Machine learning material models for multiscale topology optimization

Presented to the MOM group

Charles F. Jekel and Seth E. Watts

Work supported by the LLNL-LDRD Program under Project No. 20-ERD-020

April 13, 2022

LLNL-PRES-835421

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

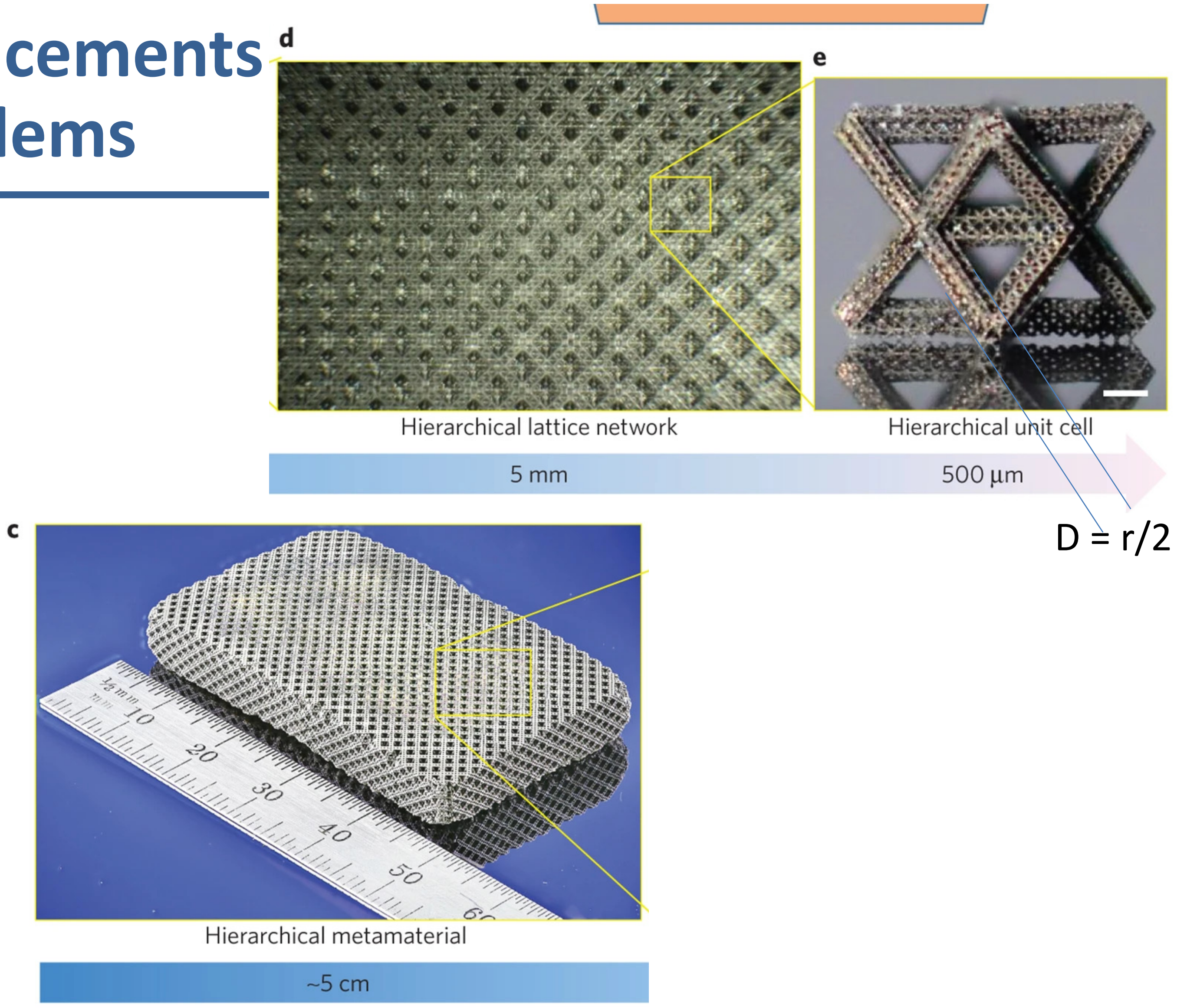
Lawrence Livermore National Laboratory

Introduction

- Work done on LDRD 20-ERD-020 Surrogates for Lattices (Seth Watts PI)
- Kenny Swartz, Dan White, and Seth Watts have been tremendously helpful
- Topics for today
  - What's been done so far:
    - Why we are doing this
    - Multiscale topology optimization
    - Homogenization databases
  - Work in progress:
    - Machine learning material models
    - C++ implementation
- Very much want feedback on work in progress!

Additive manufacturing advancements create interesting design problems

- 3D print unit cell microstructures
  - Octet truss lattice
  - Density control by changing rod radii
- Lattice of unit cells
  - 3D printed with different radii
  - Functional graded density
- Design entire structures from lattices
  - How to change all of the rod radii to give optimal structures?
  - Design possibilities exceed our intuition
  - 100x50x20 = 100,000 different unit cells



Figured modified from Zheng et al. [1].

Density based topology optimization to find optimal designs

- Topology optimization has been a way to design complex structures that exceed our intuition
- Classical Finite Element Method
  - Linear static loads
  - Small strains
- **Givens:** domain and boundary condition
- **Problem:** find the best placement of material
- **Goal:** minimize structural compliance
  - Subject to a mass constraint
- Each finite element is given a density ρ_i
  - Between zero (no material) and one (full material)

$$\min_{\boldsymbol{\rho}} \boldsymbol{f}^{\mathrm{T}} \boldsymbol{u}$$

$$\text{subject to}$$

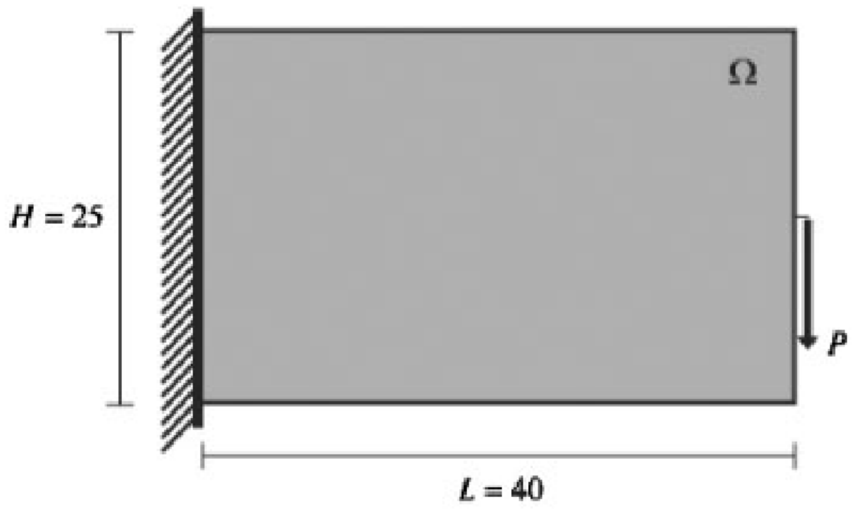
$$\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f}$$

