FEA Simulation Result Prediction using Physics Informed Neural Network

Prepared by : Arun Venkatesh T

AIML : Dissertation

- **Finite Element Analysis :** FEA is an essential computational tool in structural mechanics used to simulate and predict stress, strain, displacement for the given component under various loading and boundary conditions.

- **Problem with Traditional FEA:**
  - Despite the proven accuracy of traditional Finite Element Analysis, its inherent computational cost and time consumption pose a significant bottleneck,
  - There is a clear need for a more efficient and rapid computational method capable of predicting FEA results with acceptable accuracy, thereby reducing the dependency on traditional solvers.

- **Proposed Solution:**
  - This dissertation project aims to address this challenge by developing a Neural Network framework designed to predict FEA simulation results for 3D models in linear elastic regions

AIML : Dissertation

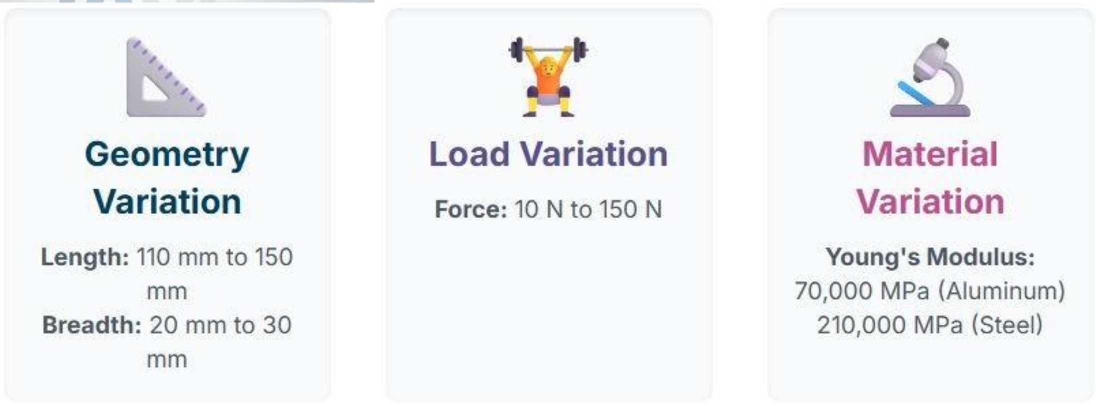
- The development of the AI model consists of five steps,as illustrated in the flowchart.

Step 1 : Dataset Creation

- **Data Generation:** Training a neural network for FEA result prediction begins with creating a diverse and representative dataset.

