Technische Universität Berlin

Institut für Mechanik

FG Strukturmechanik und Strukturberechnung

Machine Learning in Computational Mechanics

Task Sheet 2

Task 1 Multivariate, nonlinear regression

1 Univariate Nonlinear Regression

The given code implements a Neural Network(NN) model designed to fit a univariate, nonlinear sine function: $y = \sin(x)$. The following hyperparameters are used:

- Number of samples (`samples`): 16
- Number of neurons in the hidden layer (`hidden_dim`): 16
- Number of training epochs (`epochs`): 1600
- Learning rate (`lr`): 0.001
- Batch size (`batch_size`): 8

The training results are shown in Figure 1 and Figure 2.

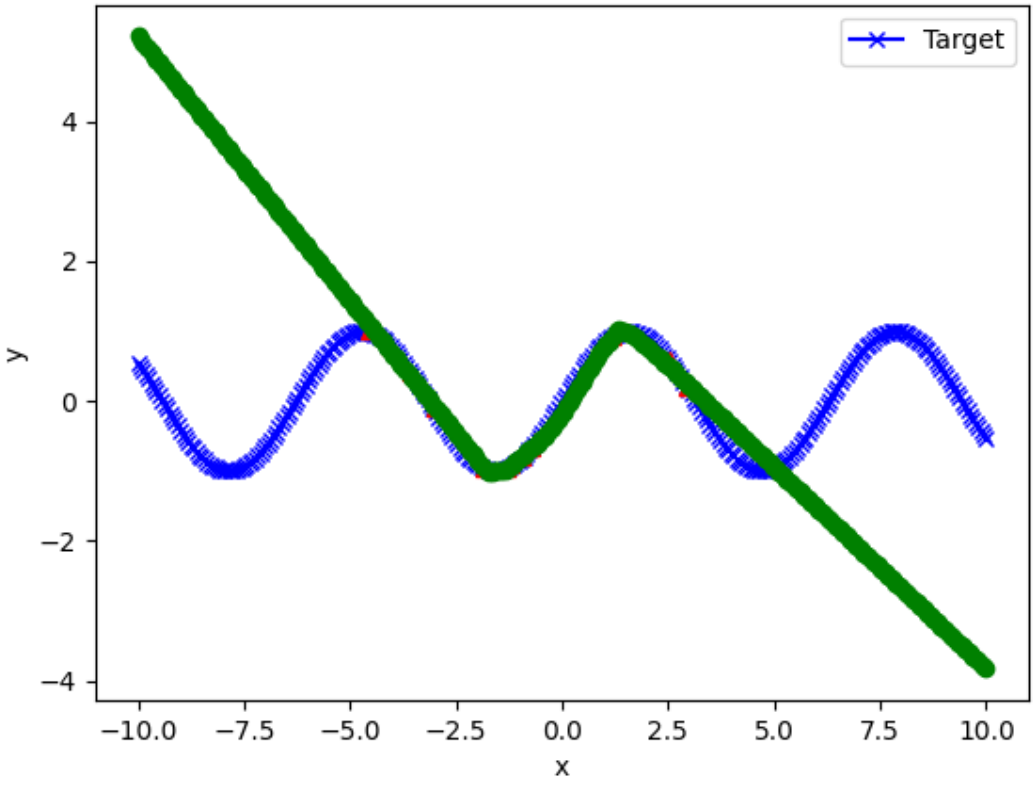

Figure 1: Fitted Surface

Figure 1 shows that the neural network successfully learned the sine function within the training range (-5 to 5), with predictions (green line) closely matching the target values (blue line). However, outside this range ($x < -5$ and $x > 5$), the model fails to generalize and produces linear approximations instead of capturing the oscillatory nature of the sine function.

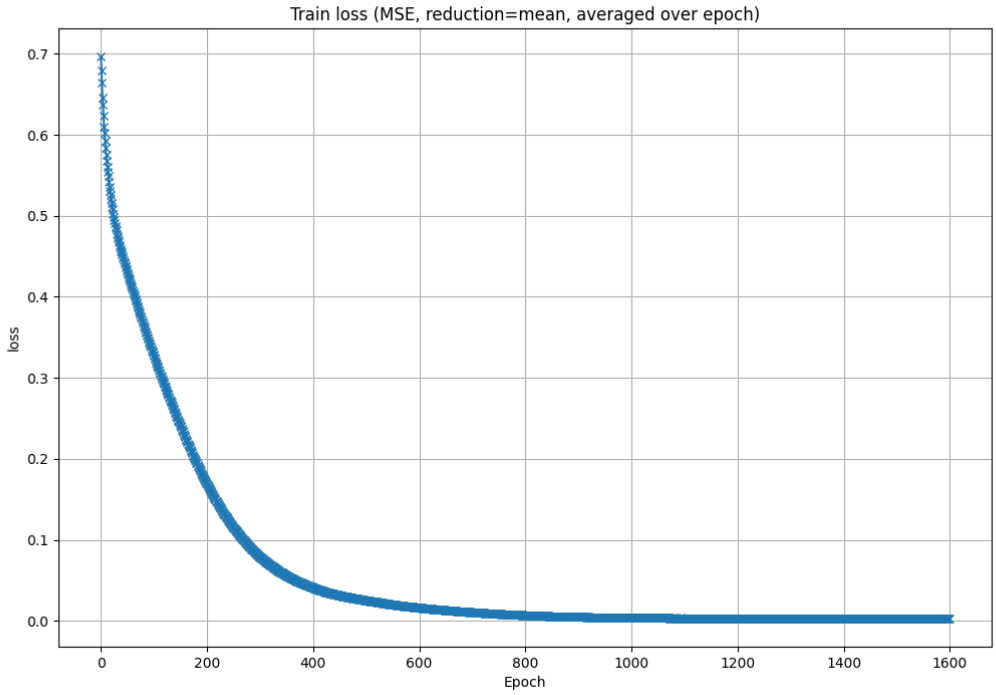

Figure 2: Loss convergence over epochs

Figure 2 illustrates a smooth decline in training loss, which rapidly decreased during the initial epochs and stabilized around zero after epoch 800, indicating effective learning without overfitting.

Although the low training loss confirms that the model fits the training data well, its poor extrapolation highlights the need for additional data, regularization, or periodicity-aware architectures to improve generalization.

2 Multivariate Nonlinear Regression

This is a nonlinear regression using a Neural Network(NN) model for a bivariate function. The target function is: $z = x^2 + y^2$. The graph of this function should form a three-dimensional paraboloid. The hyperparameters are consistent with previous univariate nonlinear regression model as follows:

- Number of samples (`samples`): 16
- Number of neurons in the hidden layer (`hidden_dim`): 16
- Number of training epochs (`epochs`): 1600
- Learning rate (`lr`): 0.001
- Batch size (`batch_size`): 8

The training results are shown in Figure 3 and Figure 4.

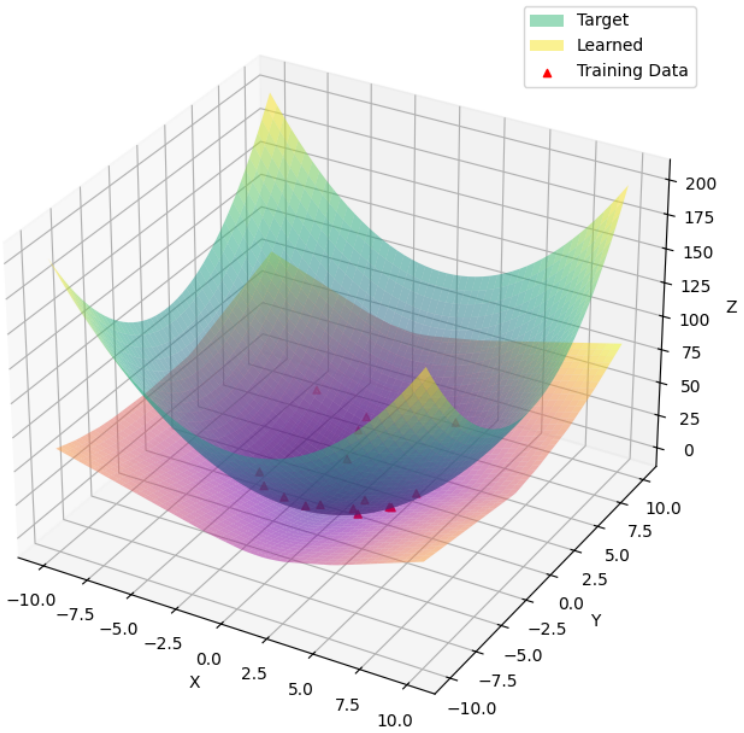

Figure 3: Fitted Surface

Figure 3 shows that the "Learned" surface (yellow) closely aligns with the "Target" surface (green) in the regions covered by the training data (red triangles), indicating effective learning within the sampled range. However, in regions far from the training data, the model's ability to generalize is limited, as limited sample coverage (16 points) restricts the network's understanding of the full quadratic surface.