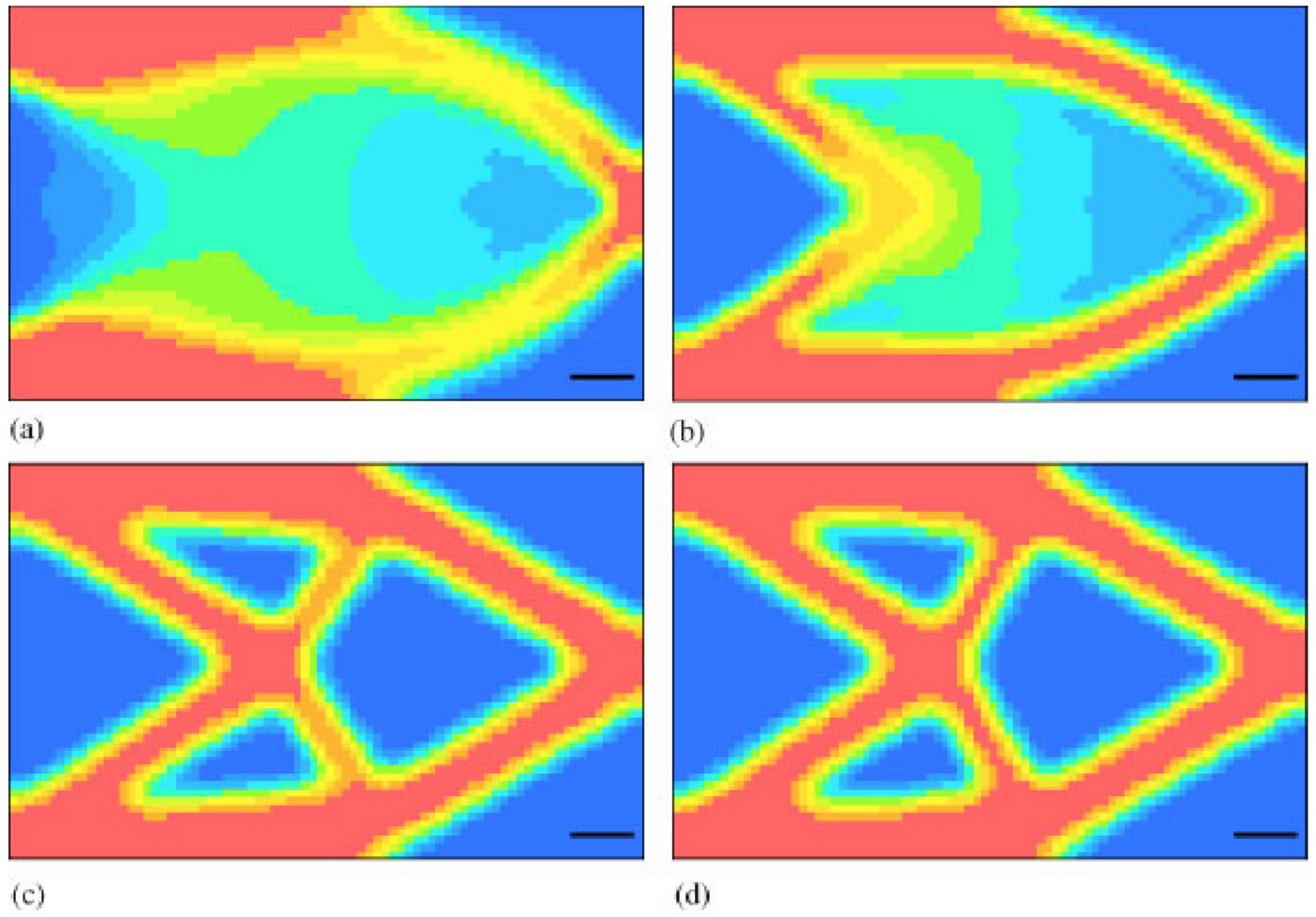
$$\sum_i^n \rho_i v_i \leq m$$

$$0 < \rho_i \leq 1$$

Topology optimization example on cantilever beam

With a limited amount of material, find the best cantilever beam that minimizes compliance.





Figures from Guest et al. [2].

Red represents density of 1, blue represents density of 0. Design evolution using continuation method.

Design of lattice structures via multiscale topology optimization

- Just like the previous topology optimization problem
- **Givens:** domain and boundary condition
- **Problem:** find the best distribution of **rod radii**
- **Goal:** minimize structural compliance
  - Subject to a mass constraint
- Each finite element has a fixed rod radius r_i
  - Lower and upper bound on the rod size
  - Based on what is physically possible to 3D print
- Optimally designed structure that can be directly created with additive manufacturing!

$$\min_{\boldsymbol{r}} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{u}$$

$$\text{subject to}$$

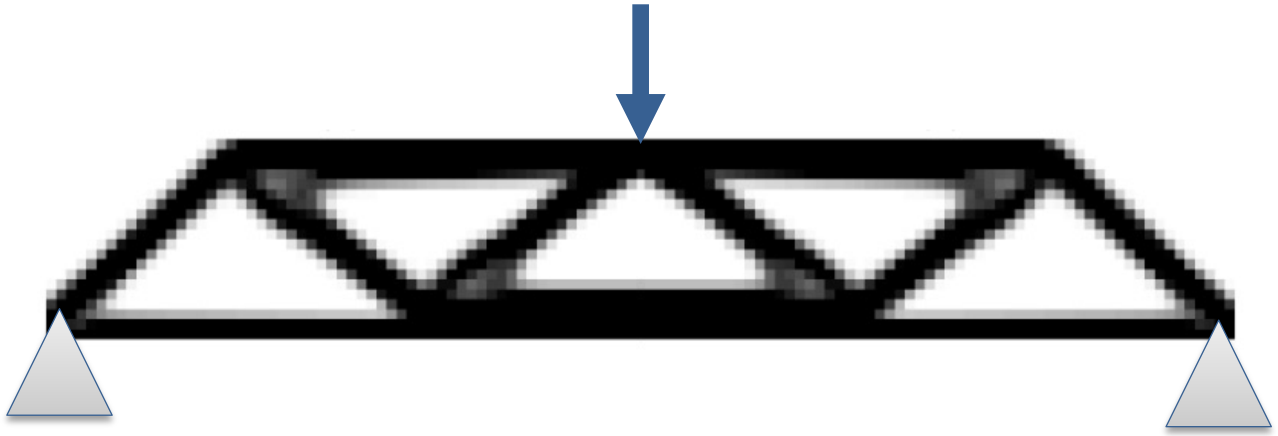
$$\boldsymbol{K}(\boldsymbol{r})\boldsymbol{u} = \boldsymbol{f}$$

$$\sum_{i}^{n} \hat{\rho}(r_i)v_i \leq m$$

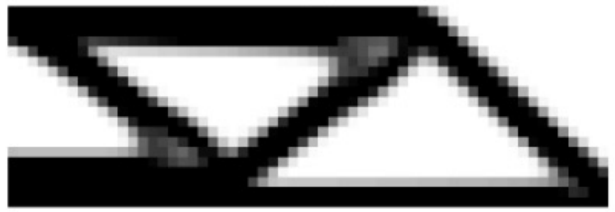
$$r_{lb} < r_i \leq r_{ub}$$

Comparison of topology optimization with multiscale design

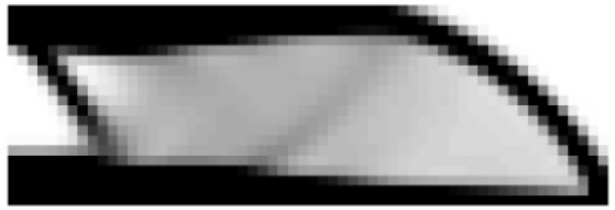
- Black is fully dense finite element cell
- White is no material
- For the truss lattices
  - Black being largest rod radius
  - White no material (or smallest radius)
  - Grey is some radius in between



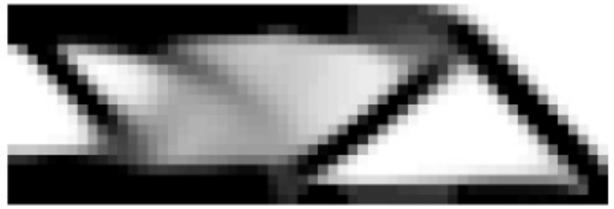
Classical topology optimization of MBB beam (with domain symmetry)