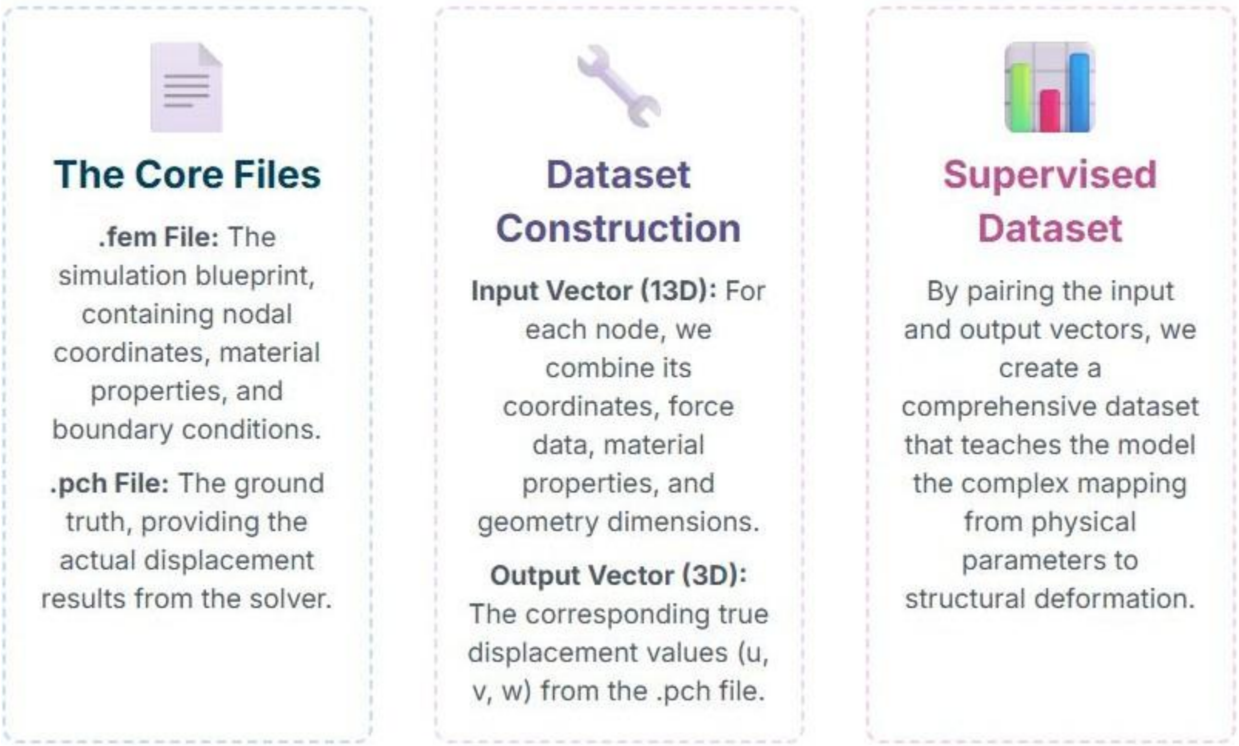
- **Base Geometry & Variations**: A cantilever beam was selected as the primary geometry for initial experimentation, with multiple simulation cases produced by varying key parameters.

- **Robustness & Generalization:** The dataset spans a wide range of simulated scenarios to ensure the model can generalize well and make accurate predictions under varied conditions.



Step 1 : Dataset Creation

- **Supervised Dataset Creation:** Nodal coordinates (x, y, z) from the .fem file are used as input features, and their corresponding displacement components (ux, uy, uz) from the .pch file serve as output labels, forming a supervised learning dataset.



Step 1 : Dataset Creation

Simulation deck file ".fem" format

```
$$  GRID Data
$$
GRID          1        0.0     0.0     0.0
GRID          2        0.0     20.0    0.0
GRID          59       5.0     20.0    0.0
```

Simulation Result file ".pch" format

```
$TITLE    =   (OS 2021.2)
$SUBTITLE=
$LABEL    = PLATE_1
$DISPLACEMENTS
$REAL OUTPUT
$SUBCASE ID =          1
         1       G      0.000000E+00      0.000000E+00      0.000000E+00
-CONT-                  0.000000E+00      0.000000E+00      0.000000E+00
         2       G      0.000000E+00      0.000000E+00      0.000000E+00
-CONT-                  0.000000E+00      0.000000E+00      0.000000E+00
         3       G      0.000000E+00      0.000000E+00      0.000000E+00
-CONT-                  0.000000E+00      0.000000E+00      0.000000E+00
         4       G      0.000000E+00      0.000000E+00      0.000000E+00
```


Example of dataset formation

```
Parsing FEM file: O:\AIML\SEMESTER_4\LINEAR_PREDICTION\data\fem\CANTI_TIT_20N.fem
Parsed 345 nodes
Parsed 176 3D elements (CHEXA/CPENTA)
Parsed 1 materials
Parsed 45 boundary conditions
Parsed 1 loads
Loading displacement data from: O:\AIML\SEMESTER_4\LINEAR_PREDICTION\data\pch\CANTI_TIT_20N.pch
Extracted displacements for 345 nodes.
Built dataset with 39300 node-samples total
```

Step 2 : Neural Network Model

- **Regression Model :** The primary purpose of this neural network is regression, specifically to predict the displacement (u, v, w components) of individual nodes within a mechanical structure.
- **PINN :** This model incorporates physical laws (Navier-Cauchy equations for linear elasticity) into its loss function, making it a Physics-Informed Neural Network.
- The core of the neural network is a Multi-Layer Perceptron (MLP), also known as a fully connected neural network, The network consists of the following layers

Input Layer

Takes a **13-dimensional input** vector with features for geometry, load, and material properties.

