arXiv:1706.04008.
- [34] M. Stein, Gradient methods in the solution of systems of linear equations, *J. Res. Natl. Bureau Standards* 48 (1952) 407.
- [35] G. Landi, S.R. Niezgoda, S.R. Kalidindi, Multi-scale modeling of elastic response of three-dimensional voxel-based microstructure datasets using novel dft-based knowledge systems, *Acta Mater.* 58 (7) (2010) 2716–2725, <https://doi.org/10.1016/j.actamat.2010.01.007>. URL: <http://www.sciencedirect.com/science/article/pii/S1359645410000108>.
- [36] E. Kröner, Statistical continuum mechanics: course held...October 1971, Springer, 1972.

[37] G. Green, An essay on the application of mathematical analysis to the theories of electricity and magnetism (1828). arXiv:0807.0088.

[38] T. Eisler, An introduction to Green's functions, 1969.

[39] Wikipedia, https://en.wikipedia.org/w/index.php?title=Green%27s_function&oldid=966577686 (Jul 2020).

[40] R.A. Leibniz, J.P. Escobedo, E.K. Cerreta, D. Dennis-Koller, C.A. Bronkhorst, J. F. Bingert, Modeling void growth in polycrystalline materials, *Acta Mater.* 61 (18) (2013) 6918–6932, <https://doi.org/10.1016/j.actamat.2013.08.004>. URL: <http://www.sciencedirect.com/science/article/pii/S1359645413005909>.

[41] W.S. McCulloch, W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, MIT Press, Cambridge, MA, USA, 1988, pp. 15–27.

[42] F. Rosenblatt, *The Perceptron, A Perceiving and Recognizing Automaton Project Para, Report: Cornell Aeronautical Laboratory, Cornell Aeronautical Laboratory, 1957*.

[43] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Control Signals Syst.* 2 (1989) 303–314.

[44] A. Pinkus, Approximation theory of the mlp model in neural networks, *Acta Numerica* 8 (1999) 143–195.

[45] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* 86 (11) (1998) 2278–2324.

[46] Z.C. Lipton, J. Berkowitz, C. Elkan, A critical review of recurrent neural networks for sequence learning (2015). arXiv:1506.00019.

[47] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, M. Dehmer, An introductory review of deep learning for prediction models with big data, in: *Frontiers in Artificial Intelligence*, 2020.

[48] V.K. Ojha, A. Abraham, V. Snásel, Metaheuristic design of feedforward neural networks: a review of two decades of research, *CoRR* abs/1705.05584. arXiv: 1705.05584. <http://arxiv.org/abs/1705.05584>.

[49] M. Andrychowicz, M. Denil, S. Gomez, M.W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, N. de Freitas, Learning to learn by gradient descent by gradient descent (2016). arXiv:1606.04474.

[50] F.N. Iandola, M.W. Moskewicz, K. Ashraf, S. Han, W.J. Dally, K. Keutzer, SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size, *CoRR* abs/1602.07360. arXiv:1602.07360. <http://arxiv.org/abs/1602.07360>.

[51] D.E. Rumelhart, G.E. Hinton, R.J. Williams, *Learning Representations by Back-Propagating Errors*, MIT Press, Cambridge, MA, USA, 1988, pp. 696–699.

[52] R.M. Gray, 2006. doi:10.1561/0100000006.

[53] M.Z. Ullah, S. Serra-Capizzano, F. Ahmad, An efficient multi-step iterative method for computing the numerical solution of systems of nonlinear equations associated with odes, *Appl. Math. Comput.* 250 (2015) 249–259, <https://doi.org/10.1016/j.amc.2014.10.103>. URL: <http://www.sciencedirect.com/science/article/pii/S0096300314014787>.

[54] M. Mozer, A focused backpropagation algorithm for temporal pattern recognition, *Complex Syst.* 3.

[55] F. Milletari, N. Navab, S.-A. Ahmadi, V-net: Fully convolutional neural networks for volumetric medical image segmentation (2016). arXiv:1606.04797.

[56] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015). arXiv:1502.01852.

[57] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation (2015). arXiv:1411.4038.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, Curran Associates Inc, 2019, pp. 8024–8035. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

[59] PACE, Partnership for an Advanced Computing Environment (PACE) (2017).

[60] I. Loshchilov, F. Hutter, Sgdr: Stochastic gradient descent with warm restarts (2017). arXiv:1608.03983.

[61] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S.E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, *CoRR* abs/1409.4842. arXiv:1409.4842. <http://arxiv.org/abs/1409.4842>.

[62] J. Johnson, A. Alahi, F. Li, Perceptual losses for real-time style transfer and super-resolution, *CoRR* abs/1603.08155. arXiv:1603.08155. <http://arxiv.org/abs/1603.08155>.

[63] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential Eqs. (2020). arXiv:2010.08895.

[64] K. Shridhar, F. Laumann, M. Liwicki, A comprehensive guide to bayesian convolutional neural network with variational inference (2019). arXiv: 1901.02731.

[65] C. Kelly, RLNElasticity_Localization. Mendeley Data, V2, <https://doi.org/10.17632/v6dt8dwrb8.2>.

[66] C. Kelly, Rln_elasticity. URL: https://github.com/conlain-k/RLN_elasticity (2020).