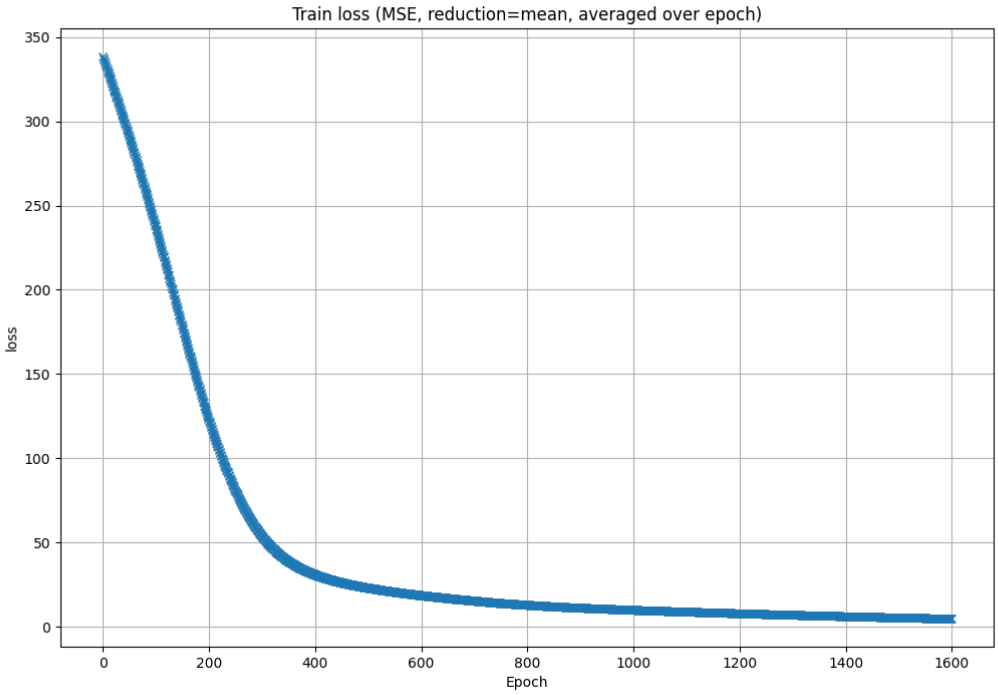

Figure 4: Loss convergence over epochs

Figure 4 illustrates a smooth decline in training loss, starting at approximately 350 and rapidly decreasing during the initial epochs before stabilizing around zero after epoch 800. This indicates that the model effectively minimized the error in the training data without overfitting.

Although the low training loss confirms that the model fits the training data well, its limited sample coverage and relatively simple architecture may hinder its ability to generalize effectively. Increasing the number of samples or improving the data distribution could enhance the model's extrapolation performance.

2.1 Improve Fitting Accuracy

2.1.1 Increase the number of training samples

The hyperparameters are changed as follows:

- **Number of samples (`samples`): 1000**
- Number of neurons in the hidden layer (`hidden_dim`): 16
- Number of training epochs (`epochs`): 1600
- Learning rate (`lr`): 0.001
- Batch size (`batch_size`): 8

The training results are shown in Figure 5 and Figure 6.

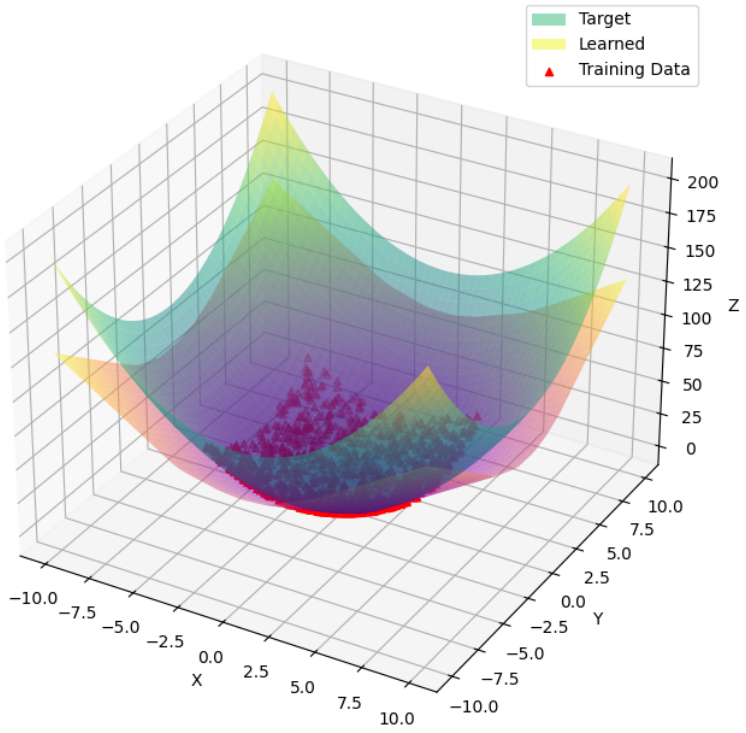

Figure 5: Fitted Surface

After increasing the number of samples, as shown in Figure 5, the fitted surface of the model is closer to the objective function, and the accuracy of the result is significantly improved. In particular, in extrapolation areas, the generalization ability of the model is significantly improved compared to the earlier scenarios with only 16 samples.

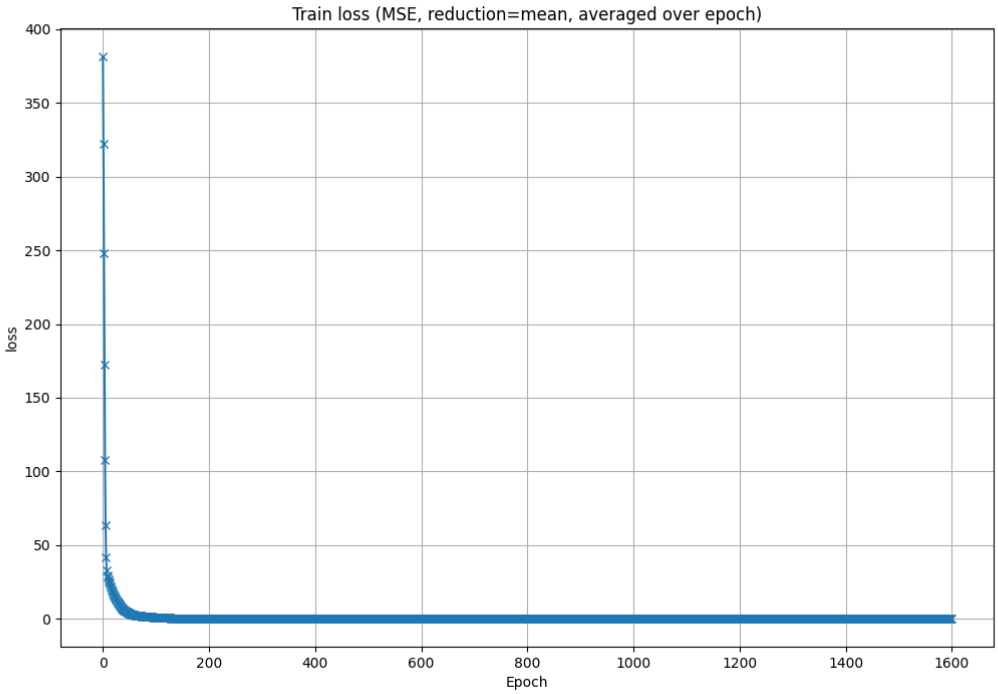

Figure 6: Loss convergence over epochs

As shown in Figure 6, the loss function graph shows that in the first few hundred rounds of training, the loss rapidly decreases, indicating that the model quickly learns the main functional characteristics in the early stages. After increasing the number of samples, the loss function decreases more smoothly and lower than before. In subsequent training, the number of training epochs can be appropriately reduced to save training time and cost.

Overall, after increasing the number of samples, the fitting accuracy, convergence speed and generalization ability of the model are improved. It indicates that for relatively complex objective functions, increasing the number of samples does help improve the learning ability of the neural network.

2.1.2 Increase the number of neurons in the hidden layer

The hyperparameters are changed as follows:

- Number of samples (`samples`): 1000
- **Number of neurons in the hidden layer (`hidden_dim`): 64**
- Number of training epochs (`epochs`): 1600
- Learning rate (`lr`): 0.001
- Batch size (`batch_size`): 8

The training results are shown in Figure 7 and Figure 8.

