Automating a FEM solution database generation and neural network learning for solid mechanics problems

Leopoldo Agorio[a], Mauricio Vanzulli[b], Bruno Bazzano[c], Jorge Pérez Zerpa[c]

a - IIE, FIng UdelaR, b- IIMPI, FIng UdelaR, c- IET, FIng UdelaR

´April 17, 2023

This project explores the use of a neural network to solve simple solid mechanics problems more quickly than the finite element method (FEM). We developed a pipeline for generating a database of FEM solutions to compression/extension problems and used it to train a neural network. We present the results of our experiments, including training and test loss, and compare our model to the FEM. Our approach has the potential to significantly improve the efficiency of solving mechanical problems, with broader implications for the engineering design process and any application requiring computational expensive simulations.

1 Introduction

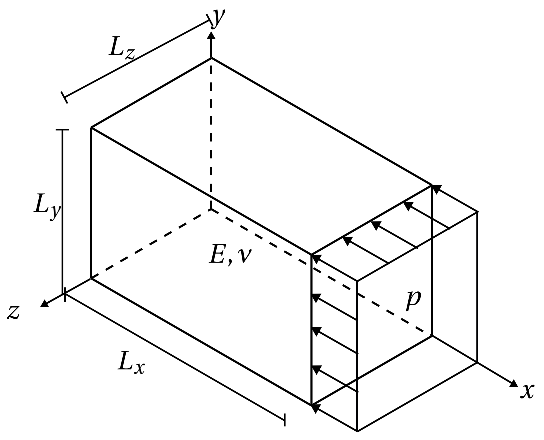
The objective of this project is to develop a surrogate model for unidimensional mechanical problems that is faster than the finite element method (FEM). To achieve this, we trained a neural network using a dataset of FEM solutions to compression/extension mechanical problems. The ultimate goal is to apply this approach to develop state-of-the-art surrogate models for biological tissues. In this project, we used a FEM solver repeatedly to generate a large dataset of results, which was used to train the neural network. The neural network was designed to learn from the FEM solutions and solve similar mechanical problems more quickly.
This documentation report will describe the methodology used to develop the neural network, the performance of the model compared to the FEM, and potential future directions for this project. By developing a faster surrogate model, this project's approach has the potential to significantly improve the efficiency of solving unidimensional mechanical problems, with broader implications for the engineering design process.

2 Methodology

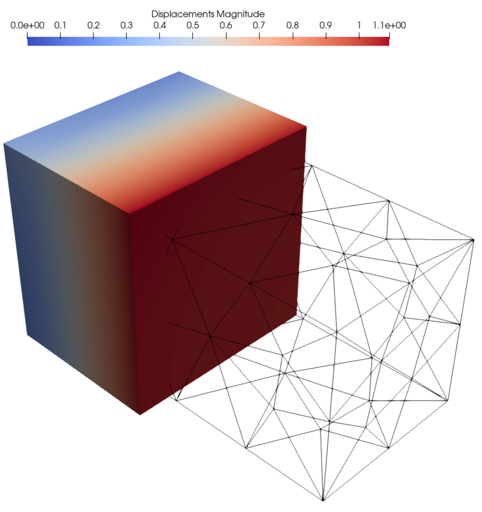
In this section, we describe the pipeline we developed to generate the dataset of FEM solutions and train the neural network. We used a simple solid mechanical models with the `ONSAS.m` FEM solver, which we automated with a bash script. The resulting dataset was preprocessed and used to train a multi-layer perceptron (MLP) implemented in PyTorch.

2.1 Uniaxial Model

This model consisted of a uniaxial mechanical problem in which a load was applied to a three-dimensional prism, causing it to deform in three dimensional. The model is uniaxial in the sense that the applied load is along the x-axis. We used the `ONSAS.m` FEM solver to simulate the deformation and generate a dataset of solutions. The uniaxial model was chosen for its simplicity and as a starting point for more complex models.
The simulated uniaxial problem can be seen in Figure 2.1. Loads are applied to the prism at the right end, causing it to deform. The prism is constrained to to mantain the lefmost face (Figure 2.1) in the y-z plane, and the origin fixed, thus the deformation is uniaxial.



(a) Reference configuration sketch.



(b) System with an applied load.