Octet truss beam

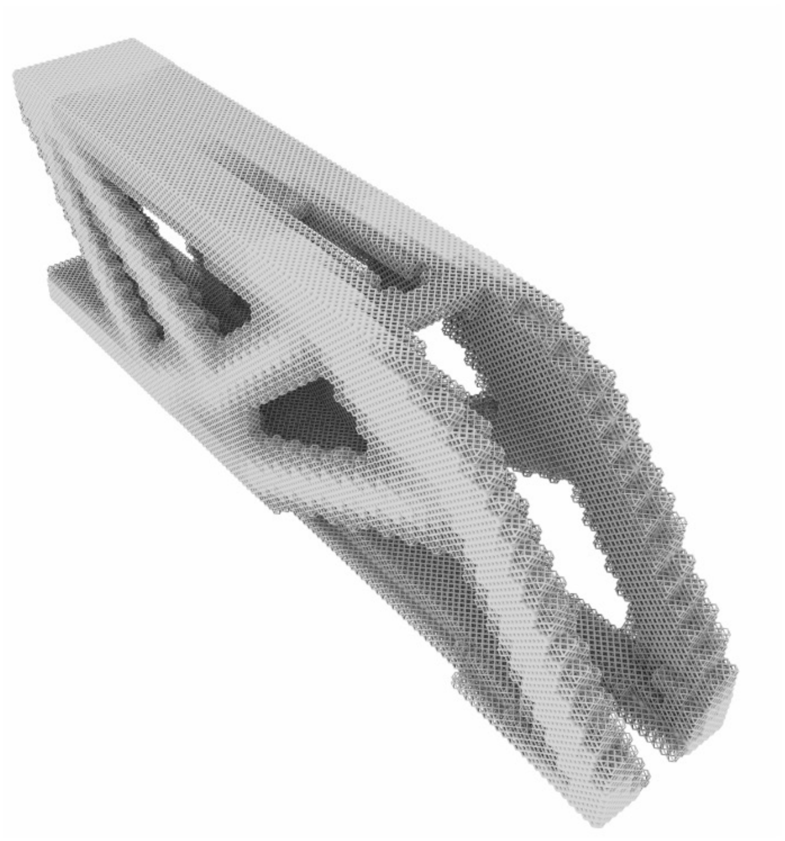


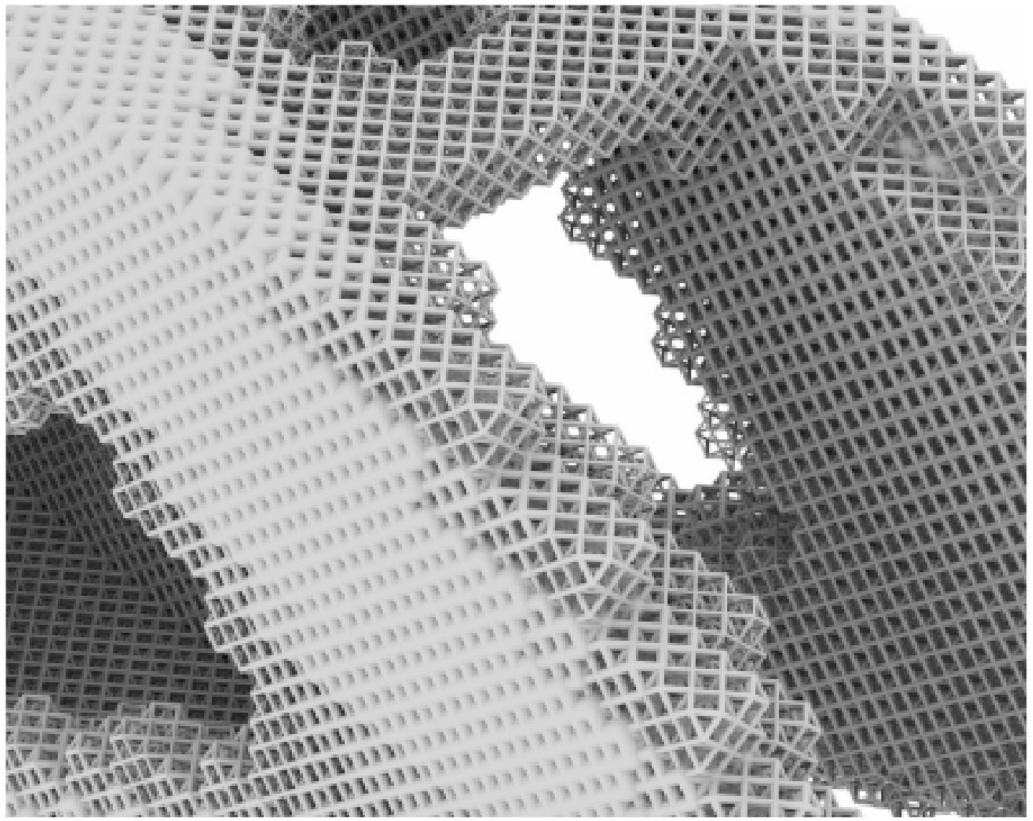
ORC truss beam



Figures from Watts et al. [3]

Multiscale topology optimization result in 3D





Optimal 3D MBB beam made of ORC truss microstructure. See the variation of rod radii throughout the structure. Figure from Watts et al. [3].

Machine learning model use in multiscale topology optimization

- Machine learning models are used to describe the microstructure scale
- Model of volume fraction as function of rod radius

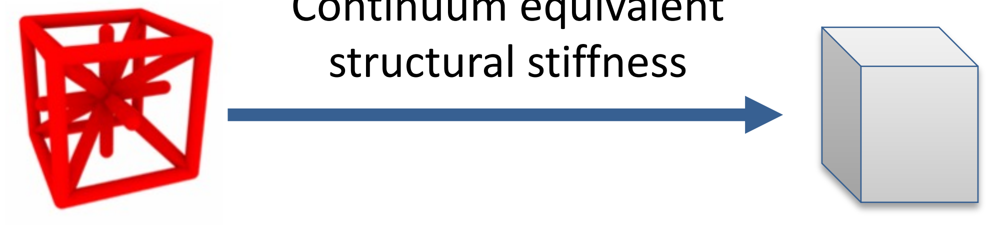
$$\hat{v}(r)$$

- Finite element stiffness as a function of rod radius
  - Use ML to predict homogenized stiffness matrix
  - As a function of the truss rod radius

$$\hat{\boldsymbol{C}}(r) = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & symm & & & C_{55} & C_{56} \\ & & & & & C_{66} \end{bmatrix}$$

Homogenized ML material models

- Performed homogenization on various truss microstructures
- Use a unit Young's modus, but specified Poisson's ratio
  - Allows us to switch between different isotropic printing materials
  - Reduces the number of unknowns by 1
- Most microstructures have an orthotropic response
- ML model learns Voigt stiffness matrix as function of geometry and Poisson's ratio

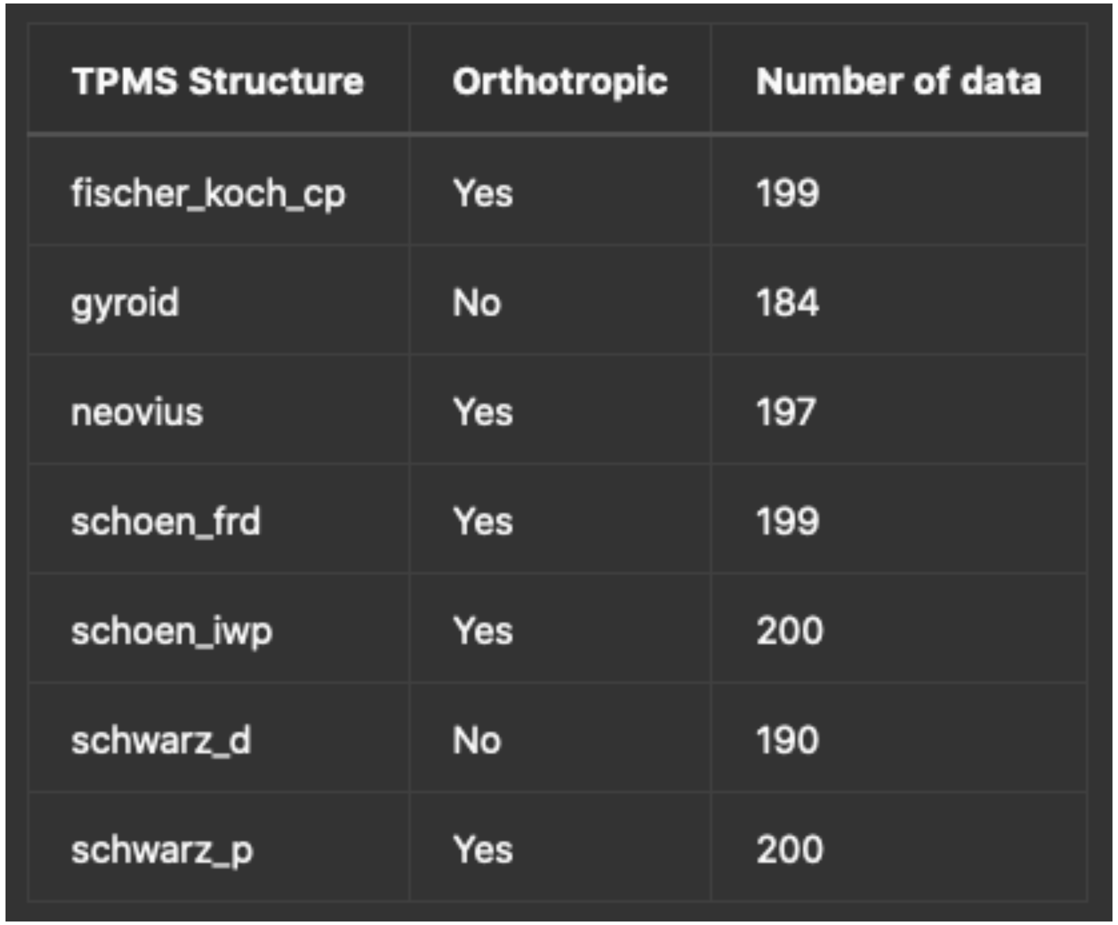


$$\hat{\boldsymbol{C}}(r,\nu) = \begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ & C_{22} & C_{23} & & & \\ & & C_{33} & & & \\ & & & C_{44} & & \\ & symm & & & C_{55} & \\ & & & & & C_{66} \end{bmatrix}$$