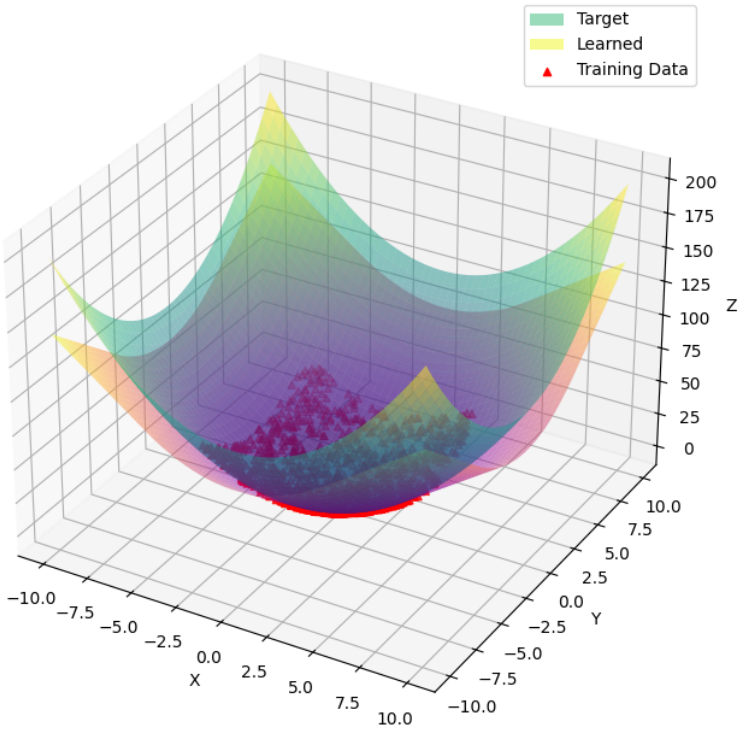

Figure 7: Fitted Surface

According to the results in Figure 7, increasing the number of neurons in the hidden layer does make the model learn smoother and more accurate surfaces than the previous model. The fitting accuracy of the model is improved both within the training data region and the extrapolation region, and its ability to generalize has been enhanced to a certain extent. However, the overall performance improvement gain from this adjustment is not significant, which indicates that increasing the number of neurons is approaching the stage of diminishing returns in this model.

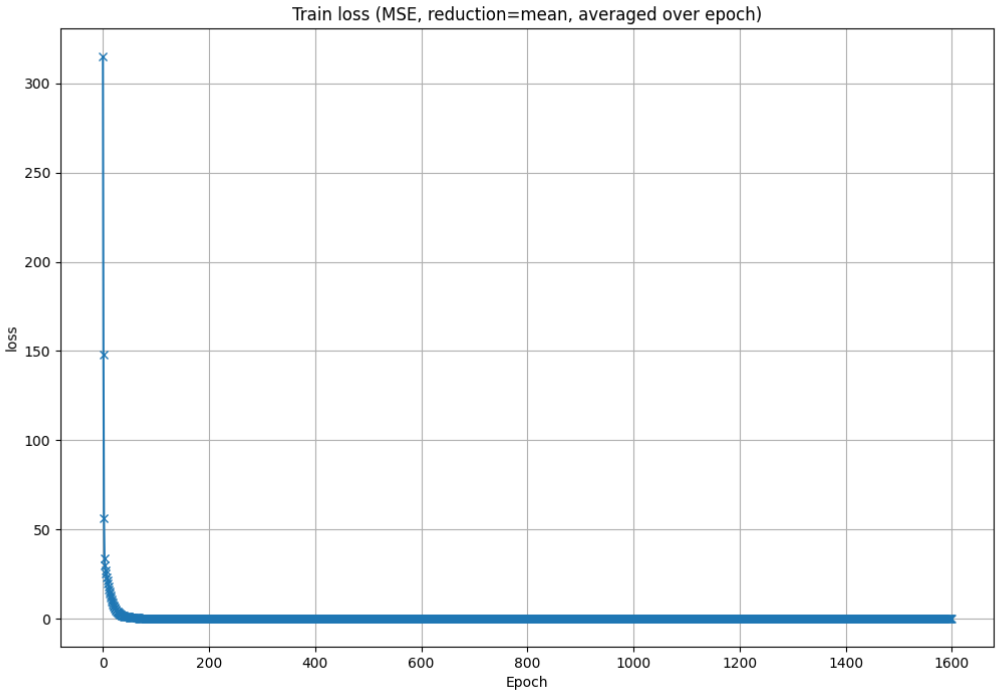

Figure 8: Loss convergence over epochs

As shown in Figure 8, the loss curve also shows rapid initial convergence and stabilizes at a lower loss value. This reflects effective learning with increased model complexity.

Overall, after increasing the number of neurons, the fitting accuracy and stability of the model are improved, but the final learning effect is limited.

2.1.3 Increase epochs and batch_size, decrease learning rate

The hyperparameters are changed as follows:

- Number of samples (`samples`): 1000
- Number of neurons in the hidden layer (`hidden_dim`): 64
- **Number of training epochs (`epochs`): 3000**
- **Learning rate (`lr`): 0.0001**
- **Batch size (`batch_size`): 16**

The training results are shown in Figure 9 and Figure 10.

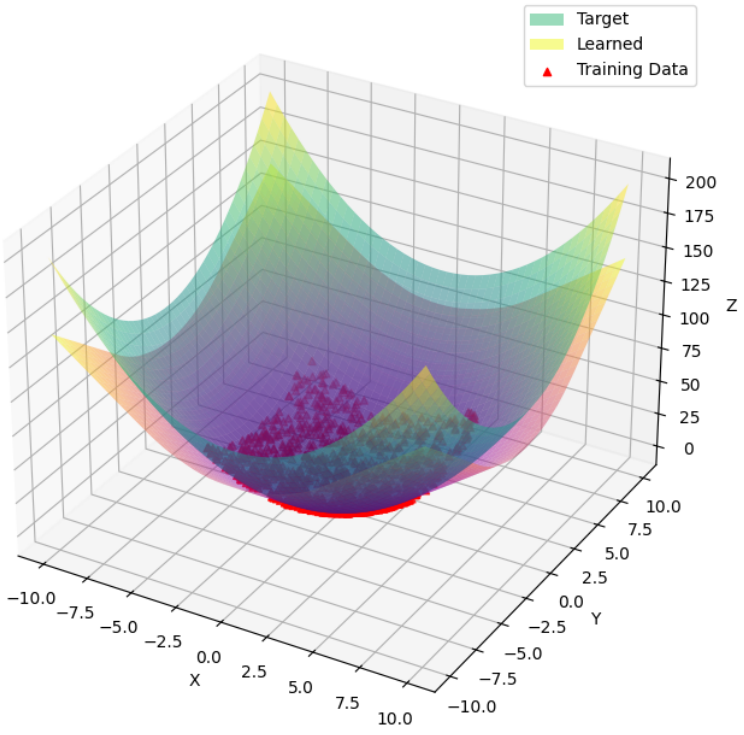

Figure 9: Fitted Surface

As shown in Figure 9, greatly increasing epochs and batch size while reducing the learning rate has brought a certain degree of improvement to the fitting accuracy of the model. However, this improvement is very slight and does not significantly differ from the previous model.

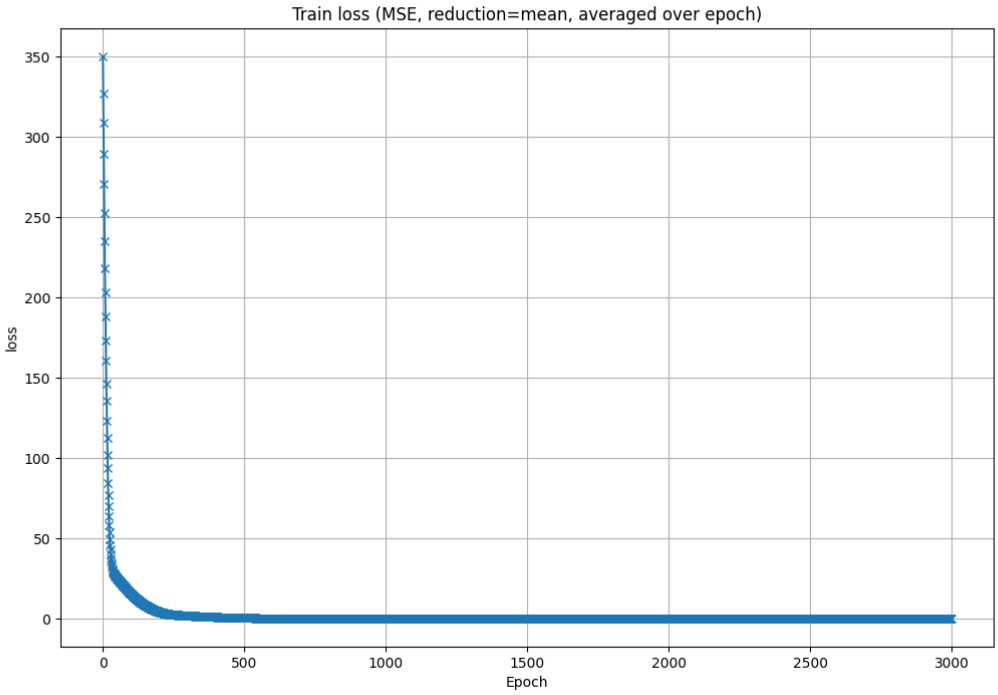

Figure 10: Loss convergence over epochs

As shown in Figure 10, the loss function has not significantly changed either, comparing to Figure 8.

To sum up, both Figure 9 and Figure 10 indicate that the model is close to convergence. epochs = 3000 and lr = 0.0001 only provide marginal improvements at the cost of excessive training time, especially if the loss function has already stabilized with fewer epochs and learning rate as in Section 2.1.2.

Therefore, to balance performance and efficiency, we choose the hyperparameters samples = 1000, hidden_dim = 64, epochs = 3000, lr = 0.0001 and batch_size = 16 from Section 2.1.2 as our solution to get sufficiently accurate results. The visualisation results are shown in Figure 7 and Figure 8.

