10:00-10:25 Wed. (Palm 2) MS78 Machine Learning and Data-Driven Methods for Forward and Inverse Problems
SIAM Conference on Mathematics of Data Science (MDS22)
26–30 September 2022, San Diego, CA

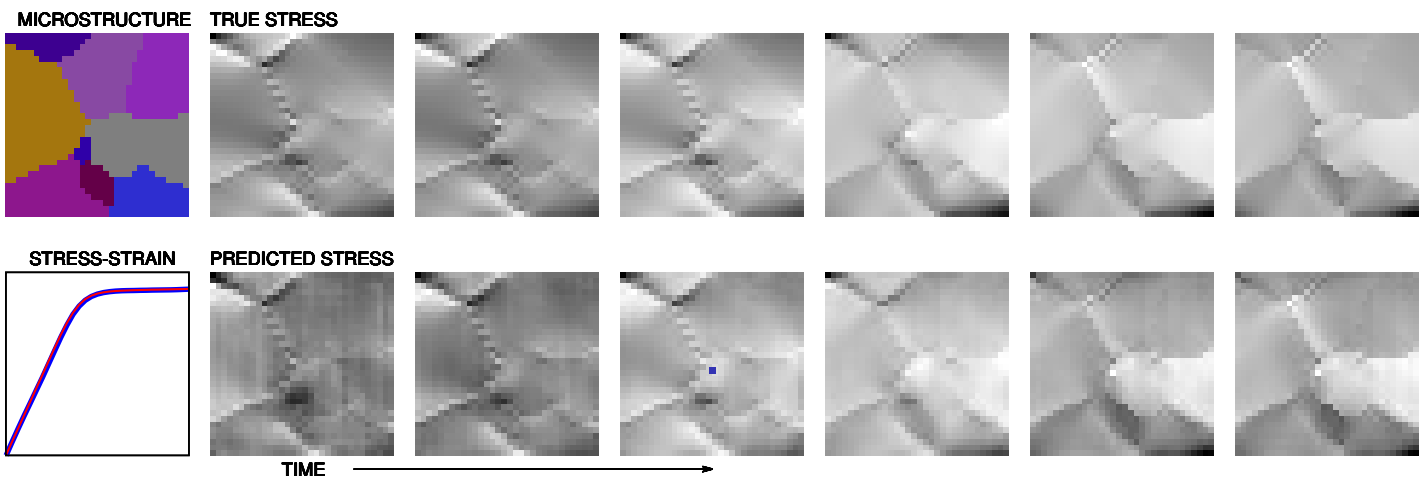
# Machine learning constitutive models of inelastic materials with microstructure

Reese E. Jones, A. Frankel, C. Safta, K.L. Johnson

*Sandia National Laboratories, Livermore, CA 94551, USA*

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY



# Overview: computational physics & mechanics

**Motivation:** compared to governing eqs. & numerical methods, constitutive models are weakness of simulation

**Goal:** efficient accurate surrogate models of material processes

*Everyone is doing machine learning, it is easy and sometimes useful.*

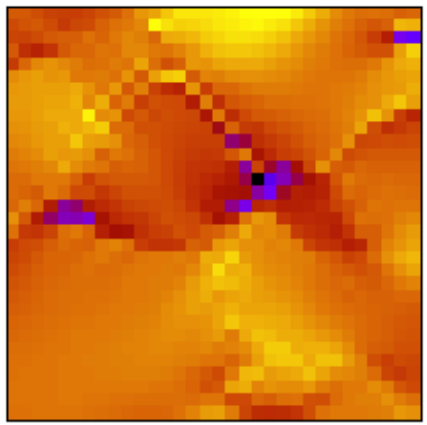
- a paraphrase of George Box

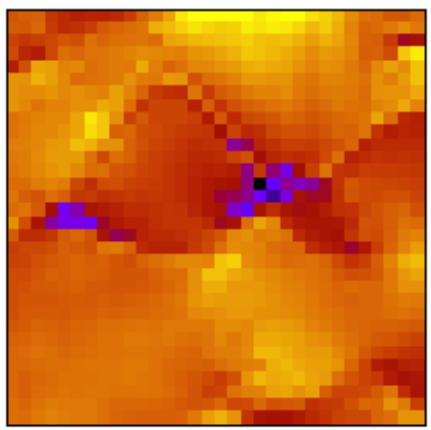
## Outline



**Please ask questions**

**Which one is the ML prediction?**

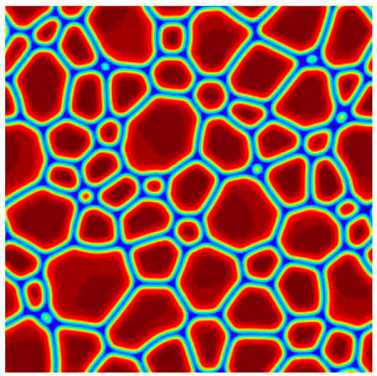




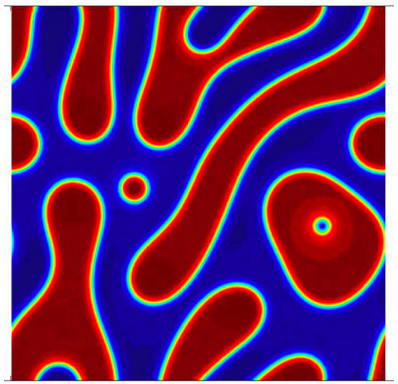
one is a "deep fake"

# Microstructural problems of interest: problem statement

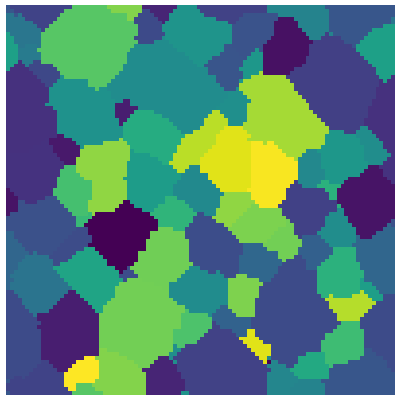
Premise: the state of each of these systems/processes can be encoded as an **image**/field with multiple **channels** $\phi(\mathbf{X})$ predict the output QoI.



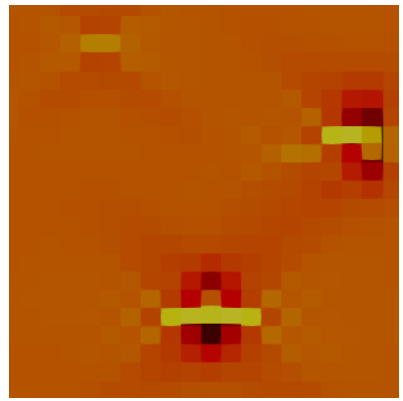
bubbles



multi-phase



polycrystal



pores/inclusions

Classes of closure problems:

- **property estimation**: map initial image $\phi(\mathbf{X})$ to a *static* quantity $\varepsilon$, e.g. diffusivity
- **homogenization**: map initial image $\phi(\mathbf{X})$ and forcing $\epsilon(t)$ to *evolving* scalar quantity $\Psi(t)$, e.g. energy
- **field prediction**: map initial image $\phi(\mathbf{X})$ and forcing $\epsilon(t)$ to an evolving *field* $\sigma(\mathbf{X}, t)$, e.g. stress field

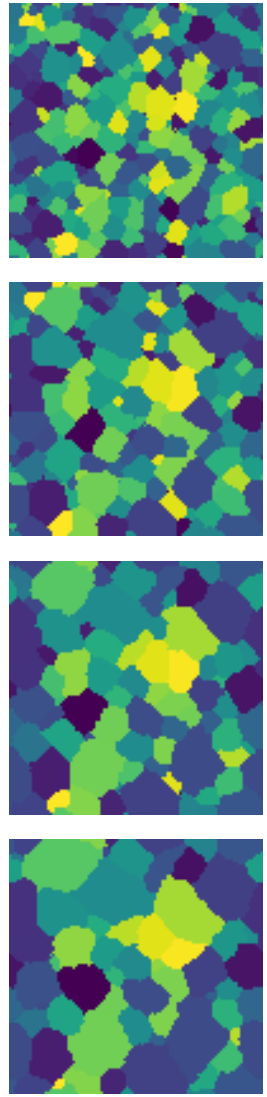
each are specialized PDE solves: e.g. regular domain, nominally homogeneous loading, output QoIs scalars/system averages not fields.

Applications: subgrid models, structure-property exploration /optimization, & material uncertainty quantification

# Microstructural problems of interest: challenges

Homogenization has challenges in common with many ML tasks:

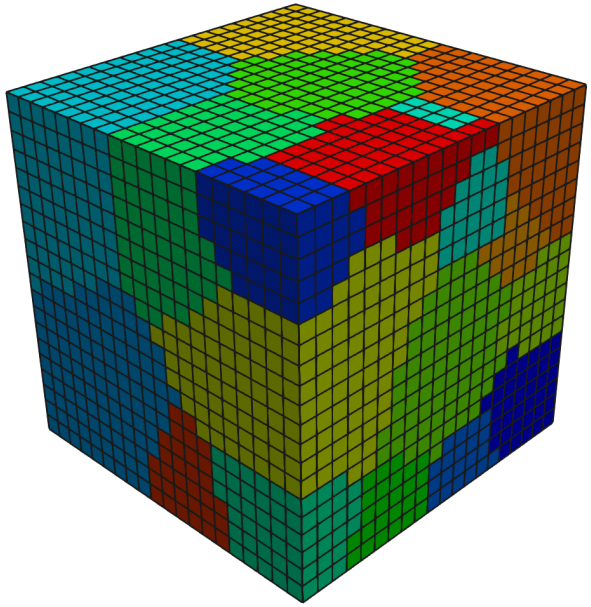
- **encoding the image into a compact, representative latent space** which serves as an initial condition and avoids manual featurization
- **handling the time evolution** of the QoI in the IBVP, e.g RNN (LSTM) / NODE
- **imposing physical constraints & symmetries** e.g. rotational symmetries, 2nd law considerations
- interpretability of the latent space & other network outputs – improved "trustworthiness"
- data sparsity / limited modality (low & limited data) – go beyond training models to models.
- multifidelity / transfer learning
- noisy data



... but there is also **specific domain knowledge coming from classical theory** we use to guide designs.