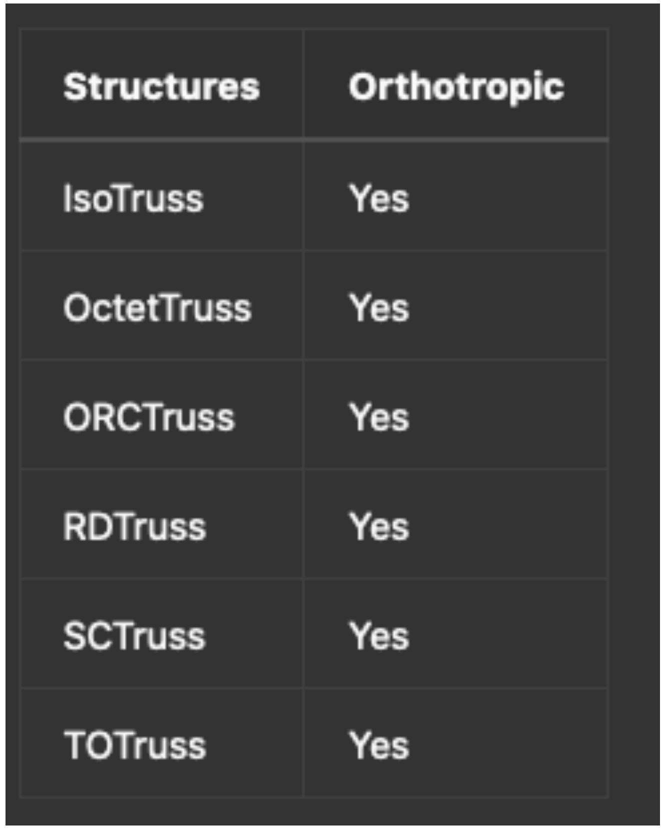
The ML material models are meant to be used in a design tool

- Assumptions: Linear static and small strain
- This is not going to give you highly resolved structural behavior
  - No impacts, no failures
  - Still need to run highly resolved simulations for these
- The design tool is meant to answer the question:
  - Given a possible brick of material
  - Where and how can we place unit cell microstructures

Homogenization databases of various structures

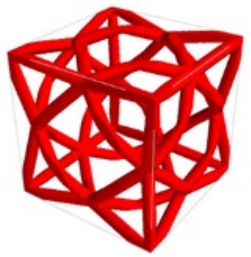
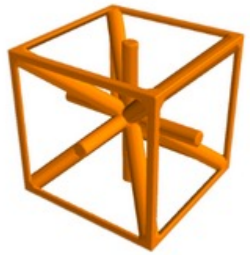
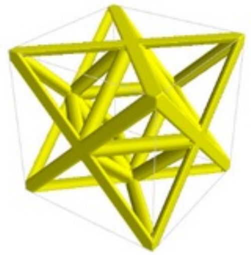
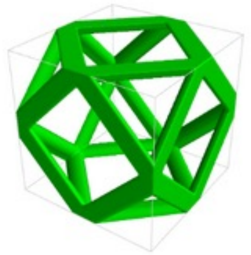
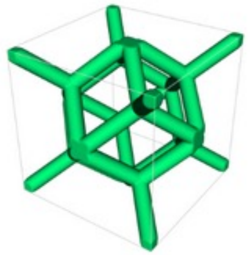
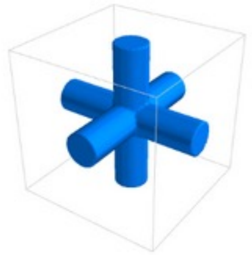
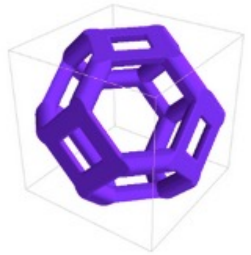
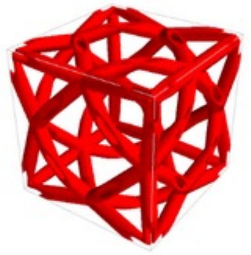
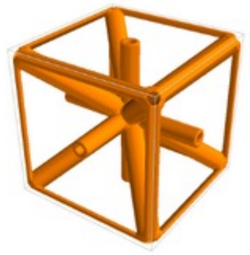
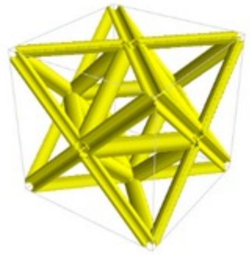
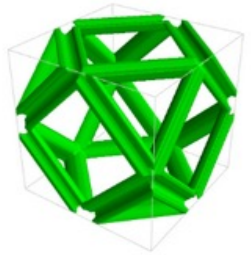
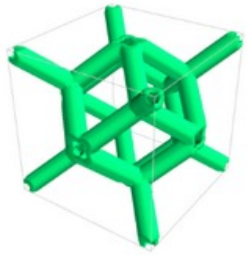
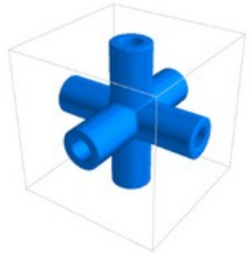
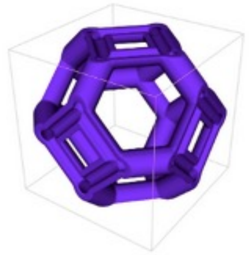


TPMS Structure	Orthotropic	Number of data
fischer_koch_cp	Yes	199
gyroid	No	184
neovius	Yes	197
schoen_frd	Yes	199
schoen_iwp	Yes	200
schwarz_d	No	190
schwarz_p	Yes	200

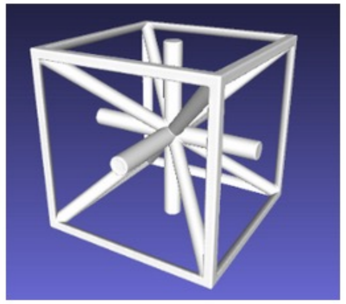
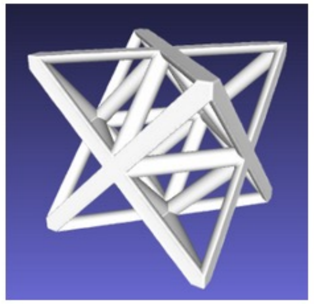
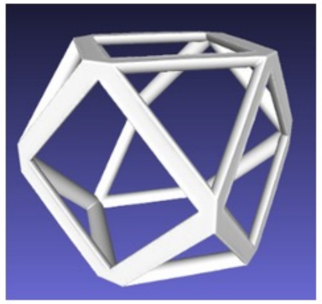
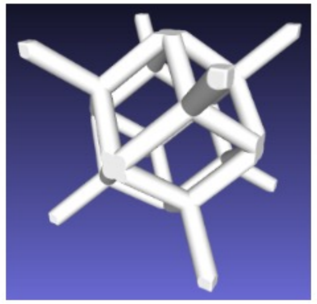
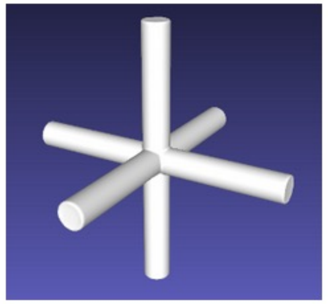
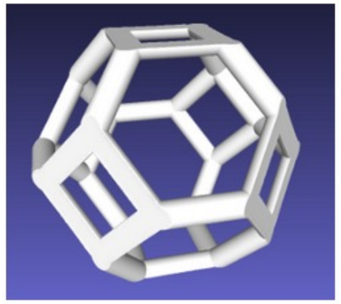
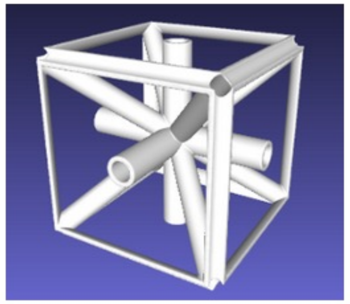
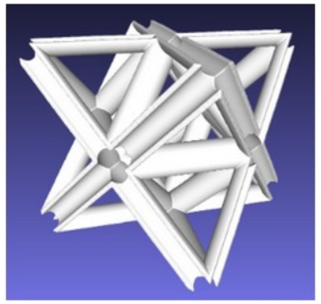
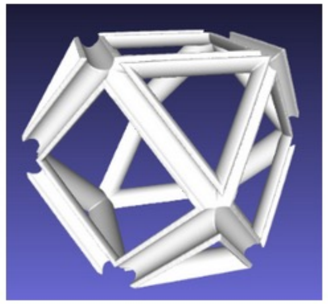
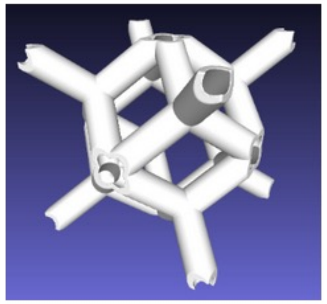
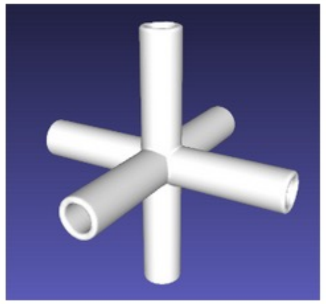
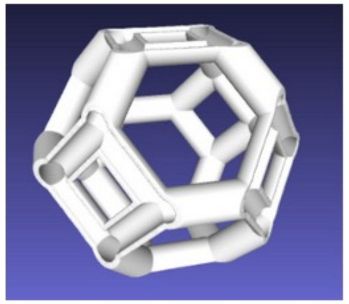
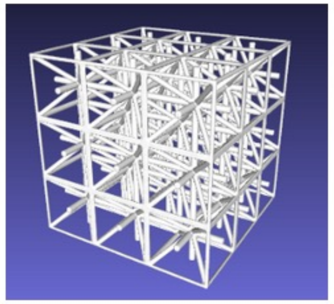
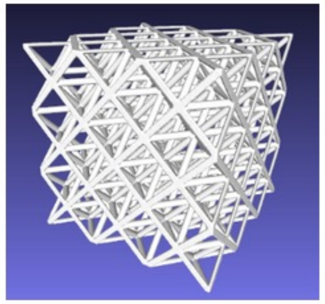
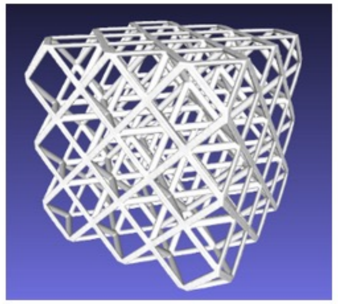
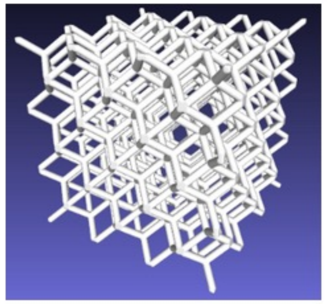
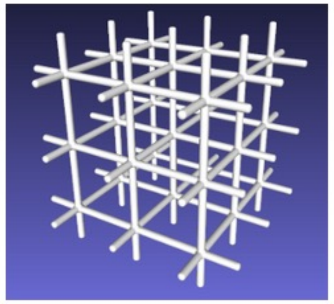
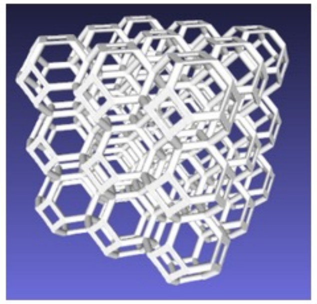