2.2 Inappropriate Hyperparameter

2.2.1 Large Learning Rate

The hyperparameters are changed as follows:

- Number of samples (`samples`): 1000
- Number of neurons in the hidden layer (`hidden_dim`): 64
- Number of training epochs (`epochs`): 1600
- **Learning rate (`lr`): 0.1**
- Batch size (`batch_size`): 8

The training results are shown in Figure 11 and Figure 12.

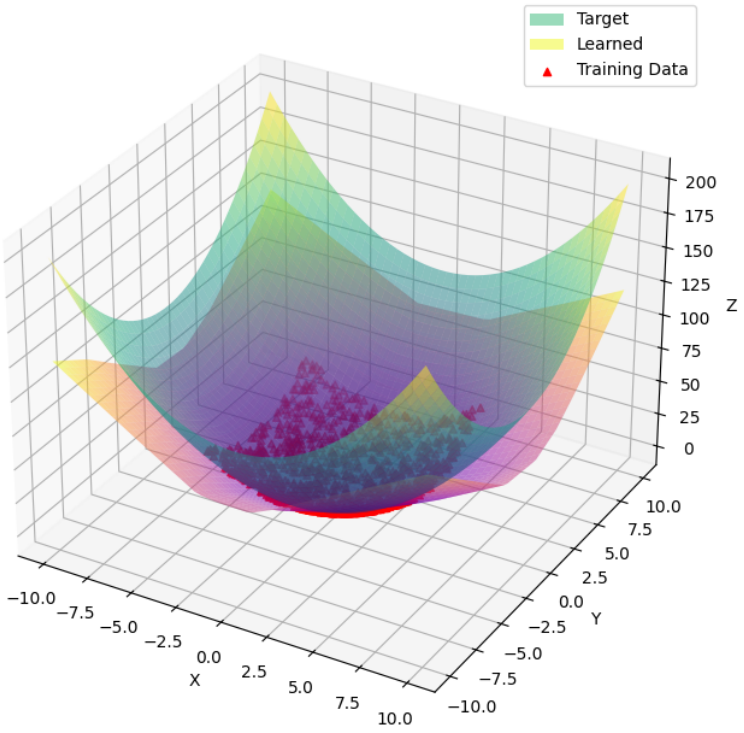

Figure 11: Loss convergence over epochs

As shown in Figure 11, when the learning rate increased to lr = 0.1 under the same conditions as in Section 2.2, the model's fit deteriorates significantly compared to Figure 7. This comparison highlights that, with other hyperparameters held constant, an excessively high learning rate leads to a substantial decrease in model accuracy, reduced generalization capability, and a less smooth surface.

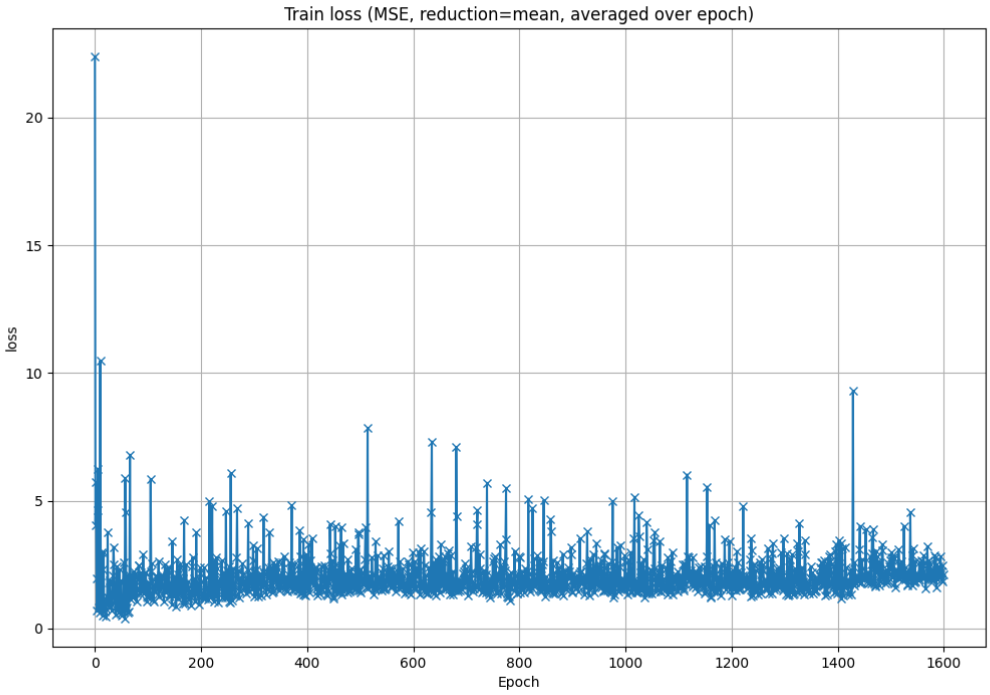

Figure 12: Loss convergence over epochs

As shown in Figure 12, the loss value can be quickly reduced at the beginning. However, due to the excessive update range, the loss curve is highly irregular, with significant fluctuations throughout the training process. The lack of consistent decline and the presence of spikes indicate that the model struggles to achieve a stable optimization. It cannot gradually approach the optimal solution and stabilize at the optimal solution, resulting in low training efficiency.

A large learning rate can lead to excessively large update steps for the parameters, potentially causing the model to skip over the optimal solution. This can result in an unstable training process that fails to converge effectively, as reflected in the fluctuating loss curve. While a higher learning rate may accelerate convergence initially, it often sacrifices precision and stability.

2.2.2 Small Batch Size

The hyperparameters are changed as follows:

- Number of samples (`samples`): 1000
- Number of neurons in the hidden layer (`hidden_dim`): 64
- Number of training epochs (`epochs`): 1600
- Learning rate (`lr`): 0.001
- **Batch size (`batch_size`): 1**

The training results are shown in Figure 13 and Figure 14.

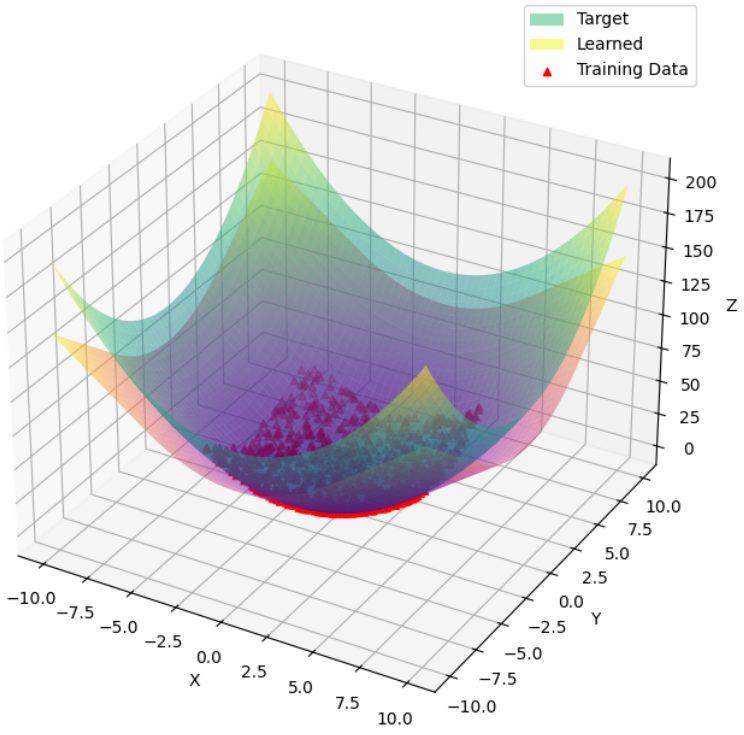

Figure 13: Fitted Surface

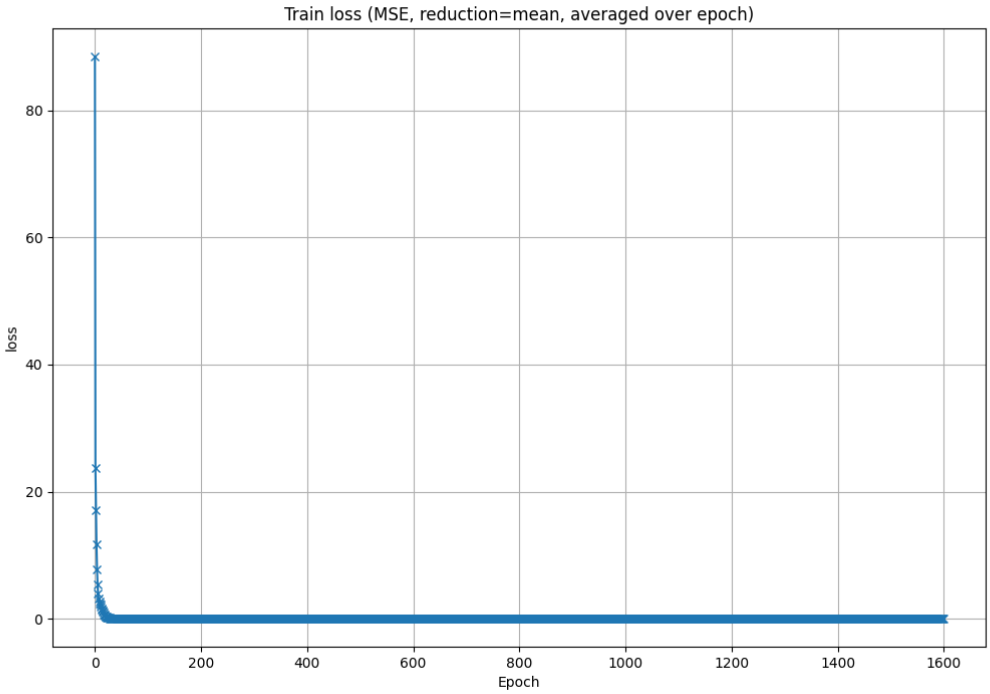

Figure 14: Loss convergence over epochs

Adjusting the batch size to 1 leads to a significant increase in training time. As shown in Figures 13 and 14, the model achieves comparable fitting accuracy compared to Figures 7 and 8 in Section 2.2 (under the same conditions, but with a batch size of 8). However, the computational cost increases significantly. Efficiency is also very important during training.