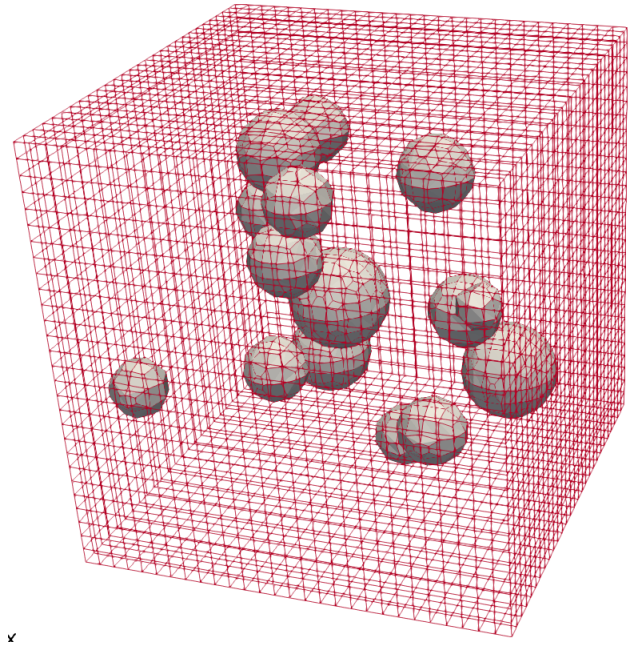
# Exemplars: RVEs of materials with microstructure

**Polycrystalline materials**



- entities: grains/crystals
- neighbor-neighbor interface interactions
- input: orientation angles
- output: stress(time)

**Composite materials**



- entities: inclusions
- all to one inclusion-matrix interactions
- input: phase (label)
- output: stress(time)

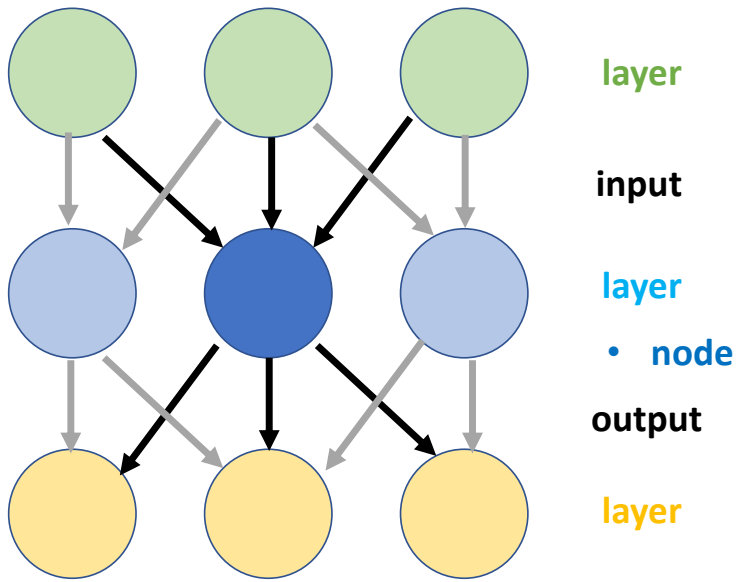
**Goal:** predict the variations in response due to the particular arrangement/configuration of entities in the sample.

# Neural networks - basics/background

The simplest **neural network** (NN) is a *multilayer perceptron* (MLP), a directed graph of densely connected nodes organised in layers. Inputs are weighted, summed and transformed to outputs by *non-linear* ramp/switch-like activation functions.

$$y_j = f\underbrace{\left(\sum_i w_{ij}x_i + b_j\right)}_{\text{linear transform}}$$

The parameters $w$, $b$ are trained via backpropagation and stochastic descent. NN are compounded trainable affine transforms with non-linear maps & can be compact universal approximators.




A NN is basically a functional form to be fit with chosen inputs, output, & information flow. Like LEGOS™, **modular** layers with particular characteristics can be linked to **create architectures that follow physical principles & inspired by traditional modeling frameworks.**

# Deep learning: convolutional neural networks

One of the types of specialized layers is convolution, which relies of spatial or temporal correlation in the data. Convolution addresses many of our challenges:

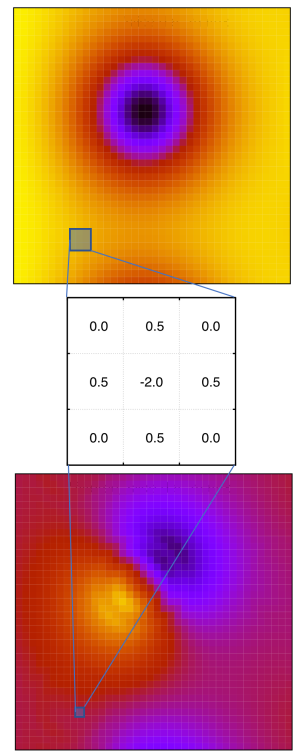
- provides **deep featurization** of the image relevant to the QoI
- approximate **differential operators** to assist emulating the IBVP

**Convolution** with a **kernel** is a standard technique in (time) signal and (spatial) image processing that has been adapted to ML.

$$\text{Size of kernel} \ll \text{size of image}$$

For example, filters can:

- **Smooth/filter noise:** convolving an image with a Gaussian kernel.
- **Average/coarsen:** multiplying with constant moving patch
- **Gradients and higher derivatives:** filter corresponding a finite difference stencil.
- **Features:** edge detection, clustering, segmentation, ...




A convolutional NN trains the weights W and bias b for a (small) kernel and multiple filters/kernels can detect multiple hidden features.

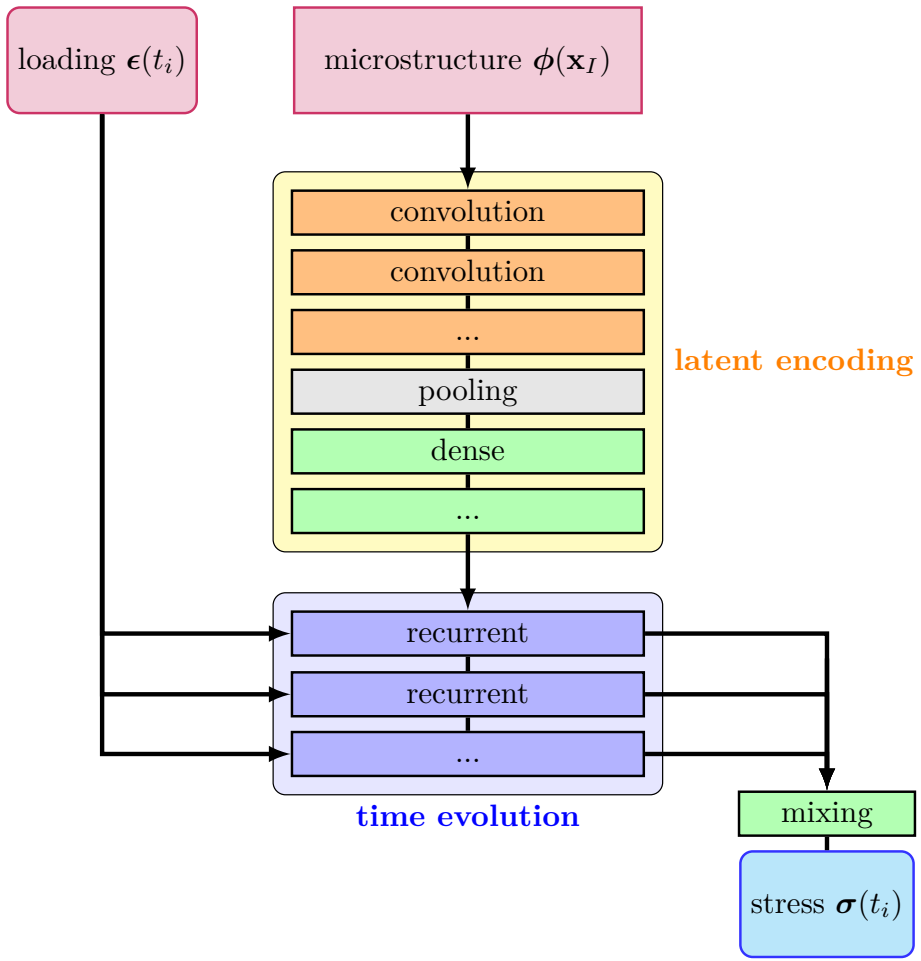
# A hybrid CNN-RNN network for time evolution

To predict a static property we can apply a **CNN** to a microstructure image. To predict the evolution of a system average we augment the **CNN** with an **RNN/NODE** that models the **loading/time dependence** to emulate the IBVP.

The CNN encodes the image into a latent space correlated with the QoI: an evolving system average. It also implicitly handles aspects related to the governing PDE, e.g. spatial derivatives.

A RNN uses a causal time filter combine the time-dependent loading information and the hidden image features to predict the QoI.

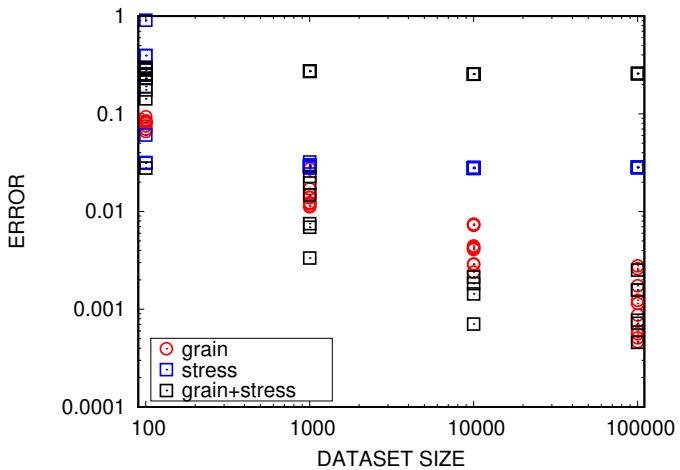
Design question:
**How many latent features should the CNN reduce the image to?**
... depends on the application.