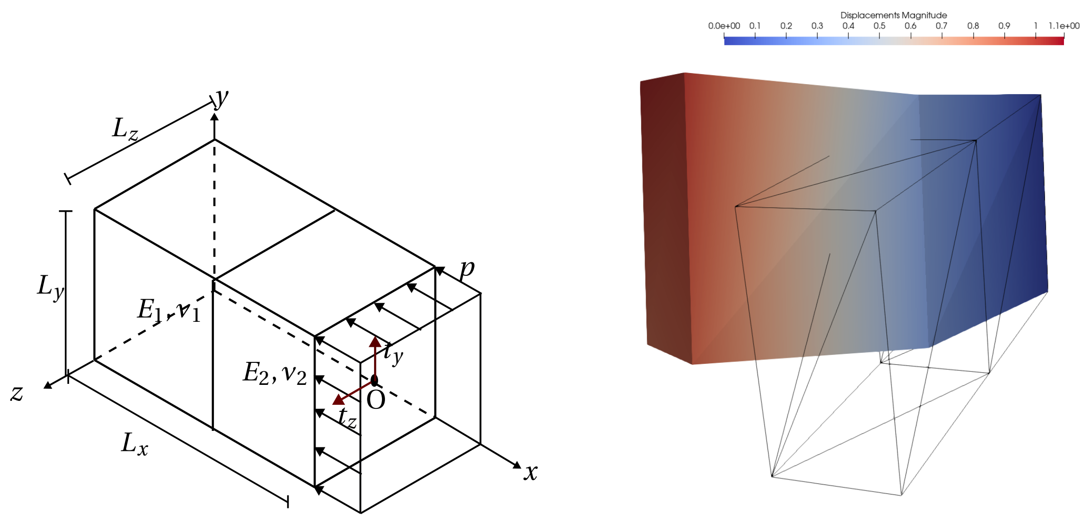
Figure 2.1: Uniaxial problem simulated and solved in `ONSAS.m`.

2.2 Composed Cantilever Model

Our composed cantilever model consists of two prisms made of different materials, denoted as 1 and 2, for which the elastic moduli and Poisson ratios are E_1, ν_1 and E_2, ν_2, respectively. Prism 1 is fixed by its face at the $x = 0$ plane. We applied a load of $(-p_x, t_y, t_z)$ at the plane located at $x = Lx$ along the x, y, and z axes, respectively as shown in Figure 2.2.

2.3 Bash Script

To automate the FEM solver, we developed a bash script that generated the input files for the `ONSAS.m` solver and executed it repeatedly to generate a large dataset of solutions. The script also handled the output files, extracting the relevant data and formatting into a `.csv` it for further processing.
The variables swept by the script were the length of the prism Lx (ranging from 1 to 3), the applied load p (ranging from 0.1 to 3), and the elastic modulus of the material E (ranging from



(a) Reference configuration sketch.

(b) System with an applied load.

Figure 2.2: Composed cantilever problem simulated and solved in `ONSAS.m`.

1 to 4). Ly and Lz, were fixed at 1, and the Poisson ratio, ν, was fixed at 0.3. The output of the solver was processed to obtain the displacement of the center point of the prism's face in the three dimensions. The script generated a total of 18900 data points, which were used to train the neural network.

For the composed cantilever model, we swept the variables including the length of the prism (Lx, ranging from 1 to 2), the applied load (p, ranging from 0.05 to 0.14), and the elastic moduli of each material (E_1 and E_2, ranging from 1.4 to 2.4). Fixed values were $Ly = 1$,$Lz = 0.5$ and $\nu_1 = \nu_2 = 0.3$. The output of the solver was processed to obtain the displacement of the center point of the prism's face in the three dimensions. The script generated a total of 13310 data points, which were used to train the neural network.

2.4 Neural Network

We implemented an MLP in PyTorch to learn from the FEM solutions and solve similar mechanical problems more quickly. The MLP had two hidden layers with 20 and 10 neurons, respectively, and used the ReLU activation function. The output layer had three neurons, corresponding to the three dimensions of the displacement of the center point of the prism's face. The optimizer was Adam with a learning rate of 0.001 and 100 epochs were used.

In the uniaxial case we used a 50/50 split for the training and validation sets, and 200 test samples using the analytic solution. For the composed cantilever case we used 1000 samples for both the training and validation sets.