2.3 Introducing Noise

In this experiment, we added **Gaussian white noise** to the target values of the training data. The noise follows a normal distribution with a mean of 0 and a standard deviation of 10. This approach simulates the measurement errors or uncertainties that may exist in real-world scenarios, thereby increasing the randomness of the training data to enhance the model's robustness under noisy conditions.

Furthermore, the following hyperparameters are adjusted as follows:

- Number of samples (`samples`): 1000
- Number of neurons in the hidden layer (`hidden_dim`): 512
- Number of hidden layers: 3
- Number of training epochs (`epochs`): 5000
- Learning rate (`lr`): 0.001
- Batch size (`batch_size`): 8

The training results are shown in Figure 15 and Figure 16.

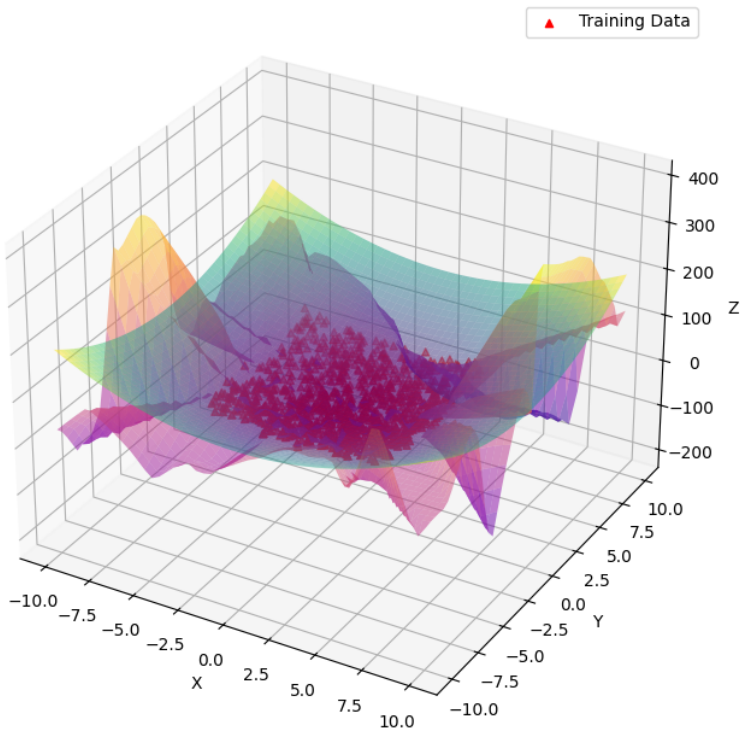

Figure 15: Fitted Surface

In Figure 15, the three-dimensional fitting surface exhibits overly complex oscillatory patterns, further confirming that the model has not only captured the global patterns of the training data but also overfitted to the random noise.

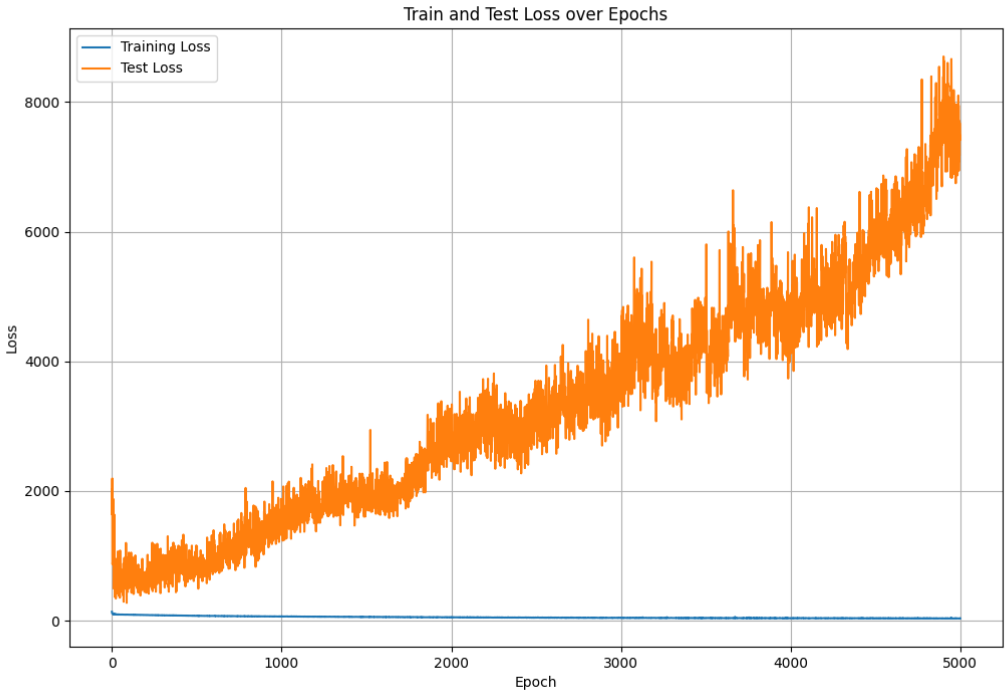

Figure 16: Train and Test Loss over Epochs

In Figure 16, the training loss curve (blue line) decreases rapidly in the initial stages and stabilizes at an extremely low value, indicating that the model fits the training data very well. However, the test loss curve (orange line) initially decreases but then rises significantly as the number of training epochs increases. This suggests that the model fails to generalize to the test data.

This phenomenon of divergence between training and test loss is known as **overfitting**, where the model performs exceptionally well on the training data but poorly on the test or unseen data, demonstrating weak generalization ability. The root cause of overfitting lies in the model's excessive focus on the details and noise in the training data, while neglecting the global patterns and general rules of the data.

Regularization, early stopping strategies, or appropriate reductions in model complexity are needed to alleviate the overfitting problem and improve the generalization ability of the model.

2.4 Regularization

To mitigate overfitting in the model, we implemented the following three methods:

2.4.1 Model Modifications

Adding Dropout Layers

Dropout is a regularization technique that randomly deactivates a portion of neurons during each training iteration. This breaks the dependencies between neurons, effectively preventing the model from relying too heavily on specific features. As a result, the model's robustness is enhanced, and the risk of overfitting is reduced.

We added `nn.Dropout` to the hidden layers with a dropout rate 0.2.

Simplifying the Hidden Layers

Overly complex models, such as those with a high number of neurons in the hidden layers, can lead to overfitting by memorizing training data, including its noise, instead of learning the global patterns. Simplifying the model by reducing the number of neurons or layers can decrease the model's capacity and help prevent overfitting.

We halved the number of neurons in the hidden layer and reduced the network to a single hidden layer.

2.4.2 Training Modifications

Adding L2 Regularization

L2 regularization adds a penalty term proportional to the square of the weights to the loss function, constraining the size of model weights. This prevents excessively large weights, reduces model complexity, and ensures smoother parameter distributions, thereby improving generalization.

We introduced the `weight_decay=1e-5` in the optimizer to control the strength of L2 regularization.

The training results are shown in Figure 17 and Figure 18.

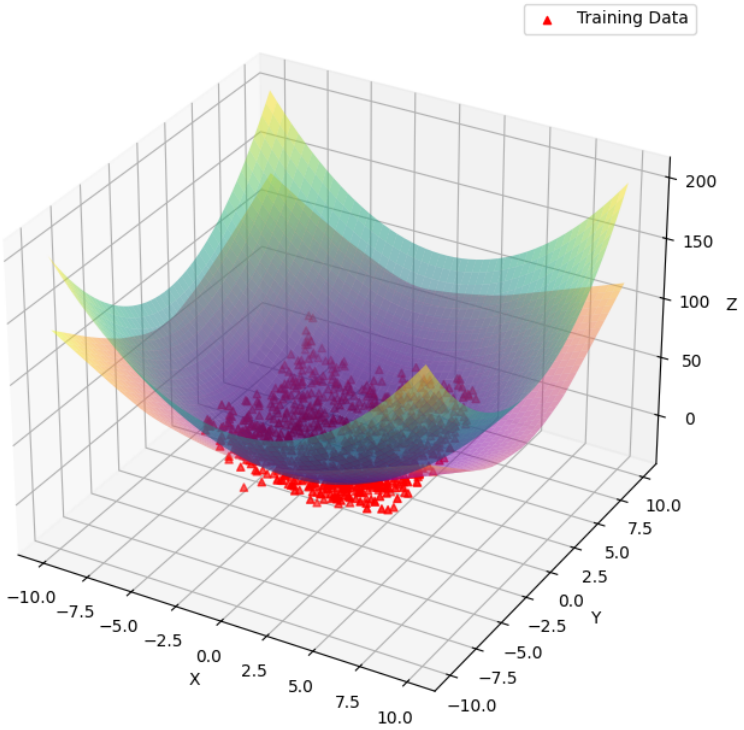

Figure 17: Fitted Surface

As shown in Figure 17, the fitted surface demonstrates that the regularized model effectively suppresses overfitting. Compared to the previous model, the learned surface aligns well with the target surface, indicating that overfitting has been significantly mitigated. Even though the training data (red triangles) still includes noise, but the learned surface appears robust to this, indicating the model's improved generalization ability.