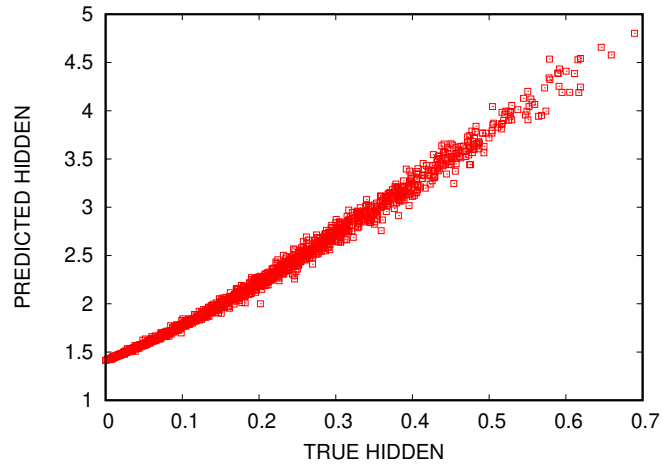
# Predicting the response due to "hidden" features

## Does the deep NN discover the hidden features?

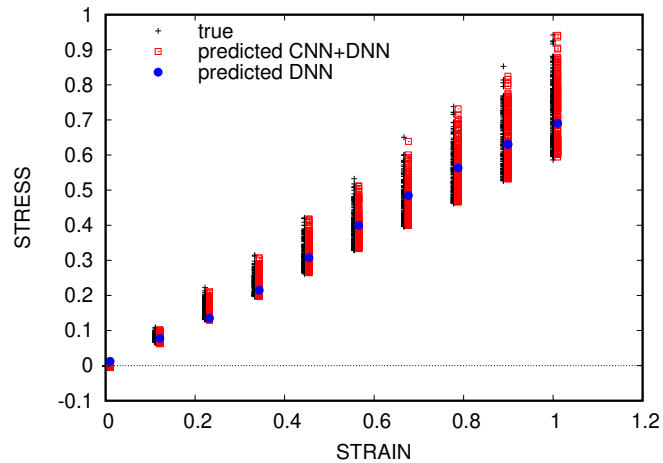
A **test problem** where we *know* what "hidden" microstructural features the observable stress depends on, e.g. average misorientation




Training NN on stress-strain alone stalls, but given initial **microstructures** continues to learn




True and learned hidden feature are **highly correlated** - but not identical

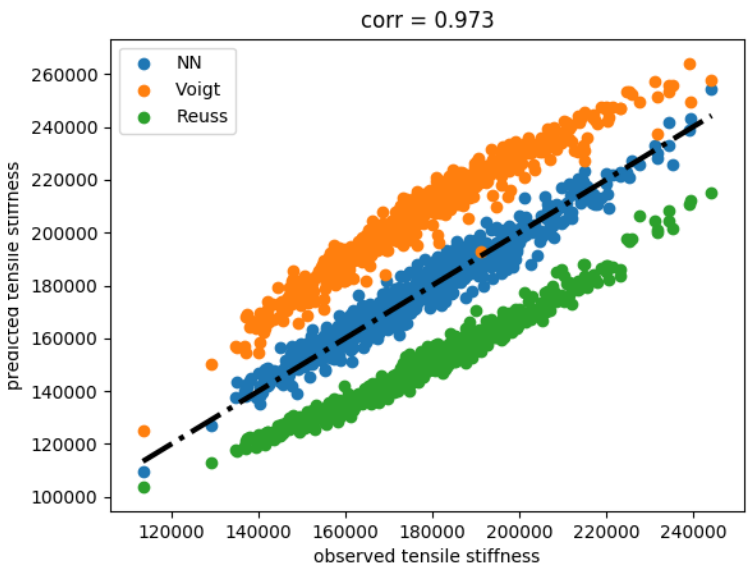



Observing the microstructure enables prediction of microstructure variations

# Predicting the particular response to microstructure

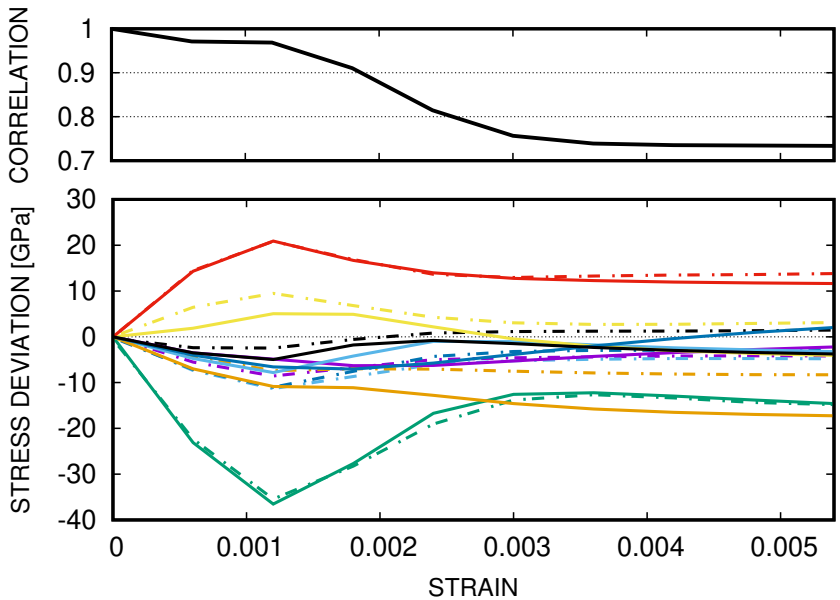
Using data from the ensemble of polycrystals, we can make predictions of the mechanical response that are **significantly better than traditional homogenization theory**.

## Close to IC




**Correlation** of elastic response (NN, Voigt and Reuss predictions), NN on par with Hill average.

## Far from IC



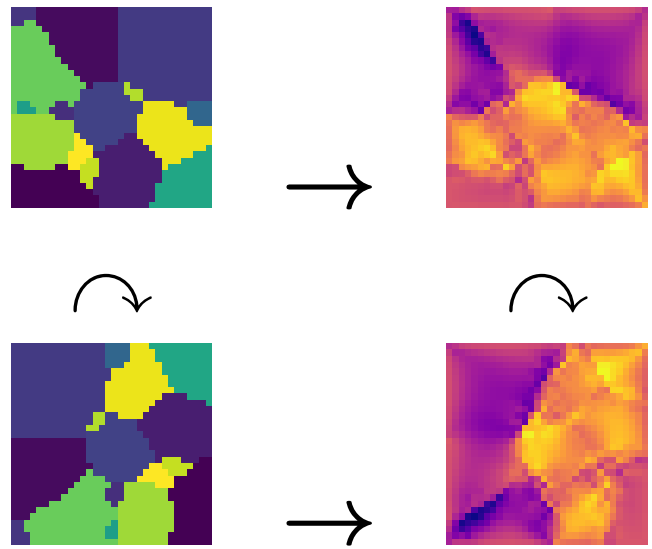

**Trajectories of discrepancy** from mean: solid lines data, dashed: NN prediction. Trajectories drift with accumulated error. Plastic response is better than Sachs or Taylor estimates.

# Physical symmetries

With the basic architecture in mind how **satisfaction of physical constraints and symmetries** which is expected in physical models and is necessary for conservation, stability, etc

**How do we learn/impose physical constraints?**

- **Augment** the dataset with many examples of what should happen, e.g. rotate the inputs and outputs (soft and inefficient)
- **Penalize** loss / training objective function (soft & introduces a meta parameter and can be hard to converge)
- **Embed** the symmetry in the NN architecture so that the response exactly preserves the symmetry (can be hard to formulate)



$$\mathbf{Q} \boxtimes \bar{\boldsymbol{\sigma}}(t, \boldsymbol{\phi}) = \frac{1}{V} \int \boldsymbol{\sigma} \left( \mathbf{Q} \boxtimes \boldsymbol{\epsilon}(t), \mathbf{Q} \boxtimes \boldsymbol{\phi}(\mathbf{X}) \right) \, \mathrm{d}\mathbf{X} \, ,$$

where $\mathbf{Q}$ is an orthogonal tensor (rotation) and $\boxtimes$ is the Kronecker product.

# Objectivity and representation theory

We prefer to **embed symmetries** in the NN structure – so that they are exact/not learned. Let's go back to classical theory [Spencer,1980s]...

*Material frame indifference* for constitutive function $\mathbf{M}(\mathbf{A})$

$$\mathbf{G}\mathbf{M}(\mathbf{A})\mathbf{G}^T = \mathbf{M}(\mathbf{G}\mathbf{A}\mathbf{G}^T) \ ,$$

$\mathbf{M}$ model must commute with the symmetry op for every member $\mathbf{G}$ of the orthogonal group.

Based on the spectral $\mathbf{A} = \sum_{i=1}^{3} \lambda_i \mathbf{a}_i \otimes \mathbf{a}_i$ , and Cayley-Hamilton theorems

$$\mathbf{A}^3 - \mathrm{tr}(\mathbf{A})\mathbf{A}^2 + \frac{1}{2}\left(\mathrm{tr}^2\,\mathbf{A} - \mathrm{tr}\,\mathbf{A}^2\right)\mathbf{A} - \det(\mathbf{A})\mathbf{I} = \mathbf{0}$$

one can obtain a compact **general representation**/model form:

$$\boxed{\mathbf{M}(\mathbf{A}) = c_0(\mathcal{I})\mathbf{I} + c_1(\mathcal{I})\mathbf{A} + c_2(\mathcal{I})\mathbf{A}^2 = \sum_i c_i(\mathcal{I})\mathbf{A}^i}$$

in form of **unknown** coefficient functions of invariants and a **known** tensor basis. Inputs: scalar invariants $\mathcal{I}$ & tensor basis $\mathcal{B} = \{\mathbf{A}^0, \mathbf{A}^1, \mathbf{A}^2\}$.