Hidden Layers

8 hidden layers, each with **128 neurons** and a **Tanh** activation function.

Output Layer

Produces a **3-dimensional output** vector for the predicted displacements (u, v, w).

Step 2 : Neural Network Model

- **Data Normalization :** Feature standardization is performed to stabilize training and accelerate convergence:
- Mean (in_mean) and standard deviation (in_std) are computed for each input feature using the training set.
- Target values are standardized using out_mean and out_std from the training set.
- All normalization parameters are stored and reused for both test evaluation and unseen inference.

- **Trainable parameters :**

Layer	Formula	Parameters
Input to first hidden layer	13 * 128 + 128	1792
Six middle hidden layer	6 * (128 * 128 + 128)	99072
Last hidden layer to output	128 * 3 + 3	387

Total Trainable Parameters

101,251

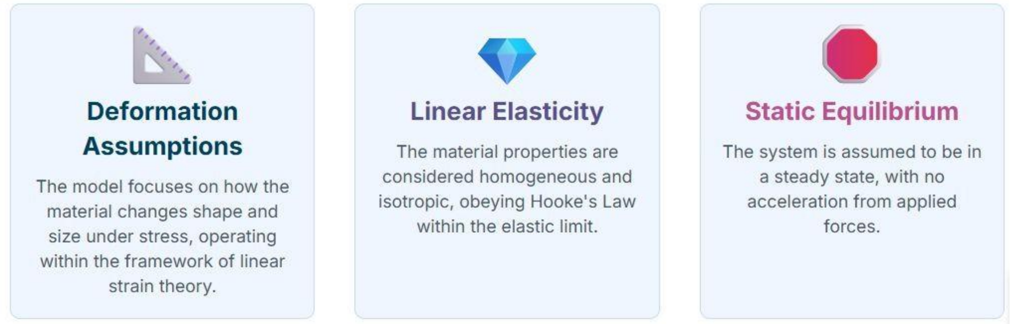
Step 3 : Physics Loss Integration

- **Total Loss :** The framework uses a hybrid loss combining MSE data loss, which measures squared differences between predicted and actual displacements from .pch files to fit training data, and a physics-informed residual loss that enforces the Navier-Cauchy equations of linear elasticity.

- **Navier-Cauchy equations :** The Navier-Cauchy equation is a set of partial differential equations (PDEs) that govern the equilibrium of a deformable solid in the context of linear elasticity

- They are the mathematical expression of Newton's Second Law of Motion applied to continuous materials, relating a material's displacement to the forces acting upon it.

- They describe the relationship between displacement in a solid and the applied forces or stresses, under the assumptions of:



Step 3 : Physics Loss Integration

- PINN Loss Function Workflow :

  This flowchart illustrates the step-by-step process of calculating the total loss of the AI model.

  Navier-Cauchy equation :

  For a displacement vector **u(x, y, z) = (u, v, w)**, the **static, isotropic form** is:

$$\mu \nabla^2 \mathbf{u} + (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) + \mathbf{f} = 0$$

  Where:

  - **λ** and **μ** → Lamé's constants (material properties derived from E and ν)
  - **E** → Young's modulus
  - **ν** → Poisson's ratio
  - **f** → Body force vector (force per unit volume, e.g., gravity)

- The goal of the model is to minimize this loss, forcing the residual to approach zero and thereby satisfying the physical equilibrium equation

Step 4 : Prediction and Export

- **Optimizer:** The Adam optimizer is used to update the model's weights during training. A learning rate of 1e-3 with no weight decay is used.