Structures	Orthotropic
IsoTruss	Yes
OctetTruss	Yes
ORCTruss	Yes
RDTruss	Yes
SCTruss	Yes
TOTruss	Yes

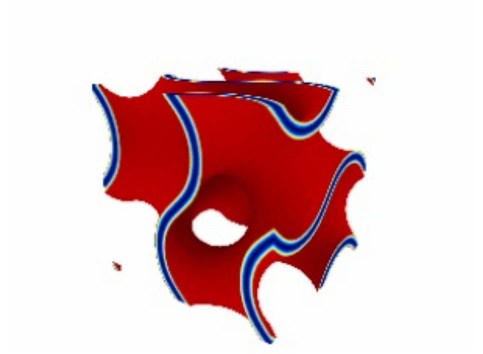
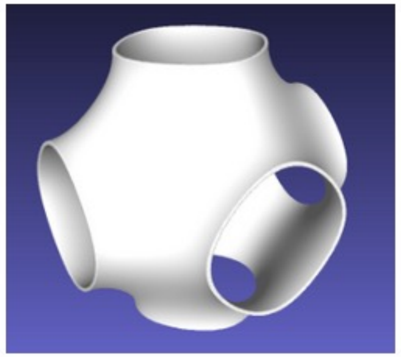
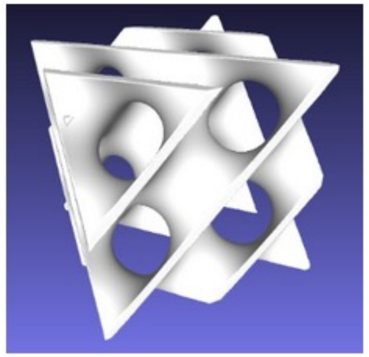
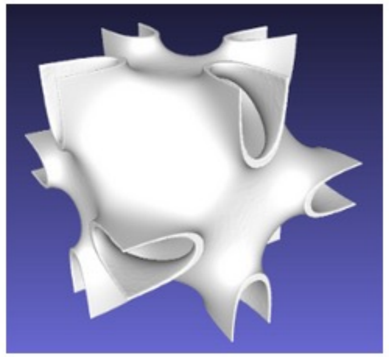
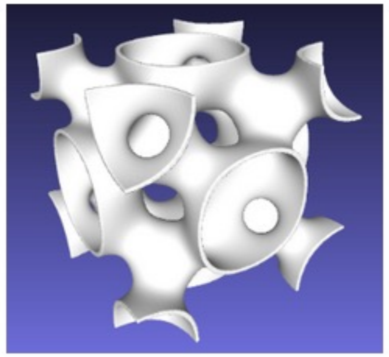
Solid and hollow truss lattices

	CORC	Iso	OCtet	ORC	RD	SC	TO
Solid							
Hollow							

Implicit lattice representations

	Iso	Octet	ORC	RD	SC	TO
Solid						
Hollow						
3x3 Array						

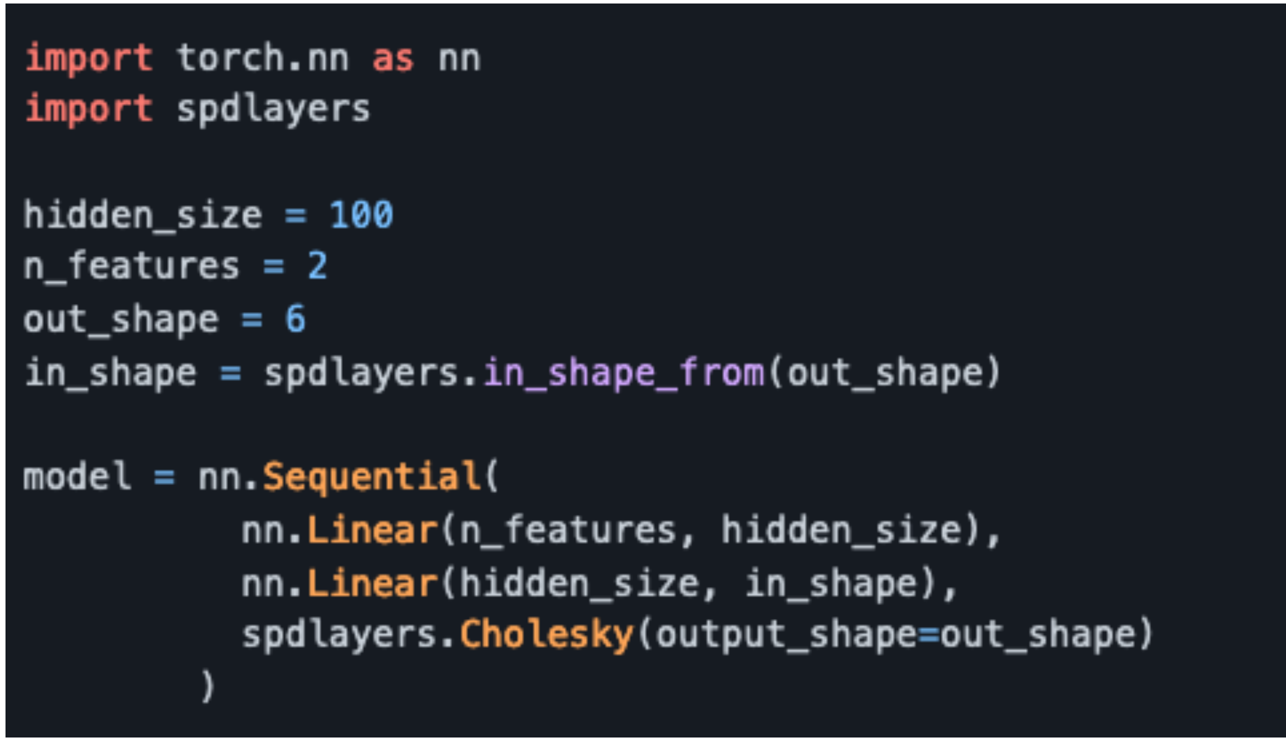
Triply Periodic Minimal Surfaces (TPMS)