# A tensor basis neural network

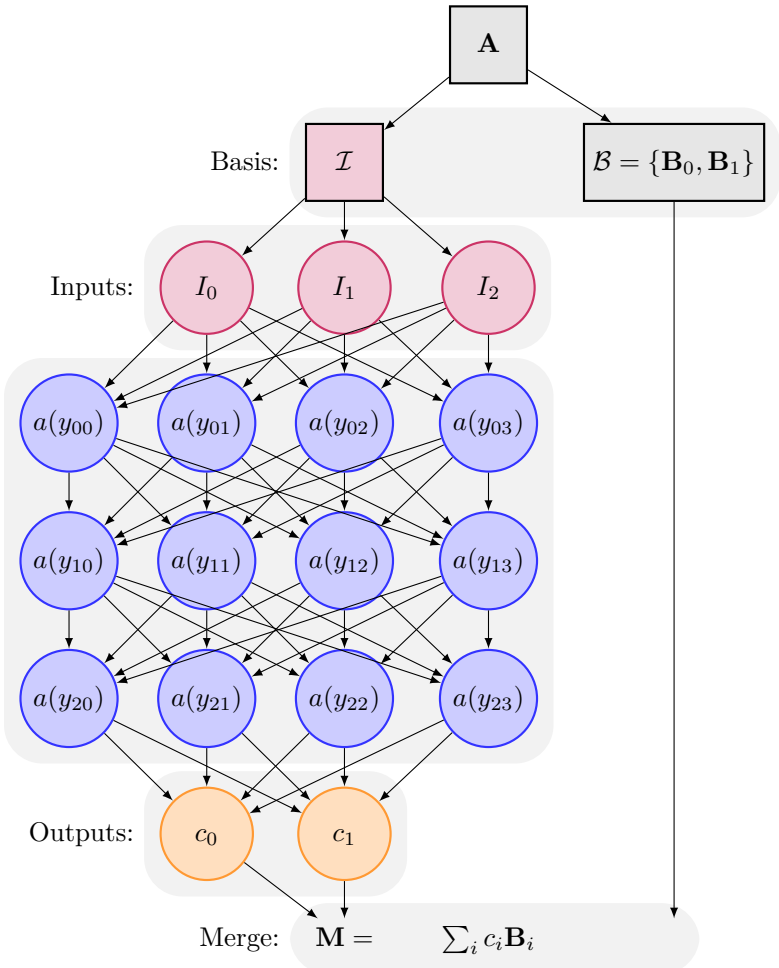
A **tensor basis neural network** is an NN implementation of this representation [Ling *JCP* 2016]: where the coefficients are unknown scalar functions of the invariants $\mathcal{I} = \{I_0, I_1, \ldots\}$

$$\mathbf{M} = \sum_i c_i(\mathcal{I})\mathbf{B}_i$$

and a final merge/sum layer associates $c_i$ with the **tensor basis** $\mathcal{B} = \{\mathbf{A}^0, \mathbf{A}^1, \ldots\}$.

**Effectively a MLP mapping invariants to coefficients + a sum with a known basis.**

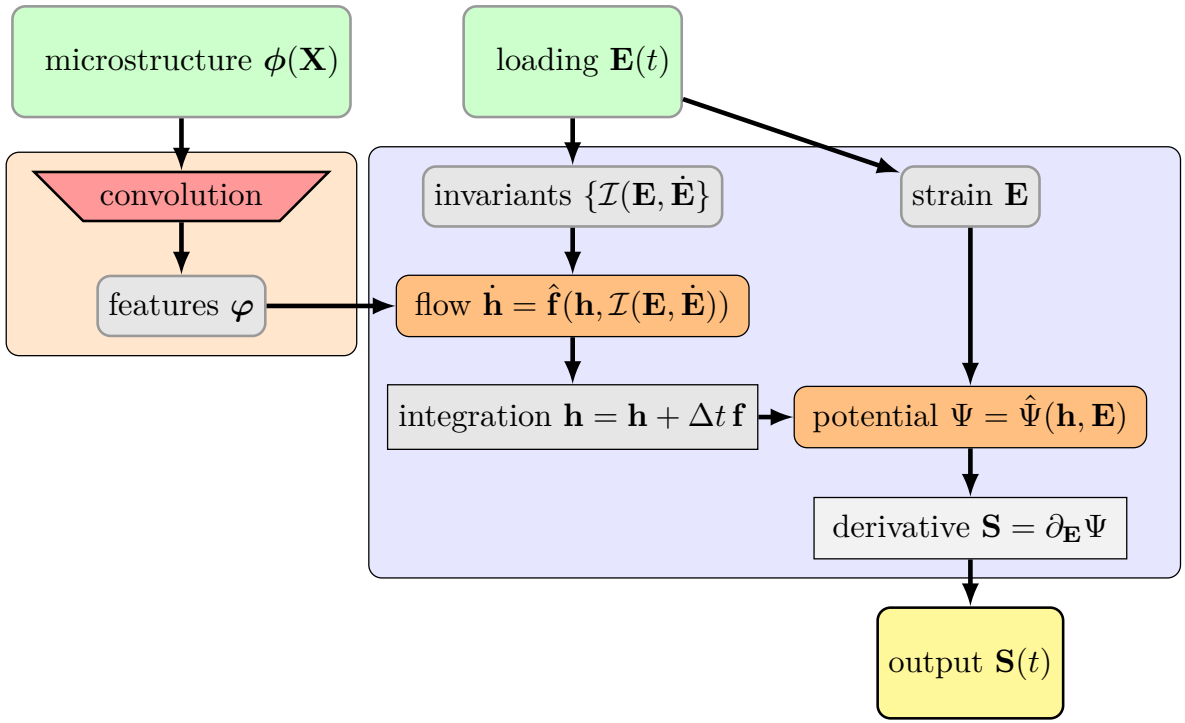
It is adept at representing the response with exact invariance / avoiding the need for data augmentation for symmetry.




A TBNN looks like a component based NN albeit with a basis constructed from the input.

# An internal state variable neural ODE model

**Premise:** it is better to **infer** internal state variables, like damage, than **prescribe** them *a priori*. So we augment the observable state with hidden states that are learned. Rivlin 1950's




Stress

$$\mathbf{S} = \mathrm{NN}_{\mathbf{S}}(\mathbf{h}, \mathbf{E})$$

Flow

$$\dot{\mathbf{h}} = \mathrm{NN}_{\mathbf{h}}(\mathbf{h}, \mathbf{E})$$

The hidden states/latent space can be augmented by microstructural information.
RNN are locked into a particular time step. NODE have the time scaling of the dynamical models & employ the same time integrators.

# Model variants and accuracy

There are multiple ways of formulating a general stress response:

- potential, as in thermodynamics

$$\mathbf{S} = \partial_{\mathbf{E}} \mathrm{NN}(\mathbf{E}, \mathbf{h})$$

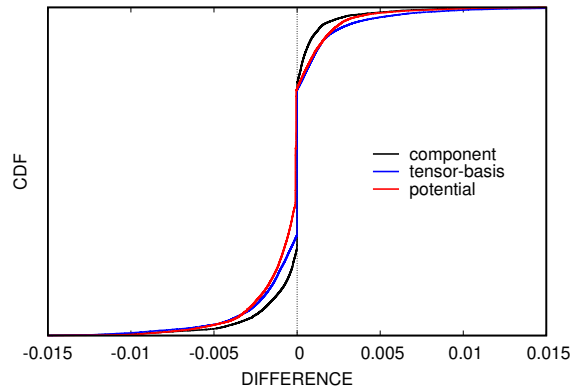
- equivariant tensor basis

$$\mathbf{S} = \sum_i \mathrm{NN}_i(\mathbf{E}, \mathbf{h}) \mathbf{B}_i(\mathbf{E})$$

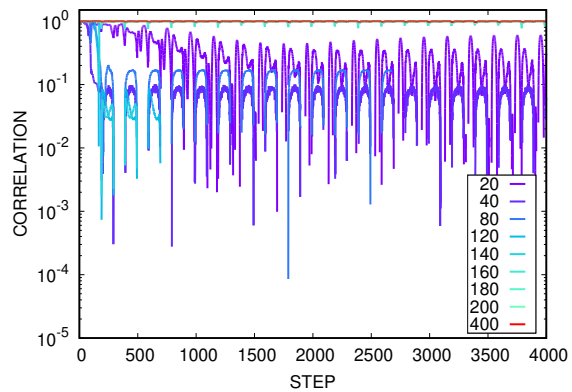
- components of a fixed basis

$$\mathbf{S} = \sum_i \mathrm{NN}_{(ij)}(\mathbf{E}, \mathbf{h}) \mathbf{B}_{(ij)}$$

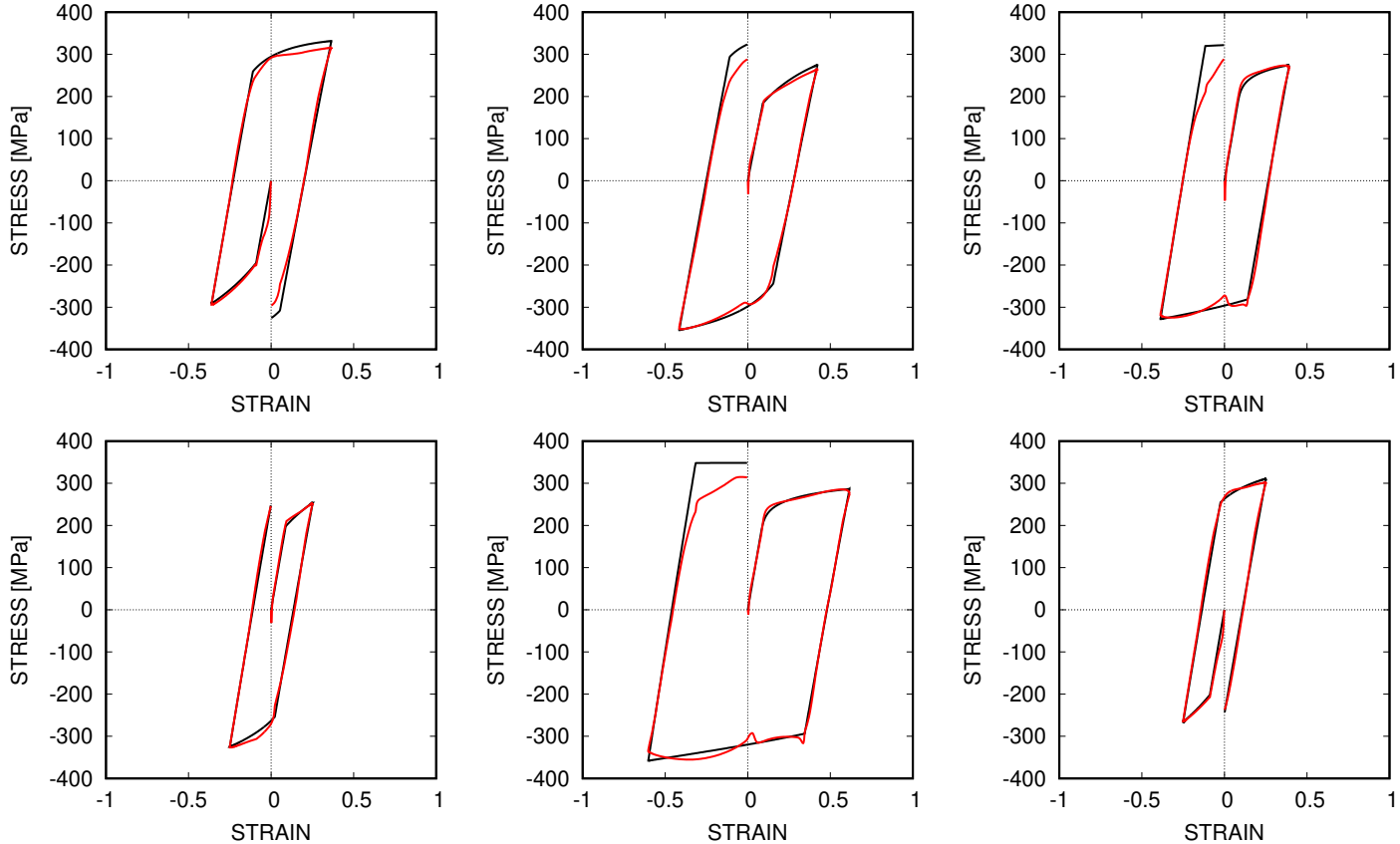
CDF of errors for TB, potential, component