- **Evaluation Method:** After each training epoch, Test Metrics are Computed:
  - MAE (Mean Absolute Error)
  - RMSE (Root Mean Squared Error)

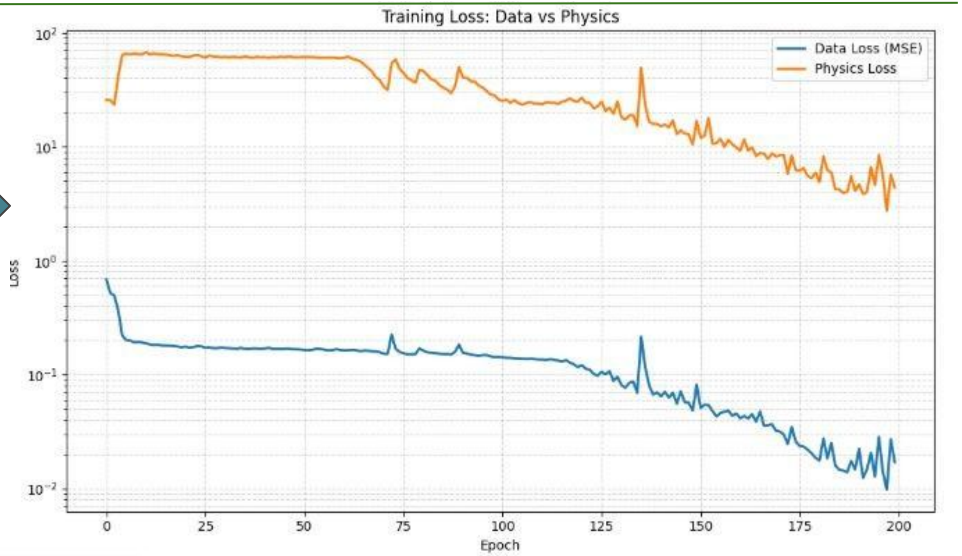
- Logging is performed for every 5 epochs, showing:
  - Data Loss
  - Physics Loss
  - MAE
  - RMSE

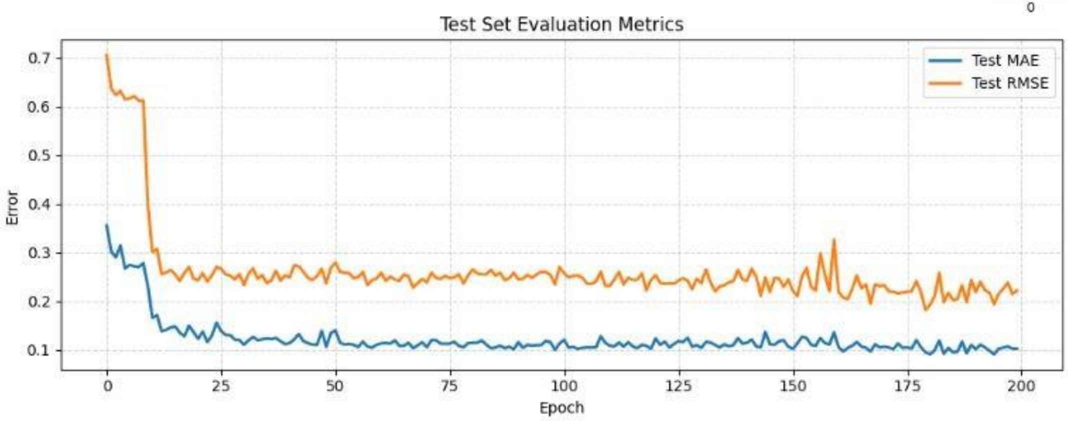
Training Parameters

Parameter	Value
Epochs	200
Batch size	1024
Split ratio	80/20 (Train/Test)