	Gyroid (Schoen G)	Primitive (Schwarz-P)	Diamond (Schwarz-D)	Neovius	Schoen F-RD
Unit Cell					
Generating Function	f(x,y,z) = sin(x)cos(y) + sin(y)cos(z) + sin(z)cos(x)	f(x,y,z) = cos(x) + cos(y) + cos(z)	f(x,y,z) = sin(x)sin(y)sin(z) + sin(x)cos(y)cos(z) + cos(x)sin(y)cos(z) + cos(x)cos(y)sin(z)	f(x,y,z) = 3(cos(x)+cos(y)+cos(z)) + 4cos(x)cos(y)cos(z)	f(x,y,z) = cos(x)cos(y)cos(z) - 0.1(cos(2x)cos(2y)cos(2z)) + 0.1(cos(2x)cos(2y) + cos(2y)cos(2z) + cos(2z)cos(2x))

Enforcing that our ML models produce Symmetric Positive Definite (SPD) matrices

- Stiffness matrix predictions from ML model should be SPD
- Implemented SPD enforcement methods from literature as "neural network layers"
  - "spdlayers" improve model accuracy compared to models that don't enforce SPD [4]
- Model on the right:
  - Two inputs (rod radius, Poisson ratio)
  - Outputs Voigt stiffness matrix (6 x 6)
  - One hidden layer neural network, with SPD enforcement
- https://github.com/LLNL/spdlayers

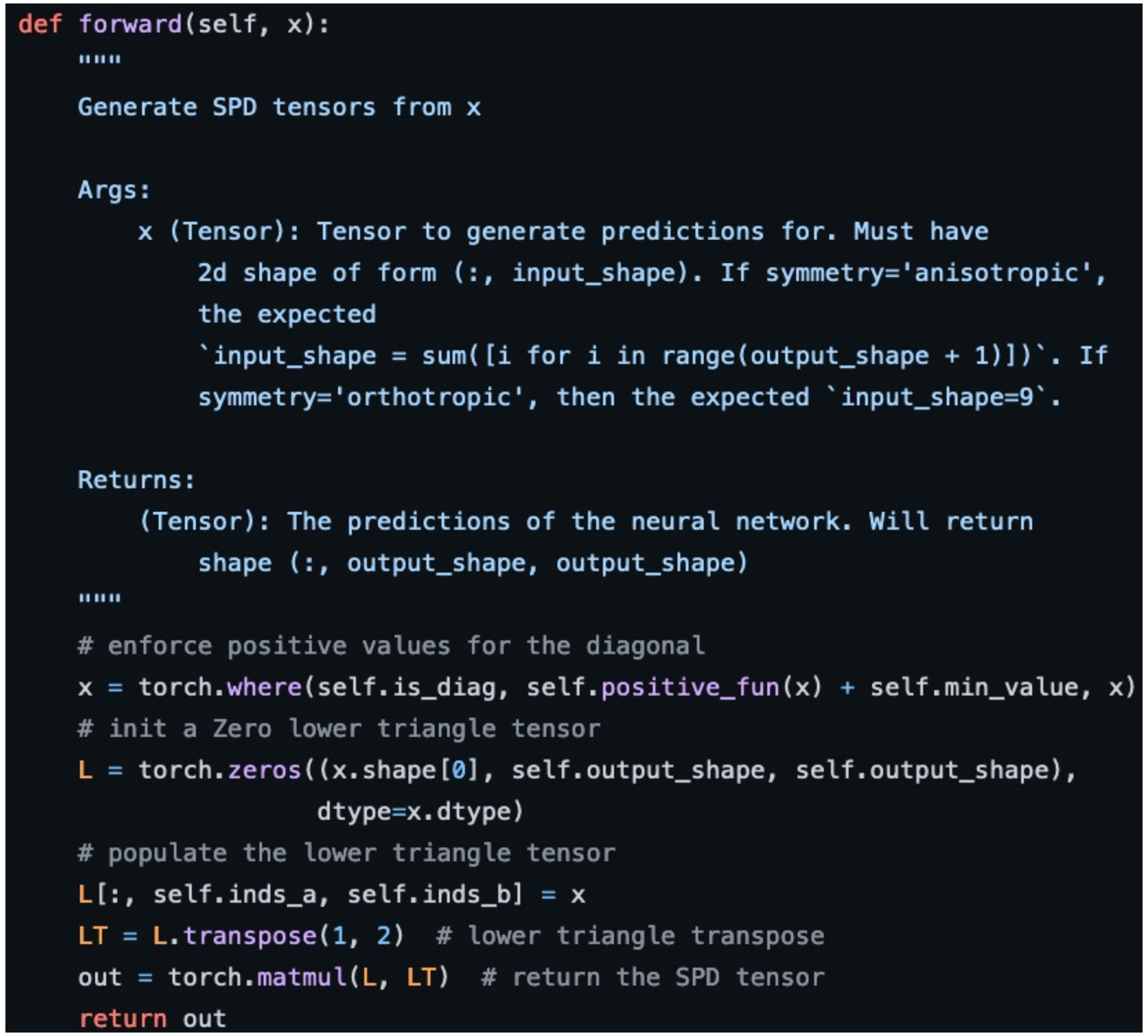
PyTorch



```python
import torch.nn as nn
import spdlayers

hidden_size = 100
n_features = 2
out_shape = 6
in_shape = spdlayers.in_shape_from(out_shape)

model = nn.Sequential(
          nn.Linear(n_features, hidden_size),
          nn.Linear(hidden_size, in_shape),
          spdlayers.Cholesky(output_shape=out_shape)
      )
```

How does spdlayers work?

- Cholesky decomposition [5] $C = LL^{\mathrm{T}}$
  - ML model predicts lower triangular form
  - "FIX" negative diagonal
  - Do LL^{T} with positive diagonal L
  - Code shown on right
- Exponential map [6]
  - Described as a tangent plane that is always SPD
  - Do eigenvalue decomposition
  - "FIX" negative eigenvalues
  - Reassemble C with corrected eigenvalues and original eigenvectors



```python
def forward(self, x):
    """

    Generate SPD tensors from x

    Args:
        x (Tensor): Tensor to generate predictions for. Must have
            2d shape of form (:, input_shape). If symmetry='anisotropic',
            the expected
            `input_shape = sum([i for i in range(output_shape + 1)])`. If
            symmetry='orthotropic', then the expected `input_shape=9`.

    Returns:
        (Tensor): The predictions of the neural network. Will return
            shape (:, output_shape, output_shape)
    """
    # enforce positive values for the diagonal
    x = torch.where(self.is_diag, self.positive_fun(x) + self.min_value, x)
    # init a Zero lower triangle tensor
    L = torch.zeros((x.shape[0], self.output_shape, self.output_shape),
                    dtype=x.dtype)
    # populate the lower triangle tensor
    L[:, self.inds_a, self.inds_b] = x
    LT = L.transpose(1, 2)  # lower triangle transpose
    out = torch.matmul(L, LT)  # return the SPD tensor
    return out
```

Fitting ML model via Sobolev norm to ensure accurate derivatives for topology optimization

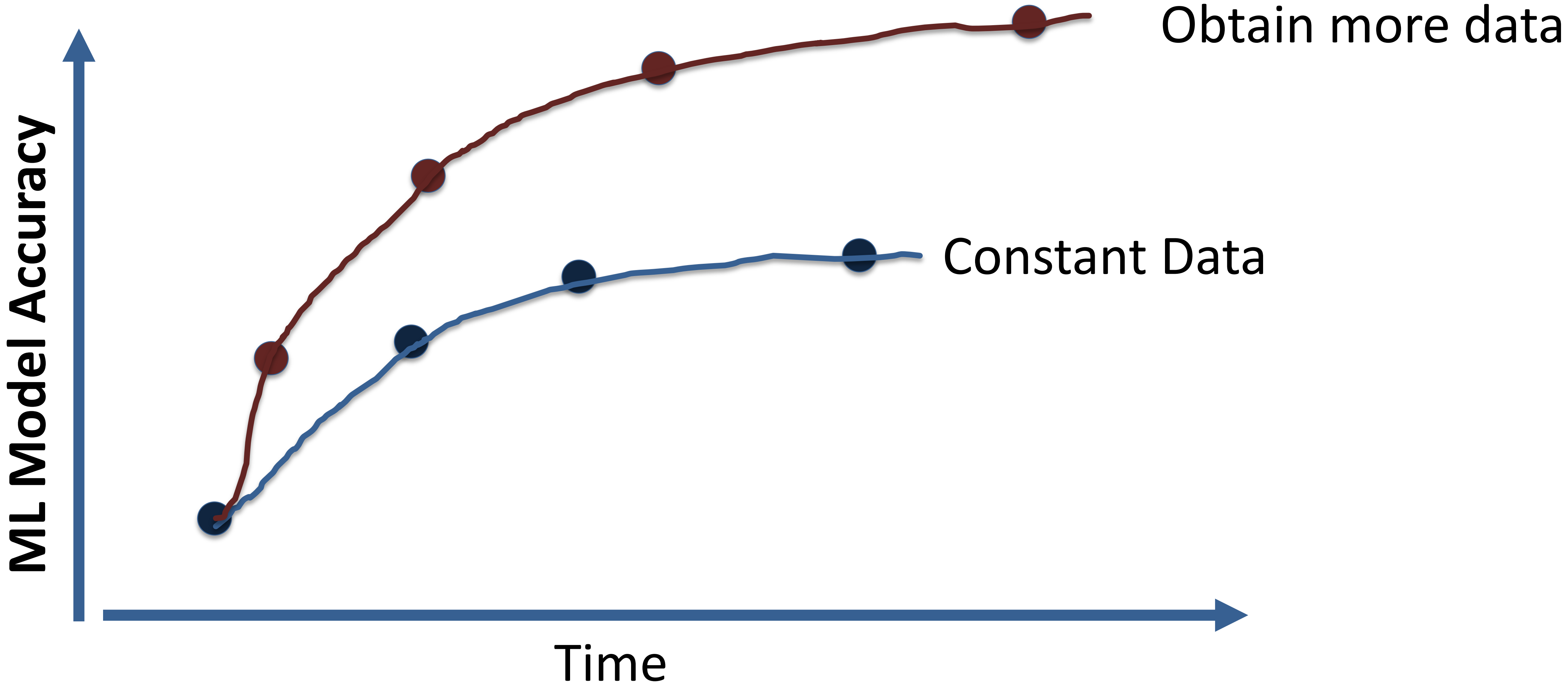
- **Derivates are important!**
- Our ML material model must be differentiable for topology optimization
- ML model is trained to minimize the Sobolev norm [7]
  - ML model parameters $\boldsymbol{\beta}$
  - **Hat** denotes ML prediction
  - Rod radius r
  - Voigt stiffness matrix $\boldsymbol{C}$
  - Sobolev weighting coefficient m