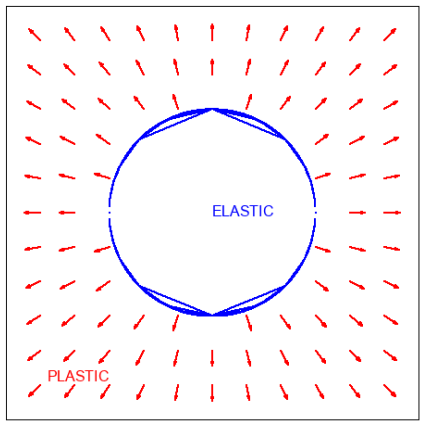

Time extrapolation with sequential training



# Elastoplasticity

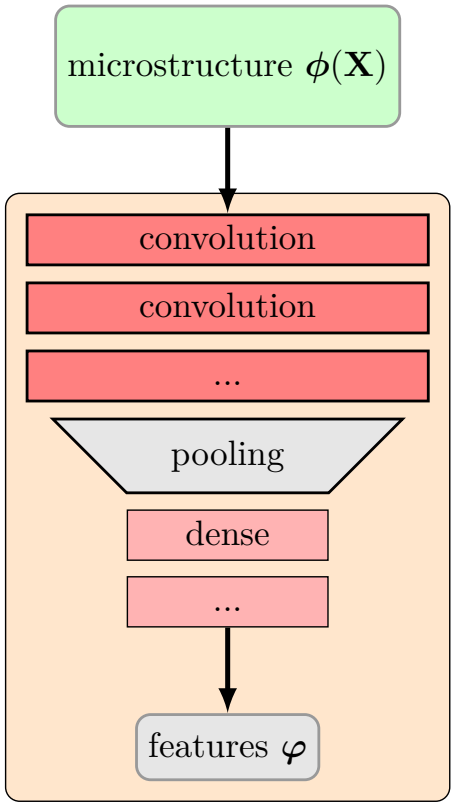



Even without an explicit yield surface, the NODE seems to learn the non-smooth flow field.



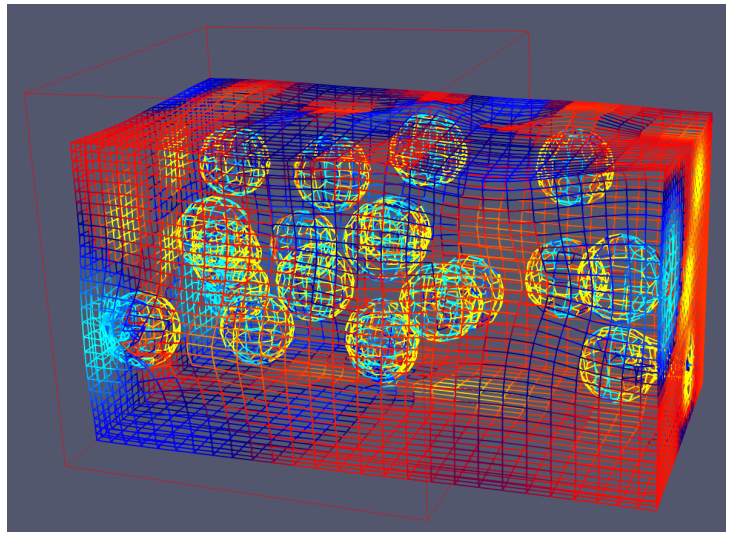
# Graphs for microstructure + ODEs for evolution

We can combine a NODE & a Graph CNN to reduce the initial microstructures to latent features

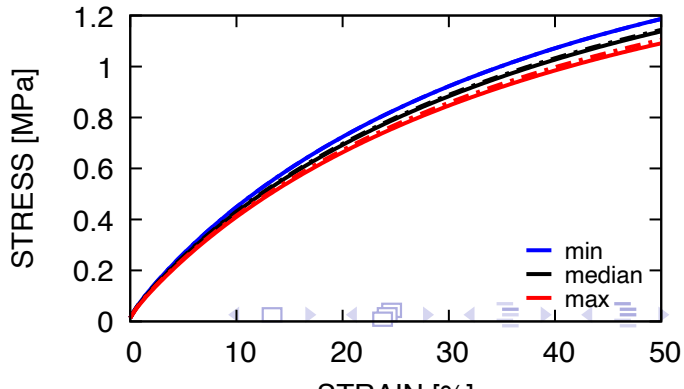



that become additional hidden state variables in the NODE flow evolution.

Microstructures with pores or hard inclusions

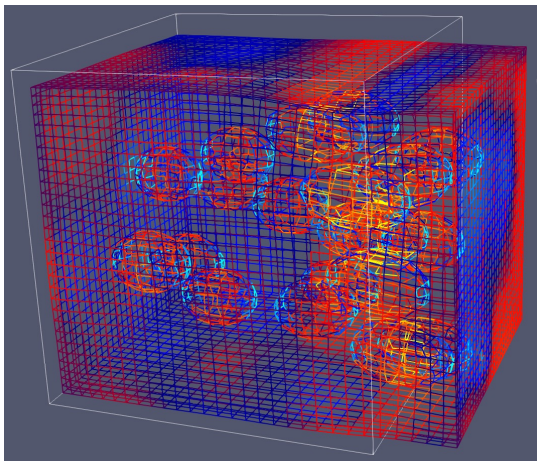


Predictions vs. truth for min, mean, max error



# Mesh data & graph-based convolutional neural networks

If we have pixelated images as inputs, CNNs work for structured grid/rastered image but the **need interpolation** for mesh-based fields and do not inherently **satisfy invariance** $\mathbf{G}\boldsymbol{\sigma}(\boldsymbol{\epsilon}, \boldsymbol{\phi})\mathbf{G}^T = \boldsymbol{\sigma}(\mathbf{G}\boldsymbol{\epsilon}\mathbf{G}^T, \mathbf{G}\boldsymbol{\phi}\mathbf{G}^T)$ where $\boldsymbol{\phi}$ is the initial microstructure.
In a graph based representation the data is essentially **lifted from the spatial domain**, that together with the fact that filters are permutationally invariant leads to a level of invariance in the overall network.

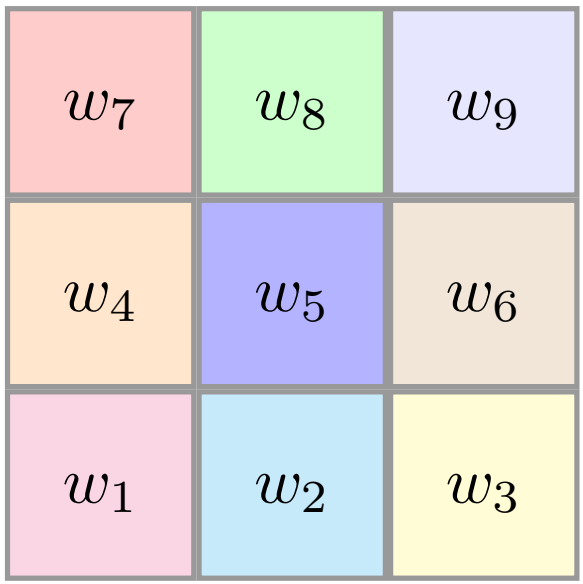


Reducing the grains to nodes and shared interfaces to edges has been shown effective. However this approach loses information (eg the details of the grain and interface geometry) and hence **requires featurization**.

We have applied graph convolutions **directly to the mesh topology**. This approach does not require featurization but can benefit from it. It does **not increase the number of parameters** since the same kernels are being employed.