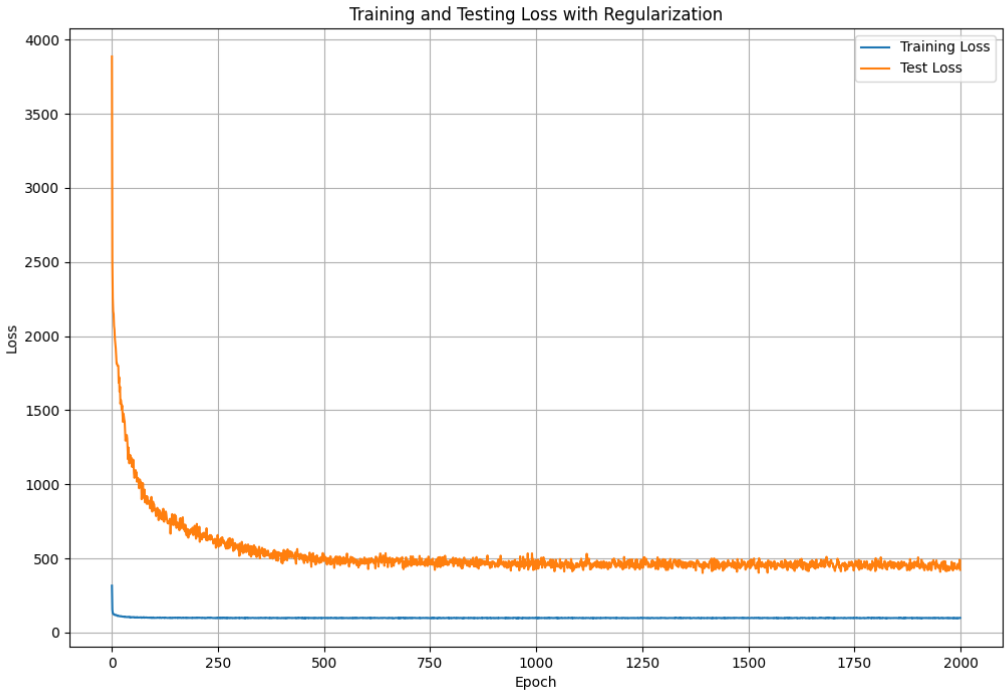

Figure 18: Train and Test Loss over Epochs

As shown in Figure 18, the loss curves further illustrate the effect of regularization. The training loss decreases rapidly and stabilizes at a relatively low value, which indicates that the model has efficiently learned the underlying function while avoiding overfitting.

Meanwhile, the testing loss also declines significantly during the early training stages and stabilizes at a value slightly higher than the training loss, showing good generalization without significant divergence between training and testing losses, which indicates its ability to generalize well to unseen data.

These observations demonstrate that regularization improves the model's robustness against noisy data, achieving a good balance between training and testing data. The model successfully learns the main patterns in the training data while avoiding overfitting to noise, resulting in superior fitting performance and enhanced generalization.

Task 2 Material modeling

1 General analysis

This task aims to implement a function for calculating the Helmholtz free energy density of a two-dimensional compressible Neo-Hookean material under a given displacement field. The input of the function is a set of coordinate points $\mathbf{X}$ in the reference configuration, and a displacement field $u(\mathbf{X})$ defined on the reference configuration. The output is the Helmholtz free energy value at each reference point.

The reference configuration is set to be a square domain $[0, 1] \times [0, 1]$. With `torch.linspace` and `torch.meshgrid`, a 50×50 uniform grid of points $\{(X_x, X_y)\}$ is generated in this region. Each grid point represents a coordinate in the reference configuration.

The displacement field $u(\mathbf{X})$ describes how each point in the material moves from the reference configuration to the current configuration. For any reference point $\mathbf{X} = (X_x, X_y)$, the displacement field can be generally expressed as:

$$u_x = u_x(X_x, X_y), \quad u_y = u_y(X_x, X_y) \tag{1}$$

u_x and u_y respectively represent the displacements in the x- and y-directions. With $u(\mathbf{X})$, we obtain the deformed position:

$$\mathbf{x} = \mathbf{X} + u(\mathbf{X}) \tag{2}$$

The deformation gradient $\mathbf{F}$ is the local linear mapping gradient tensor from the reference to the deformed configuration, describing strain and rotation at a given point. For a two-dimensional problem, $\mathbf{F}$ is a 2×2 matrix:

$$\mathbf{F} = \nabla_{\mathbf{X}}(\mathbf{X} + u) = \mathbf{I} + \nabla_{\mathbf{X}} u \tag{3}$$

specifically:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} \end{bmatrix} \tag{4}$$

$\mathbf{F}$ captures the deformation characteristics in an infinitesimal neighborhood. When $\mathbf{F}$ deviates from the identity matrix $\mathbf{I}$, it indicates stretching, compression, or shear at that location. With $\mathbf{F}$ known, we can compute the volume change and the Right Cauchy Green strain tensor.

$$J = \det(\mathbf{F}) \tag{5}$$

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} \tag{6}$$

which can be used to compute the invariant:

$$I_1 = \mathrm{tr}(\mathbf{C}) \tag{7}$$

Then, the Helmholtz free energy density at each reference point can be yielded by substituting these quantities into the compressible Neo-Hookean free energy formula:

$$\Psi = \frac{\lambda}{2}(\ln J)^2 - \mu \ln J + \frac{\mu}{2}(I_1 - 2) \tag{8}$$

with following Lamé constants as material parameters are given, $\mu = 384.614\,\text{Pa}, \lambda = 576.923\,\text{Pa}$.

In the following, linear, nonlinear, and compressive displacement fields will be introduced by changing the definition of the displacement field $u(\mathbf{X})$. This will allow a comparison of the differences in free energy distributions for different deformation modes.

2 Comparison of different displacement fields

2.1 Linear Displacement Field

The form of the linear displacement field is:

$$u_x = a \cdot X_x + b \cdot X_y, \quad u_y = c \cdot X_x + d \cdot X_y \tag{9}$$

This displacement field describes the deformation of the material under uniform stretching or shearing, where the parameters a, b, c, d control the magnitude of the stretching or shearing.

Here, we set $a = 2$, $b = 1$, $c = -1$ and $d = 3$, so the specific form of the displacement field $u(X)$ is:

$$u_x = 2X_x + X_y, \quad u_y = -X_x + 3X_y \tag{10}$$

The results are shown in Figure 19:

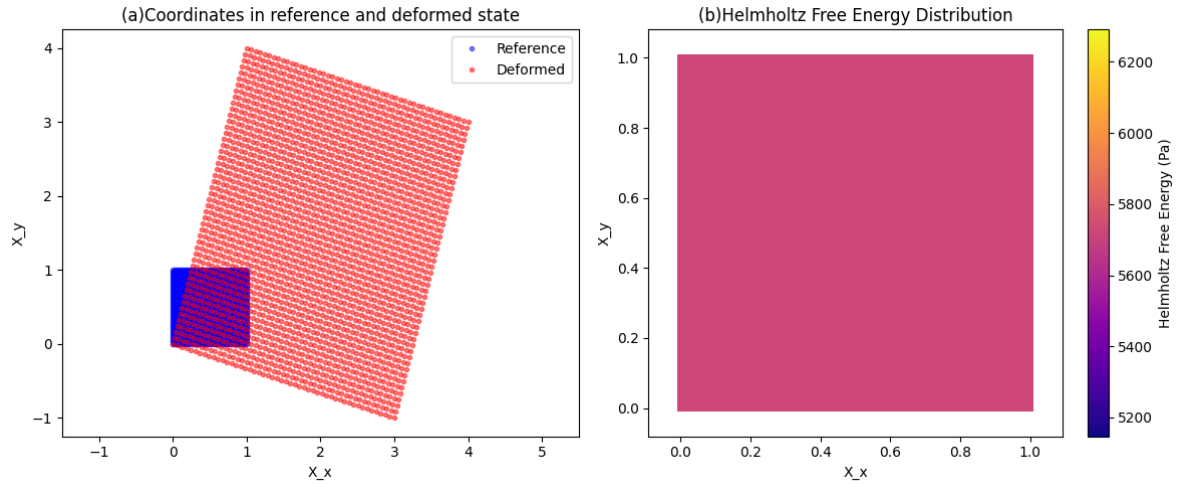

Figure 19: Results of the linear displacement field

- In Figure 19(a), the blue dots represent the reference configuration, and the red dots represent the deformed configuration. The parameters $a = 2, b = 1$ cause significant stretching in the x-direction, while $c = -1, d = 3$ result in a combination of shearing and stretching in the y-direction. This ultimately stretches the reference square domain into a parallelogram.
- Figure 19(b) shows the Helmholtz free energy mapped back to the grid of reference points X, illustrating the free energy density distribution generated by the deformation caused by the displacement field $u(X)$. From the figure, it can be observed that the free energy is nearly uniformly distributed across the entire domain, with no noticeable gradient variations.