$$\min_{\boldsymbol{\beta}} \|\hat{\boldsymbol{C}}(r) - \boldsymbol{C}(r)\| + m\|\frac{\partial \hat{\boldsymbol{C}}(r)}{\partial r} - \frac{\partial \boldsymbol{C}(r)}{\partial r}\|$$

ML model woes

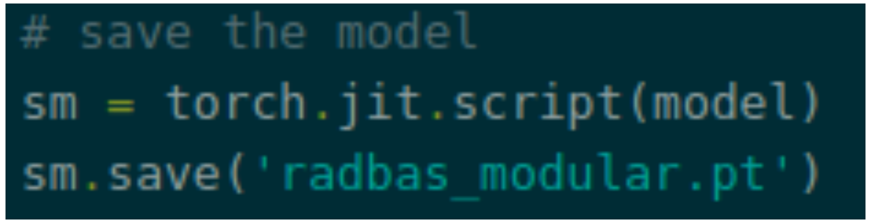
- For each microstructure we need ML models for
  - Volume fraction
  - Homogenized stiffness matrix
  - Homogenized thermal conductivity
  - Homogenized thermal expansion
- We need to train a lot of ML models
  - Not found one single ML model to rule them all that didn't sacrifice accuracy
  - **Training one ML model is easy**, **training LOTS of models is hard**!
    - Automation of this process is prone to soft failures
    - One particular model/data gives a poor fit (go back and retrain these by hand)
  - Neural networks have given more training pains
    - Occasionally the neural network training fails
  - Radial basis functions have some training pains
    - Slower and bigger than Neural Networks

How our ML knowledge changes over time

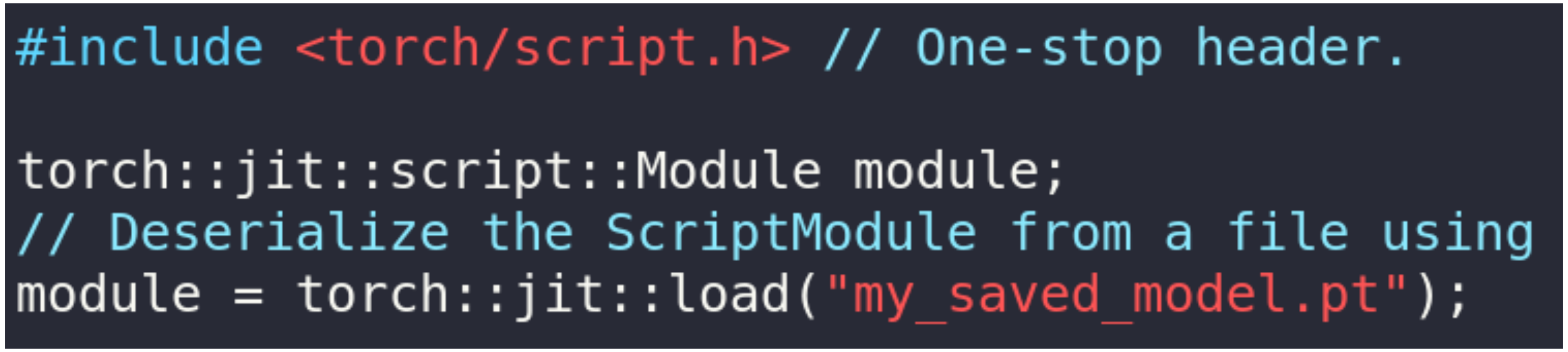


From training ML model to C++ production

- Leverage PyTorch’s Python to C++ persistence
- What we’ve proposed
  - Train ML models in Python
    - Rapid development and deployment
    - New data, new ML techniques
  - Save ML models as a single ‘.pt’ file
    - Binary zip file
    - Contains human readable metadata when extracted
  - Load the .pt file in c++ with no Python dependencies
    - Arbitrary models at runtime
    - **Get derivatives of models**
    - Using ‘libtorch’



```python
# save the model
sm = torch.jit.script(model)
sm.save('radbas_modular.pt')
```



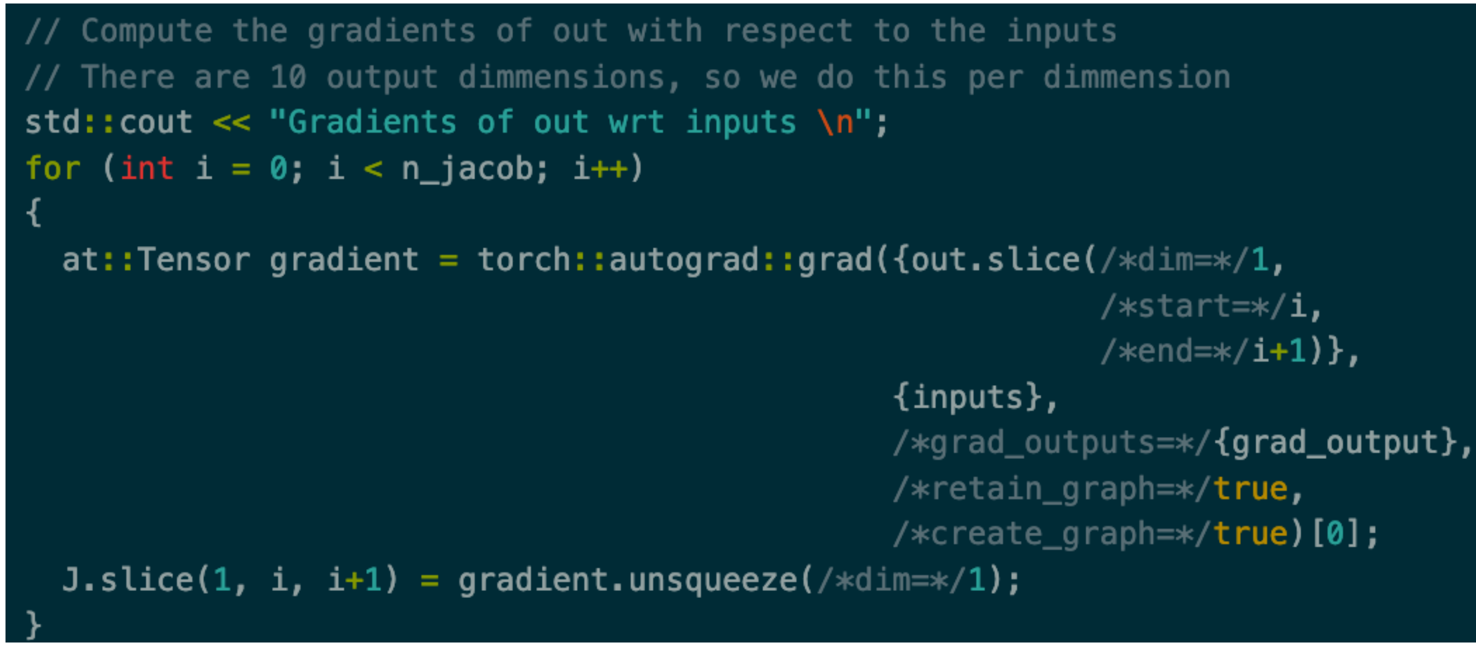
```cpp
#include <torch/script.h> // One-stop header.

torch::jit::script::Module module;
// Deserialize the ScriptModule from a file using
module = torch::jit::load("my_saved_model.pt");
```