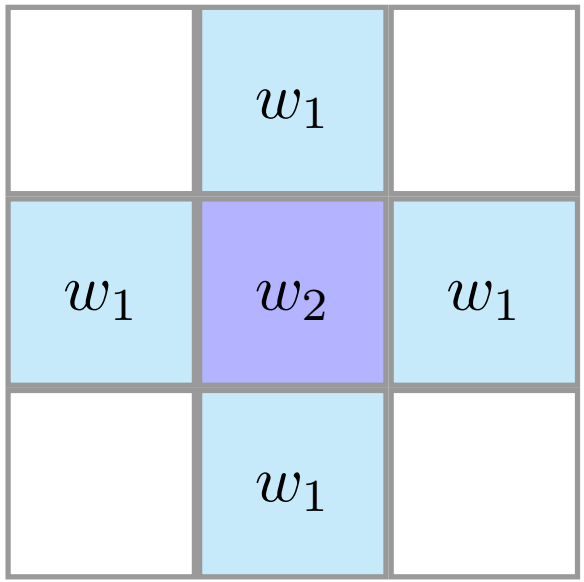
# Graph-based convolutional neural networks

In contrast to pixel-based filters which have a sense of up-down-left-right, a graph based filter only knows what are its neighbors and hence treats each as equivalent i.e. **spatial adjacency is traded for a neighbor-wise adjacency.**




CNN filter

$x = a(Wx + b)$



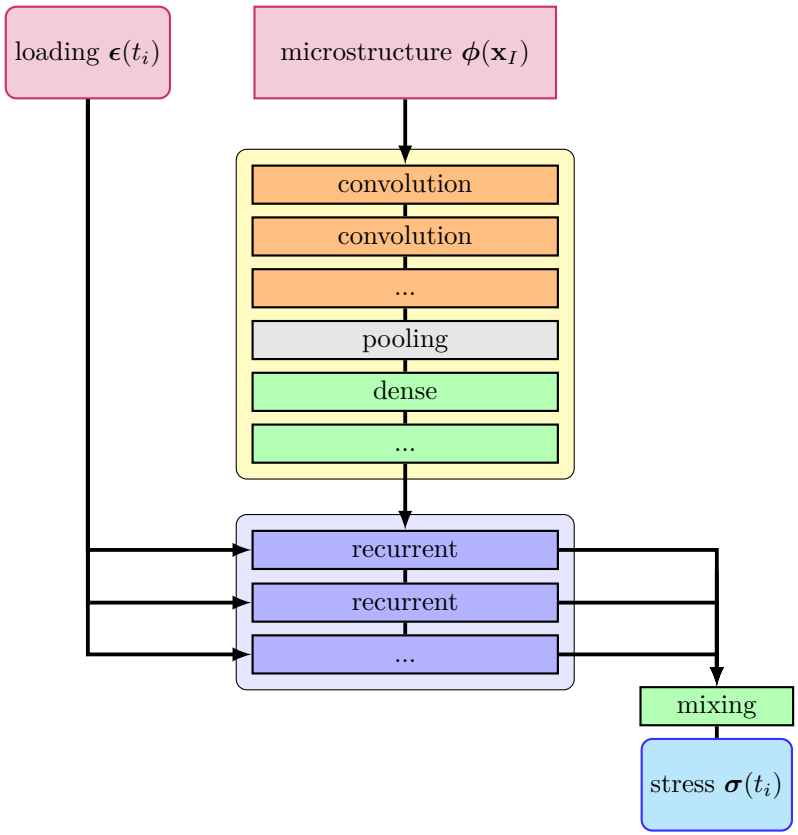

GCNN filter

$x = a([\sum_i w_i A_i]x + b)$

The GCNN filter uses the same weights for all the neighbors (permutational invariance) defined by adjacencies $A_i$, hence it produces the same output when the image is rotated.

# GCNNs vs. CNNs

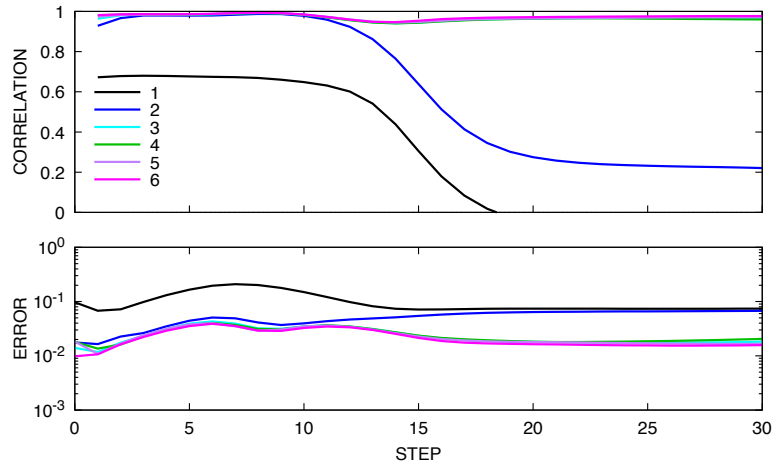
GCNNs have similar performance to CNN with fewer parameters and inherent invariance.



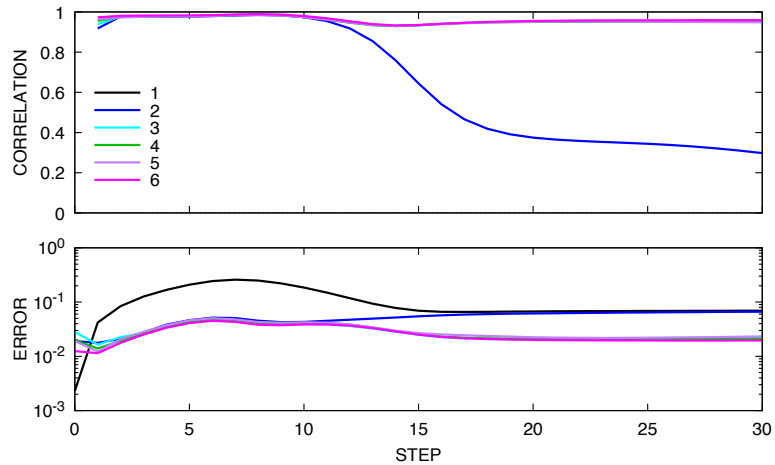

**GCNN-RNN**

filters $\sim$ features

*Convergence with number of filters*



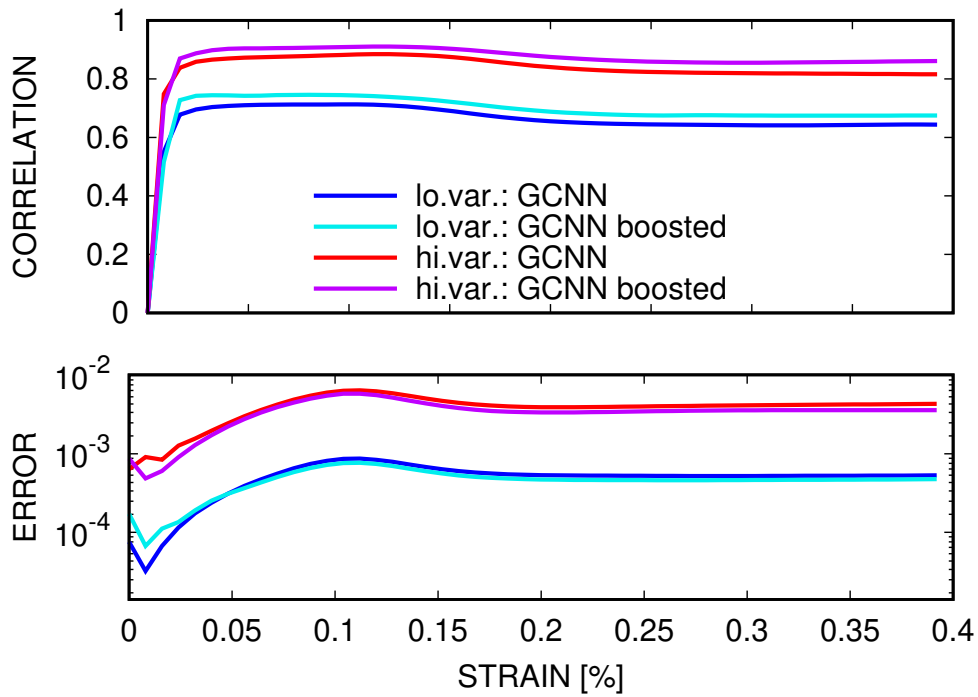

**CNN**




**GCNN**

# Feature boosting

GCNNs (and CNNs) can be **boosted** by **embedding** obvious **features** into the image (or further down the CNN-RNN pipeline)

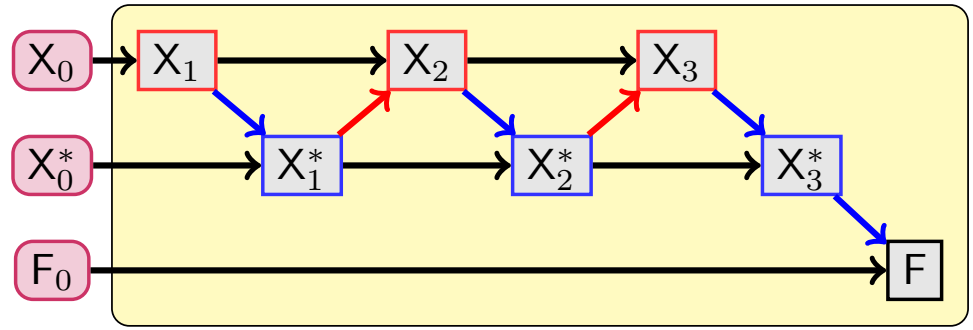


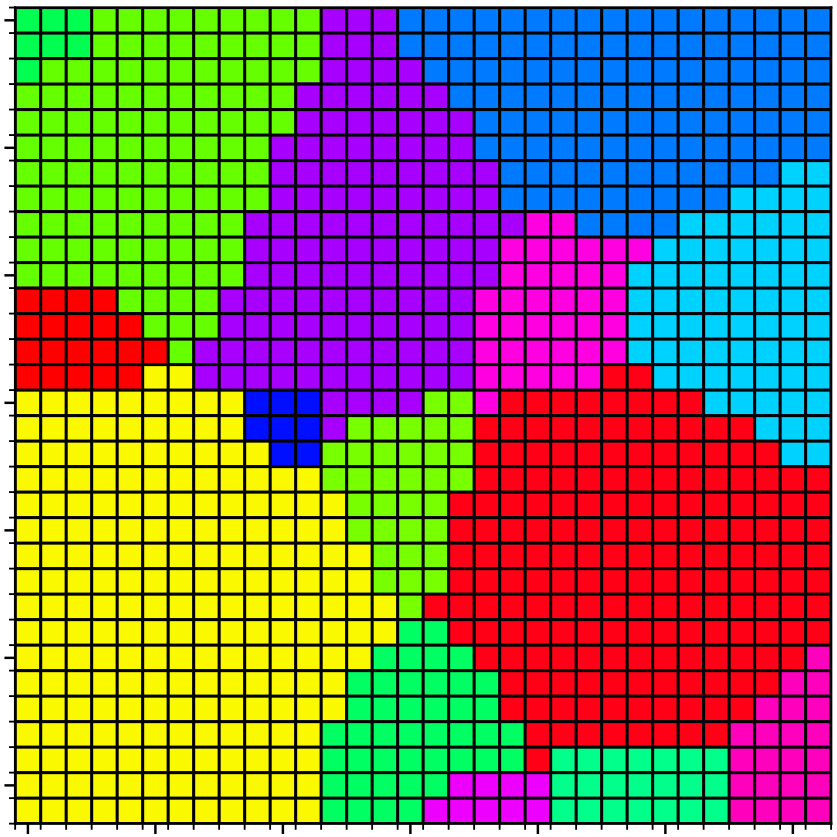

*adding node volumes to image of orientation angles*

The improvement is marginal but distinct for a NN that is already fairly accurate.