This phenomenon occurs because the displacement field is linear, making the deformation gradient F a constant matrix. Consequently, the volumetric change J and the invariant I_1 remain constant throughout the domain, resulting in a uniform distribution of the free energy.

2.2 Nonlinear Displacement Field

The nonlinear displacement field $u(X)$ is defined as:

$$u_x = X_x^2 + X_y^2, \quad u_y = \sin(X_x) + X_y^3 \tag{11}$$

The results are shown in Figure 20:

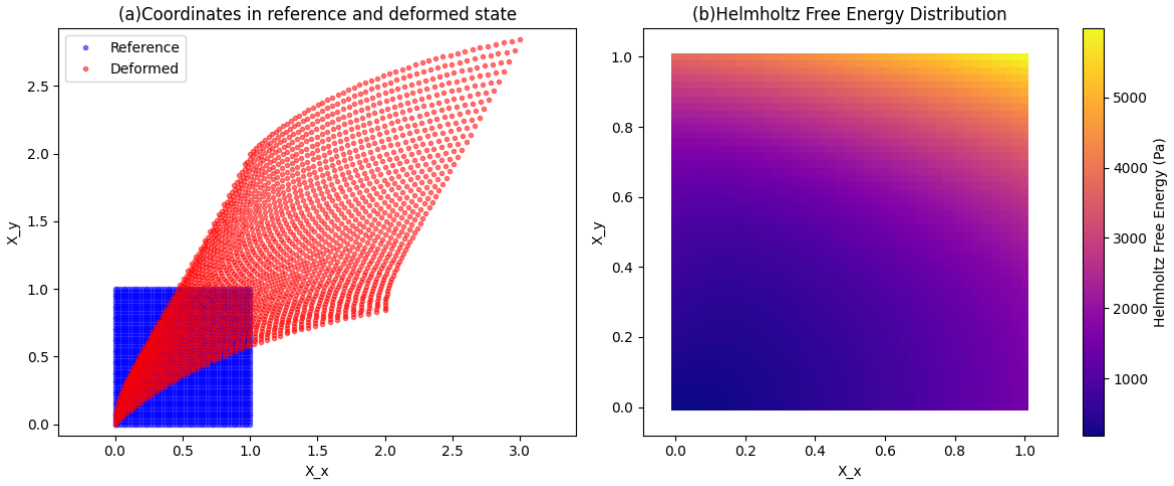

Figure 20: Results of nonlinear displacement field

- In Figure 20(a), the reference domain is stretched into a complex curved shape under the nonlinear displacement field, exhibiting significant nonlinear deformation characteristics.
- From Figure 20(b), it can be observed that the free energy values gradually increase from the lower-left corner to the upper-right corner of the reference domain. Furthermore, the increase along X_y is more pronounced than along X_x. This is because as X_y increases, the cubic term X_y^3 significantly contributes to u_y, resulting in a substantial increase in the local energy due to deformation.

2.3 Compressive Displacement Field

The compressible displacement field is defined as:

$$u_x = -0.5X_x + 0.25X_y^3, \quad u_y = -0.25X_x - 0.5X_y \tag{12}$$

The results are shown in Figure 21:

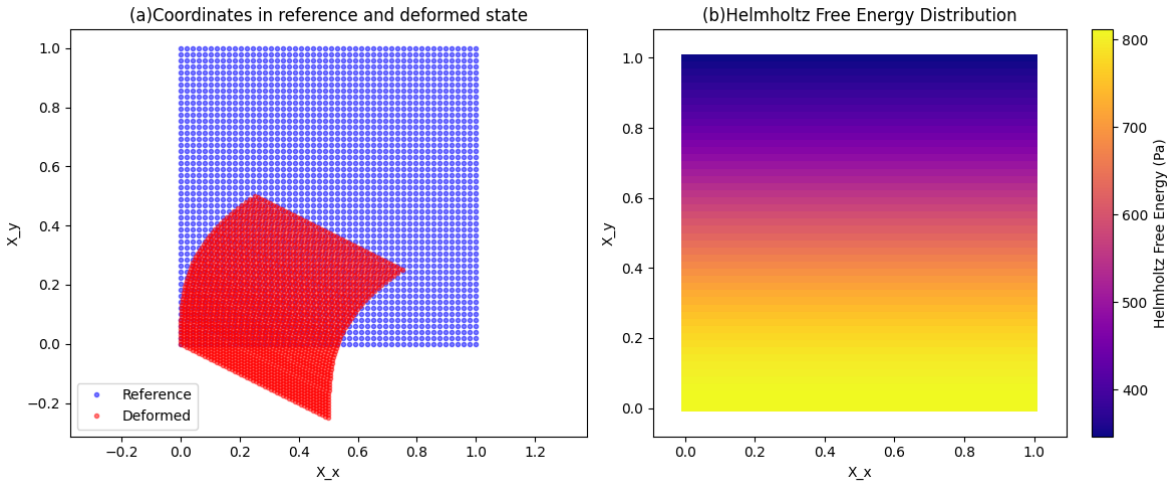

Figure 21: Results of compressible displacement field

- Figure 21(a) illustrates the deformed reference domain: the domain exhibits compression and bends towards the lower left. The right-hand region undergoes significant bending due to the nonlinear term X_y^3.
- Figure 21(b) shows the Helmholtz free energy density distribution, which decreases from the bottom to the top.

We then modify the coefficient of X_y^3 in the displacement field to 0.5. The modified compressible displacement field is defined as:

$$u_x = -0.5X_x + 0.5(X_y)^3, \quad u_y = -0.25X_x - 0.5X_y \tag{13}$$

The results are shown in Figure 22:

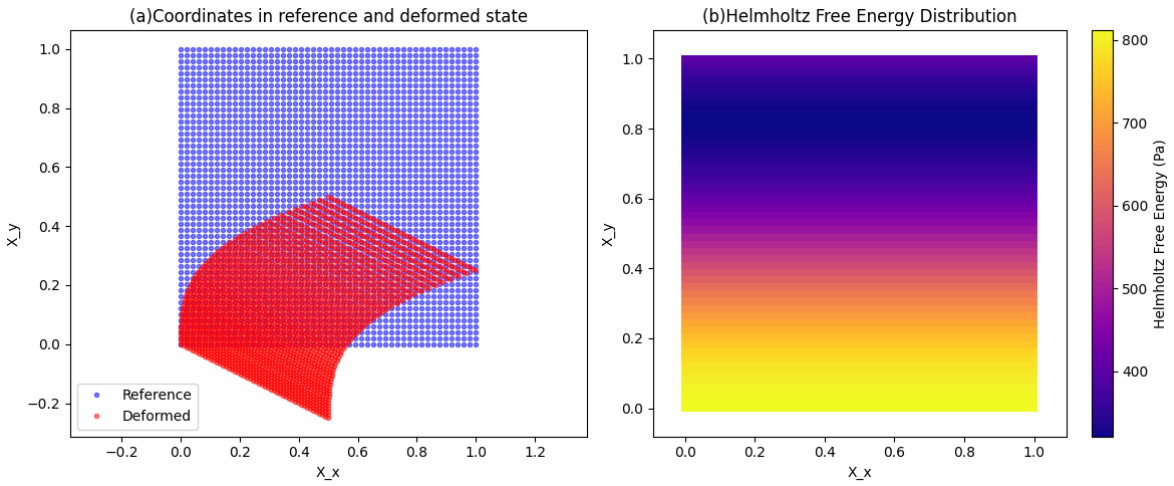

Figure 22: Results of new compressible displacement field

- Figure 22(a) demonstrates the deformed configuration after the modification. Compared to the previous case, the deformed region is noticeably stretched towards the upper right and exhibits stronger deformation, particularly in the upper regions. This is because the nonlinear term $0.5 \cdot (X_y)^3$ is enhanced, causing u_x to increase rapidly when X_y is large. Although the linear compression term $-0.5X_y$ is still present, the influence of nonlinear deformation becomes more significant in the upper regions.

- Figure 22(b) shows that the Helmholtz free energy density distribution still decreases from the bottom to the top. However, compared to Figure 21(b), the blue (low free energy) region in the upper part has expanded significantly, while the yellow (high free energy) region in the lower part has shrunk. This could be attributed to the fact that, although the off-diagonal components of the deformation gradient F increase in the upper regions, the direct impact on J is relatively small, resulting in the deformation being dominated more by shape changes rather than volume changes.

Task 3 CANN

The objective of this task is to employ the Constitutive Artificial Neural Network (CANN) model to approximate the free energy function Ψ of a Neo-Hookean material. This function depends on the deformation invariants I_1 and I_3. By constructing and training a neural network, the distribution of free energy under a nonlinear deformation field and its relationship with the invariants are captured. The model's accuracy is validated through error analysis and visualization.