Get derivatives of ML model at runtime

- Derivatives are **important**!
- Using PyTorch's autodiff
  - Don't need to code derivatives for each ML model
- Getting derivatives of **out** with respect to **inputs**

$$\frac{\partial \hat{C}(r)}{\partial r}$$

- This is getting implemented as a LiDO operator (Kenny Swartz)



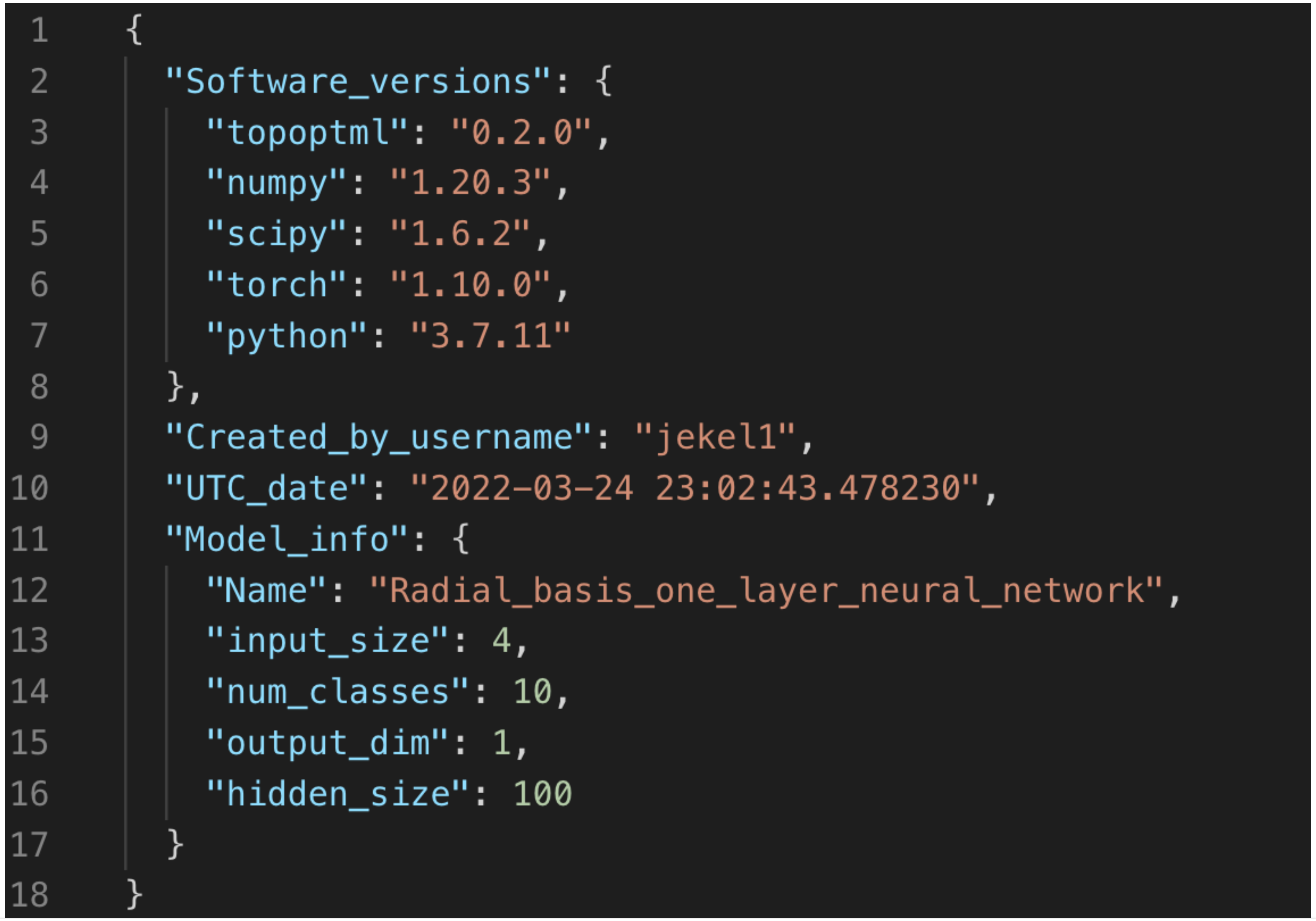
```
// Compute the gradients of out with respect to the inputs
// There are 10 output dimmensions, so we do this per dimmension
std::cout << "Gradients of out wrt inputs \n";
for (int i = 0; i < n_jacob; i++)
{
  at::Tensor gradient = torch::autograd::grad({out.slice(/*dim=*/1,
                                                         /*start=*/i,
                                                         /*end=*/i+1)},
                                              {inputs},
                                              /*grad_outputs=*/{grad_output},
                                              /*retain_graph=*/true,
                                              /*create_graph=*/true)[0];
  J.slice(1, i, i+1) = gradient.unsqueeze(/*dim=*/1);
}
```

Why have a hot swappable ML material models at runtime?

- As time goes on, we might get better ML techniques, or more data
- Can give the user a choice between
  - High accuracy but high inference time
  - Lower accuracy with faster inference time
- Quickly switch between microstructures at runtime
- Maybe you have a particular design situation that is starting to be expensive
  - Quickly train a lower accuracy ML model that would be much cheaper in inference!
- Other benefits:
  - Don’t have to implement each ML model architecture configuration in C++
  - Less C++ code to write
  - One unified interface

Metadata into the .pt files that tells us about the ML model

- You can put extra stuff into .pt files
- Example of metadata.json on the right
  - Human and machine readable
- Can access 'metadata' in c++ at runtime



```json
{
  "Software_versions": {
    "topoptml": "0.2.0",
    "numpy": "1.20.3",
    "scipy": "1.6.2",
    "torch": "1.10.0",
    "python": "3.7.11"
  },
  "Created_by_username": "jekel1",
  "UTC_date": "2022-03-24 23:02:43.478230",
  "Model_info": {
    "Name": "Radial_basis_one_layer_neural_network",
    "input_size": 4,
    "num_classes": 10,
    "output_dim": 1,
    "hidden_size": 100
  }
}
```

Thoughts on C++ implementation of the ML material models…

- I want your thoughts!
- ML (neural network) is fast when you give a large batch of computation
  - e.g. compute stiffness tensor for entire parallel mesh
  - Could run into memory issues, memory overhead is ~ **Batch_Size x Fattest_Hidden_Layer**
- ML is slow when you give only a single computation
  - e.g. compute the stiffness tensor per integration point
- ML model input:
  - Multi-dimensional array
  - Rod radii, Poisson's ratio
- ML model output:
  - Multi-dimensional array
  - Voigt stiffness matrix components

Summary

- Coming up with designs to fully leverage additive manufacturing techniques is hard
- We want to use multiscale topology optimization to design structures that can be directly 3D printed as lattices of microstructures
- Machine learning is one way to do multiscale topology optimization
- Homogenization stiffness matrix datasets for various microstructures
- Creating machine learning linear orthotropic material models for various structures
  - Input: geometric rod radius and Poisson's
  - Output: Homogenized stiffness matrix
- Happy to hear you thoughts, comments, and suggestions!

References

1. Zheng X, Smith W, Jackson J, Moran B, Cui H, Chen D, Ye J, Fang N, Rodriguez N, Weisgraber T, Spadaccini CM. Multiscale metallic metamaterials. Nature materials. 2016 Oct;15(10):1100-6.

2. Guest JK, Prévost JH, Belytschko T. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. International journal for numerical methods in engineering. 2004 Sep 14;61(2):238-54.

3. Watts S, Arrighi W, Kudo J, Tortorelli DA, White DA. Simple, accurate surrogate models of the elastic response of three-dimensional open truss micro-architectures with applications to multiscale topology design. Structural and Multidisciplinary Optimization. 2019 Nov;60(5):1887-920.

4. Jekel CF, Swartz KE, White DA, Tortorelli DA, Watts SE. Neural Network Layers for Prediction of Positive Definite Elastic Stiffness Tensors. arXiv preprint arXiv:2203.13938. 2022 Mar 25.

5. Xu K, Huang DZ, Darve E. Learning constitutive relations using symmetric positive definite neural networks. Journal of Computational Physics. 2021 Mar 1;428:110072.

6. Amsallem D, Cortial J, Carlberg K, Farhat C. A method for interpolating on manifolds structural dynamics reduced-order models. International journal for numerical methods in engineering. 2009 Nov 26;80(9):1241-58.

7. White DA, Arrighi WJ, Kudo J, Watts SE. Multiscale topology optimization using neural network surrogate models. Computer Methods in Applied Mechanics and Engineering. 2019 Apr 1;346:1118-35.

Lawrence Livermore National Laboratory

Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.