We report two different loss functions: the mean squared error (MSE) and the relative MSE. The MSE is the most common loss function for regression problems, and is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

where y_i is the true value and $\hat{y}_i$ is the predicted value. The relative MSE is defined as:

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\frac{||y_i - \hat{y}_i||_2^2}{||y_i||_2^2} \tag{2.2}$$

The relative MSE is more useful for comparing the performance of different models, since it is normalized by the true value.

2.5 Analytic Solution

To compare the performance of our neural network to the FEM, we computed the analytic solution to the uniaxial problem. The analytic solution is given by:

$$\boldsymbol{u} = (\alpha - 1)L_x\boldsymbol{e}_1 + (\beta - 1)\frac{L_y}{2}\boldsymbol{e}_2 + (\beta - 1)\frac{L_z}{2}\boldsymbol{e}_3 \tag{2.3}$$

where $\boldsymbol{u}$ is the displacement of the center point of the prism's face, L_x, L_y and L_z are the dimensions of the solid as it is shown in Figure 2.2a. The values of α and β are obtained solving the following pair of non-linear equations:

$$\alpha\left(\mu - \frac{\mu}{\alpha^2} + \frac{K\beta^2}{\alpha}(\beta^2\alpha - 1)\right) = -p \tag{2.4}$$

$$\beta\left(\mu - \frac{\mu}{\beta^2} + K(\alpha^2\beta^2 - \alpha)\right) = 0 \tag{2.5}$$

where p is the applied load, $K = \frac{E}{3(1-2\nu)}$ is the Bulk modulus and $\mu = \frac{E}{2(1+\nu)}$ is the second Lamé parameter, E is the elastic modulus and ν is the Poission's ratio.
This allows us to generate a dataset of analytic solutions, which we can use to compare the performance of the neural network to the FEM. For doing so we used lhs to generate a Latin Hypercube Sampling of the input variables, and then computed the analytic solution for each of the 200 points in the sample.

3 Results

In this section, we present the results of our experiments. We compare the performance of our neural network to the FEM, and evaluate its effectiveness on a uniaxial problem and a composed cantilever problem.

3.1 Evaluation for Uniaxial Compression

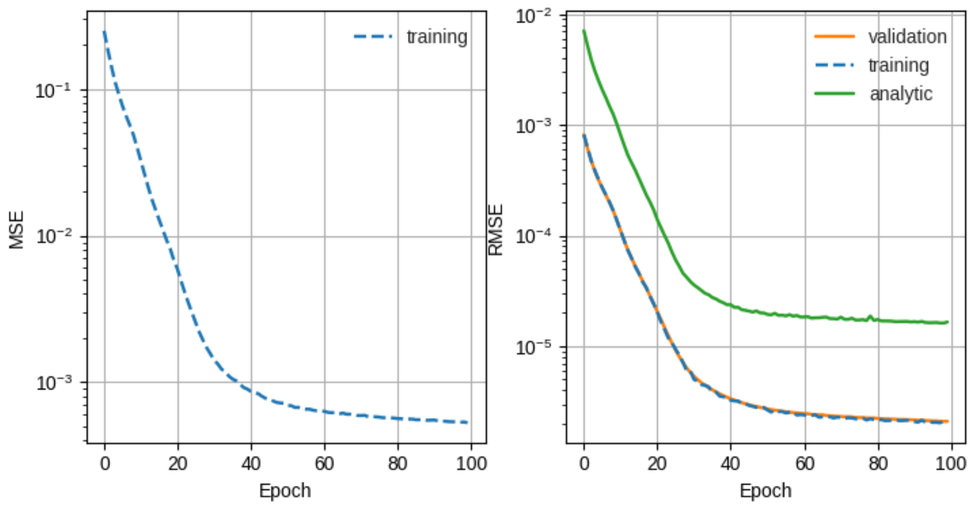


Figure 3.1: Training, validation and test loss of the neural network on the uniaxial compression/extension dataset.

Figure 3.1 shows the training, validation, and test losses of our neural network on the uniaxial compression/extension dataset. The final training loss for MSE is 5.29×10^{-4} and the RMSE is 2×10^{-6}. The final validation loss RMSE is 2×10^{-6}, and the final analytic loss RMSE is 1.7×10^{-5}. We also evaluated the ability of our model to extrapolate beyond the parameters in the training set. When we measured the RMSE extrapolation error for $(Lx, E, p) = (4.00, 0.50, 3.50)$, we found it to be 6.7×10^{-2}, while the MSE extrapolation error was $9.5 \times 10^{-1}\ m^2$. Since the variation range for the training parameters was $(Lx, E, p) = (1 \ldots 3, 1 \ldots 4, .1 \ldots 3)$, these results show that our neural network was able to extrapolate to a point that was not in the training set, with an error of 6.7 of the true value.