Using the function `helmholtz_free_energy` obtained from the previous task, we choose a compressive displacement field. The displacement field function is identical to the previous formulas (12). Based on previous formulas (3) to (7) and using automatic differentiation tools, the deformation gradient F and the determinant of the volume deformation $J = \det(F)$ are computed. The third invariant is $I_3 = \det(C) = J^2$, and the first invariant is $I_1 = \mathrm{tr}(C)$. Substituting these invariants into the free energy formula yields the true free energy values.

The `generate_data` function is used to randomly sample training data (I_1, I_3, Ψ) within the region $[0, 1]^2$. The neural network structure implemented in the `CANN` class is no longer a simple fully connected network. The `CANN` structure is shown in Figure 23, which is inspired by Figure 5 in Reference [1]. The input values (I_1, I_3) are shifted to construct: $(I_1 - 3)$, $(I_1 - 3)^2$, $(I_3 - 1)$, $(I_3 - 1)^2$.

For each of these features, both the identity transformation and the exponential transformation $\exp(x) - 1$ are applied. This process yields a total of 8 features. Finally, a linear layer maps these 8 features to predict the value of Ψ.

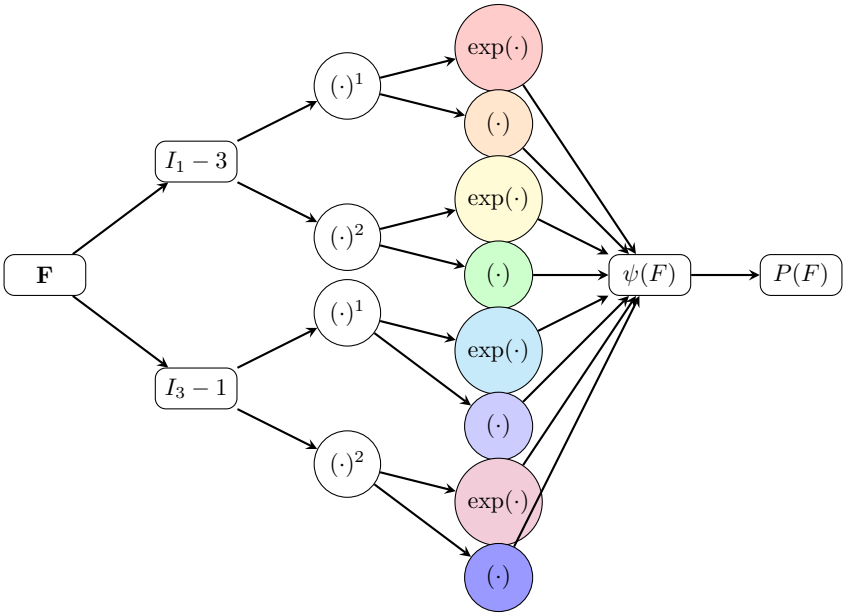

Figure 23: CANN Network Structure

During the training process, it was observed that direct training led to difficulties in loss function convergence. Therefore, both the inputs and target values were normalized to improve numerical stability. The mean squared error (MSE) was employed as the loss function, and the Adam optimizer was used to iteratively optimize the network parameters over 1000 training epochs. After training, the predicted free energy values were compared with the true free energy values for analysis.

To evaluate the accuracy of the model, the relative error is introduced and defined as:

$$\text{Relative Error} = \frac{\Psi_{\text{pred}} - \Psi_{\text{true}}}{|\Psi_{\text{true}}| + \epsilon} \tag{14}$$

where Ψ_{pred} is the predicted value, Ψ_{true} is the true value, and ϵ is a small positive number (e.g., 10^{-8}) to avoid division by zero. By visualizing and statistically analyzing the relative error, the predictive performance of the model can be examined more intuitively.

The output results are shown in Figures 24 and 25.

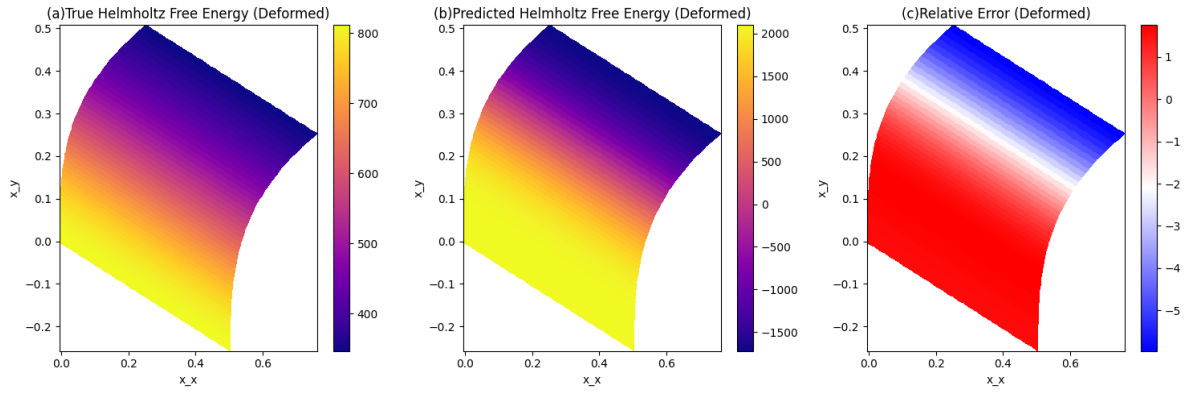

Figure 24: Comparison of true and predicted free energy distributions and relative error

From Figure 24, it can be seen that the free energy distribution predicted by the model has a certain similar trend with the real free energy distribution, both of which are higher in the lower free energy and lower the further to the X_y positive direction. However, the lower high free energy region is larger in the predicted free energy distribution. In addition, as shown by the relative error distribution, the model's prediction bias is larger in some regions, indicating that the complex nonlinear relationship between free energy and invariants has not yet been accurately fitted.

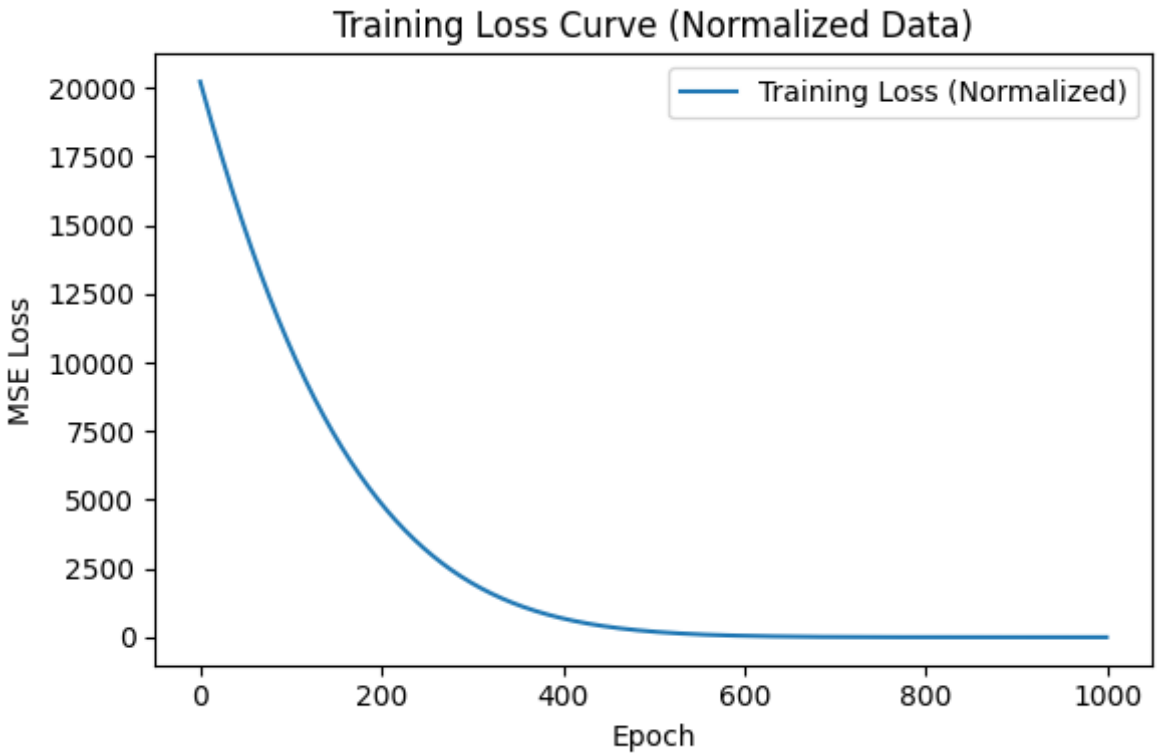

Figure 25: Training loss curve over 1000 epochs

Figure 25 shows that the training loss decreases significantly in the initial stage (about 16000) and levels off after about 400 epochs, which indicates that the model continuously learns features from the data during the training process and gradually approximates the mapping relationship of the target free energy function.

Reference

[1] Linka, K. and Kuhl, E: *A new family of Constitutive Artificial Neural Networks towards automated model discovery.* (Preprint) arXiv:2210.02202v2, 21 Oct 2022, https://doi.org/10.48550/arXiv.2210.02202