# Multilevel graphs for microstructure

- discretization (mesh),
- clusters (data)
- sample (global).



Mesh adjacency:

$$A_{ij} = \begin{cases} 1 & \text{if nodes } i,j \text{ are neighbors} \\ 0 & \text{else} \end{cases}$$

Cluster assignment matrix:

$$S_{Ki} = \begin{cases} 1 & \text{if node } i \text{ is in cluster } \Omega_K \\ 0 & \text{else} \end{cases}$$

Reduction: $\mathrm{V}^* = \mathrm{SV}$

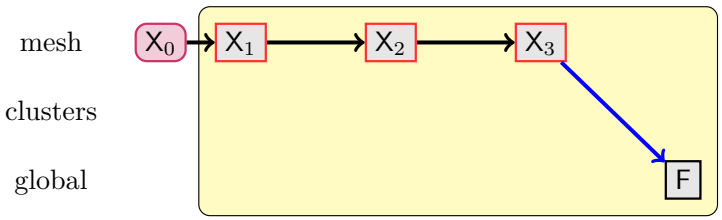
Prolongation: $\phi = \mathrm{S}^T \phi^*$

**Convolution**: $\mathrm{X} = \mathrm{Conv}(\mathrm{X}, \mathrm{A})$

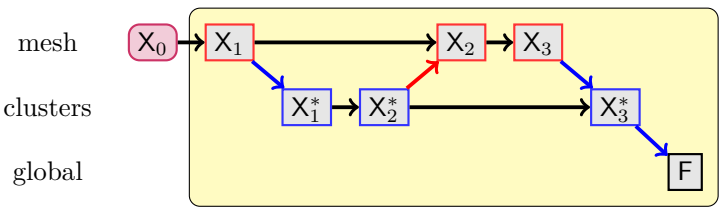
# Multilevel graphs for microstructure

Goals: deep featurization and accomodate multiscale interactions.

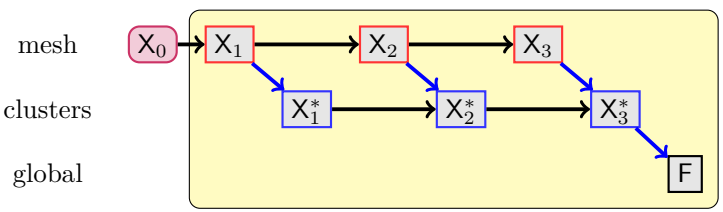
Full



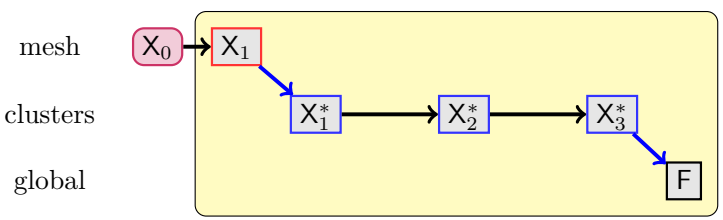

Vee



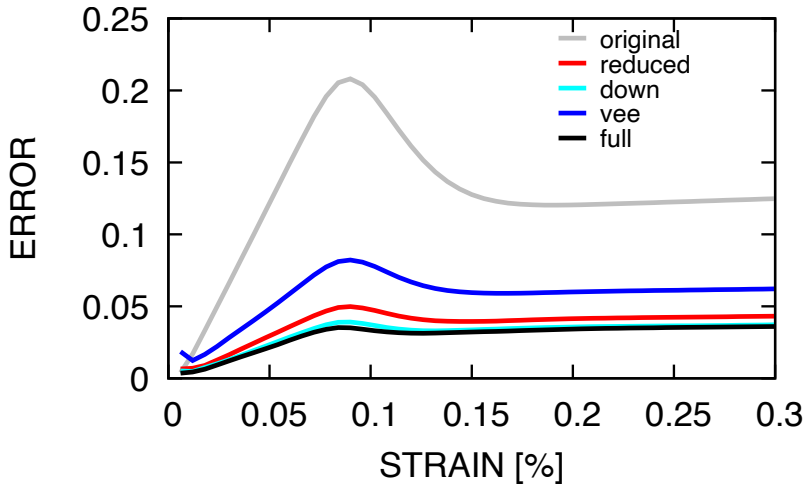

Down



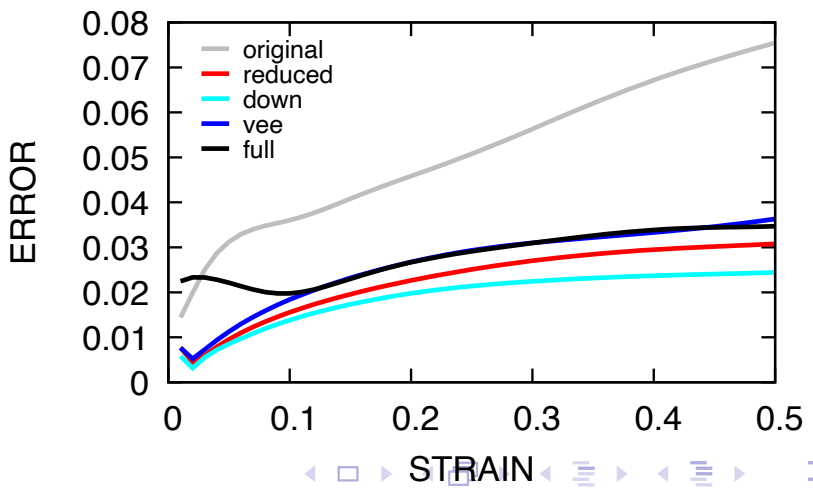

Reduced




Crystal plasticity




Viscoelastic composite with stiff inclusions



# Full field predictions: convLSTM

An architecture similar to the CNN-RNN we used to predict system-level evolution can be used to predict **full-field** (element/pixel level) evolution.

**Inputs**: pairs of

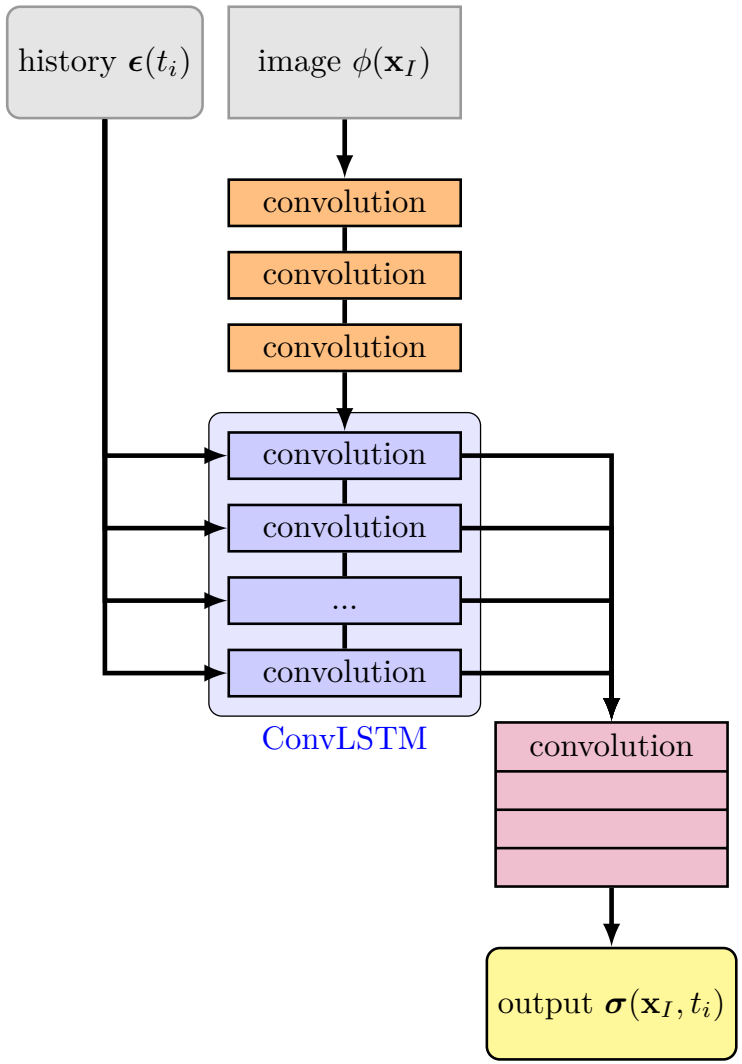
- $\phi(\mathbf{X})$: image of initial microstructure
- $\epsilon(t)$: system level strain history

The image is fed to a *convolutional* neural network to process its latent features but not reduce them to a list of scalars – **each layer/filter output is also an image** so that spatial relationships are preserved.

This initial condition-like input is combined with the strain history in a *recurrent-convolutional* neural network, a convLSTM [Shi *NIPS* 2017].