3.2 Evaluation on Cantilever Model

For the case of the composed cantilever model, the training, validation and test loss is shown in Figure 3.2. As we can see, the training loss decreases rapidly during the first few epochs and then levels off.

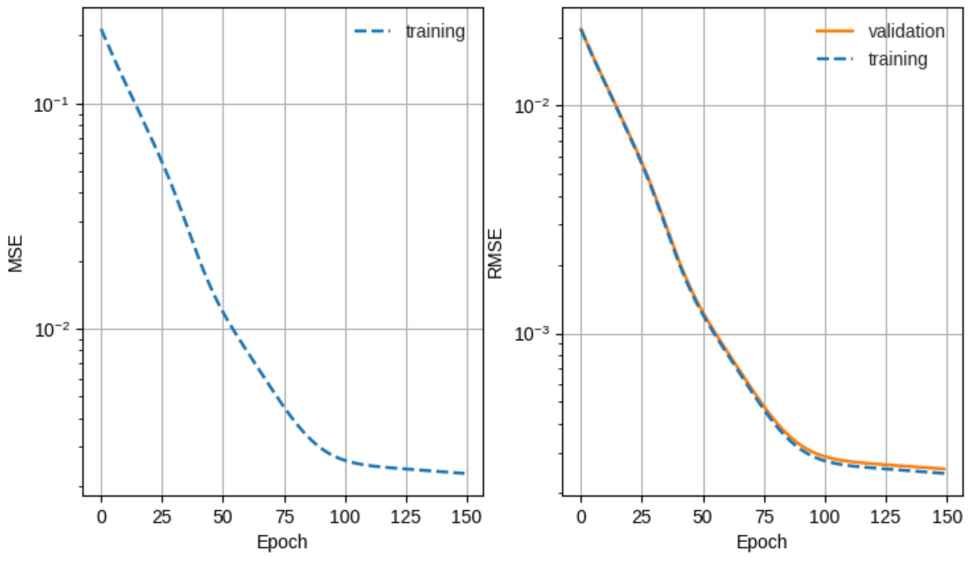


Figure 3.2: Training, validation and test loss of the neural network on the composed cantilever dataset.

Figure 3.2 shows the training, validation, and test losses of our neural network on the cantilever model. The final training loss for MSE is 2.3×10^{-3} and the RMSE is 2.4×10^{-4}. The final validation loss RMSE is 2.5×10^{-4}, and the final analytic loss RMSE is 2.9×10^{-4}.
We also evaluated the ability of our model to extrapolate beyond the parameters in the training set. When we measured the RMSE extrapolation error for $(Lx, E1, E2, p) = (1.50, 1.3, 1.3, 0.10)$, we found it to be 0.21. Since the variation range for the training parameters was $(Lx, E1, E2, p) = (1 \ldots 2, 1.4 \ldots 2.4, 1.4 \ldots 2.4, 0.05 \ldots 0.14)$, these results show that our neural network was able to extrapolate to a point that was not in the training set, with an error of 21% of the true value.

4 Conclusion

In this project, we developed a pipeline for generating a dataset of FEM solutions to simple solid mechanical problems, and used this dataset to train a neural network to solve the same problems more quickly. Our experiments showed that our approach was effective, with the neural network achieving low losses on both the training and test datasets, and closely matching the theoretical curve for uniaxial compression. We also demonstrated that our approach has the potential to be extended to more complex mechanical problems, as shown by our evaluation on a cantilever model.

One advantage of our implementation is its simplicity and scalability. By using a relatively simple mechanical problem, we were able to develop a pipeline that can be easily scaled to more complex problems. Additionally, the use of a neural network allows for faster computation times than the traditional FEM method. This has the potential to significantly reduce computational costs on any numerical processes that rely on FEM simulations, such as material identification algorithms for tissue disease diagnostics problems and stress CAD for manufacturing.

In conclusion, this project demonstrates the effectiveness of using a neural network to solve simple mechanical problems. By developing a faster surrogate model, we have shown the potential to significantly improve the efficiency of solving such problems, with broader implications for the engineering design process. Future work includes scaling our approach to more complex mechanical problems and evaluating its effectiveness on a larger dataset.