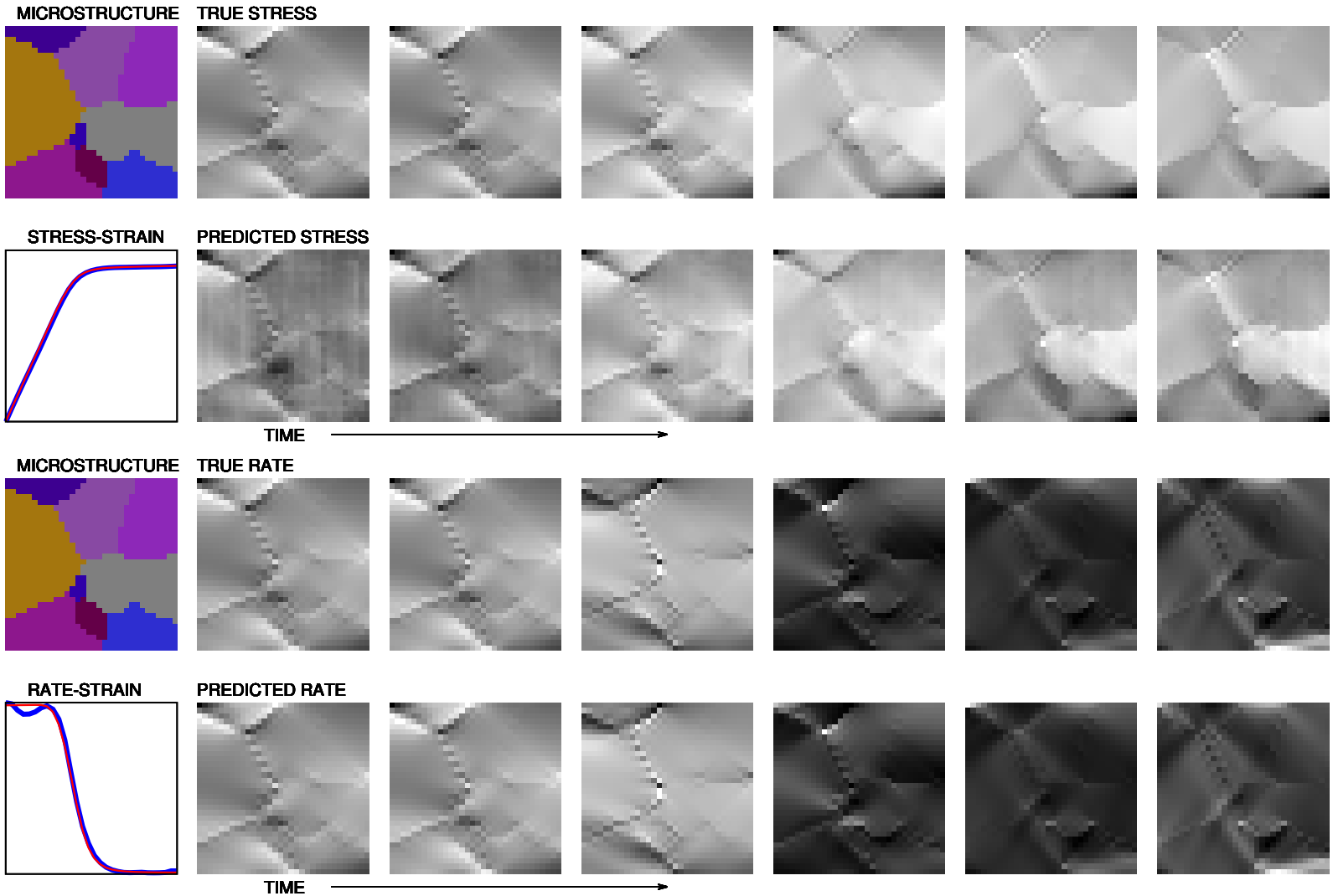
The output of the convLSTM is processed by another CNN unit to produce

**Output**: $\boldsymbol{\sigma}(\mathbf{X}, t)$ full field stress evolution



# Full field predictions

A convLSTM combines the RNN (time) and CNN (space) into PDE-like model

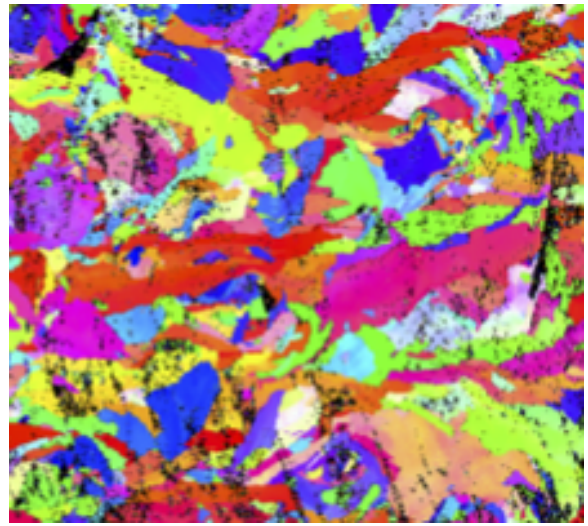



[Frankel *MLSciTech* 2020]

# Conclusion

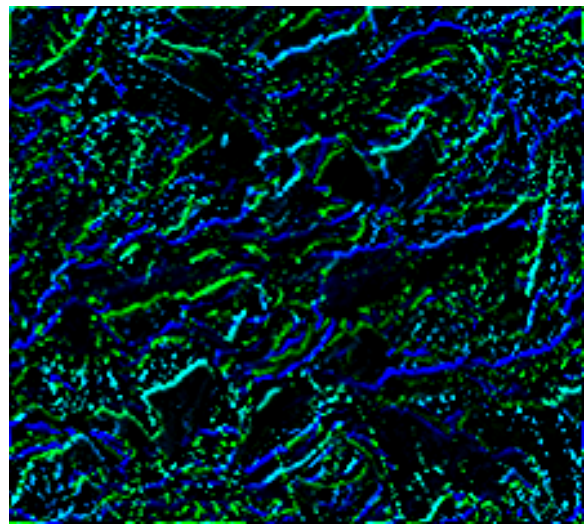
**Applications:**

- subgrid / multiscale surrogate models
- structure-property exploration / material optimization
- material uncertainty quantification



**Open issues:**

- architecture / meta parameter optimization
- interpretability ( latent space / low dimensional manifold)
- training burden / multifidelity (experimental+simulation) data



**rjones@sandia.gov**

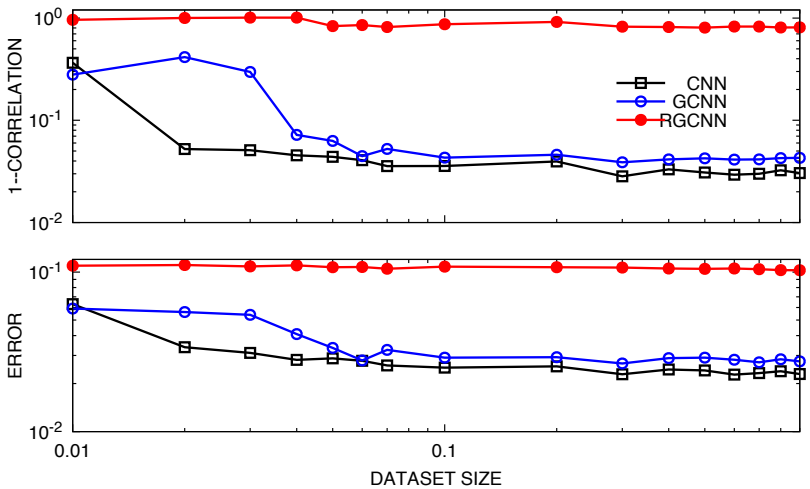
# References: **rjones@sandia.gov**


- J Ling, RE Jones, JA Templeton. *Machine learning strategies for systems with invariance properties* JCompPhys (2016).
- RE Jones, JA Templeton, CM Sanders, JT Ostien. *Machine learning models of plastic flow based on representation theory* CompModEngSci (2018)
- AL Frankel, RE Jones, C Alleman, JA Templeton *Predicting the mechanical response of oligocrystals with deep learning* CompMatSci (2019)
- AL Frankel, K Tachida, RE Jones *Prediction of the evolution of the stress field of polycrystals undergoing elastic-plastic deformation with a hybrid neural network model* MachLearn:SciTech, (2020)
- AL Frankel, RE Jones, L Swiler *Tensor Basis Gaussian Process Models of Hyperelastic Materials* JMachLearnModComp (2020)
- AL Frankel, C Safta, C Alleman, RE Jones, *Mesh-based graph convolutional neural network models of processes with complex initial states* JMachLearnModComp, (2021)
- RE Jones, AL Frankel, KL Johnson *A neural ordinary differential equation framework for modeling inelastic stress response via internal state variables*, JMachLearnModComp (2021)
- W Bridgman, X Zhang, G Teichert, M Khalil, K Garikipati, RE Jones, *A heteroencoder architecture for prediction of failure locations in porous metals using variational inference*, CMAME (2022)
- JN Fuhg, N Bouklas, RE Jones. Learning hyperelastic anisotropy from data via a tensor basis neural network. JMechPhysSol (2022).

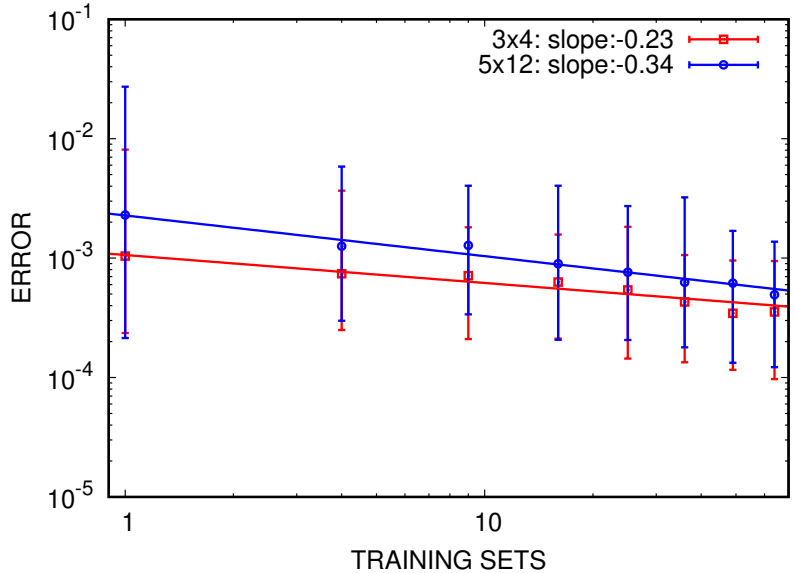

... new work on graph reduction forthcoming on arXiv.

# Training data: sampling & convergence

**Sampling** over loading modes, microstructures, etc to obtain **sufficient data is expensive** for a reasonably complex/expressive NN.




Error vs (microstructure) sample size for a fixed size test set




Error vs random sampling of modes for homogeneous material

With respect to sample size: steep decrease until number of samples $\approx$ number of parameters, then slow improvement. Similar slow improvement with mode sampling.