Step 4 : Prediction and Export

❖ Data Loss and Physics Loss on Training data



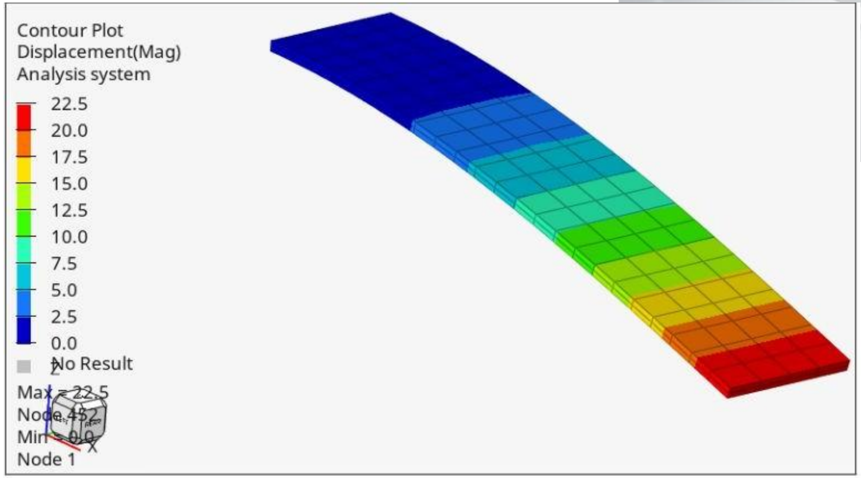


❖ MAE and RMSE trends on the test set

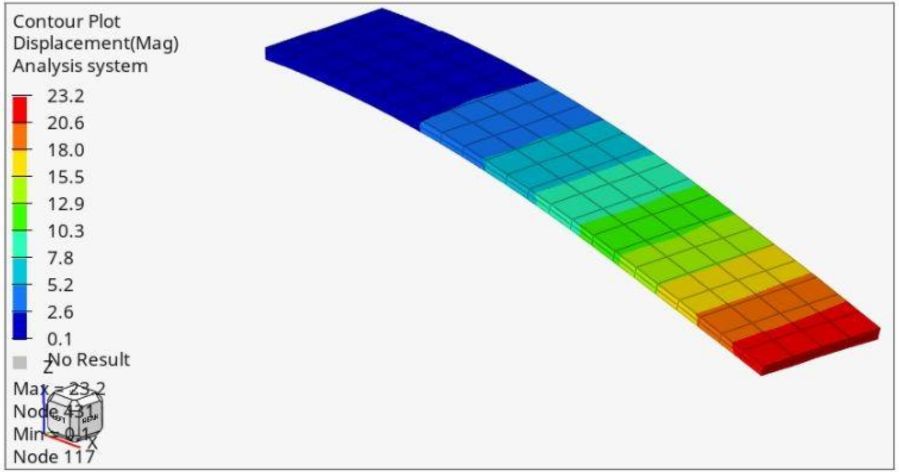
Step 4 : Prediction and Export

- **Result Visualization :** A key objective of this project is to ensure that the AI-driven predictions are not merely numerical outputs but can be readily integrated into standard engineering workflows, enabling comprehensive analysis and interpretation by engineers.
- **HDF5 Format :** HDF5 is a widely adopted, self-describing, binary data format. Its robust structure is designed to store and manage the large, complex datasets typical of engineering simulations, making it the ideal choice for exporting the predicted displacements.

Results generated from commercial software



Results generated from AI model.



Step 4 : Prediction and Export

- **Result Verification :** To ensure the model is robust and trustworthy in practical applications, its predictions are also compared directly with the results from a validated commercial FEA solver (e.g., OptiStruct).

- This post-training verification process involves:

- Running the same FEM input file through the commercial solver to obtain the displacement field.

- Feeding the identical FEM file into the trained prediction model via the deployed API.

- Exporting the model's predictions in .h5 format and visualizing them in HyperView.

- Comparing displacement magnitudes and directional components (u, v, w) between the two sources

Step 4 : Prediction and Export

In-Range Performance

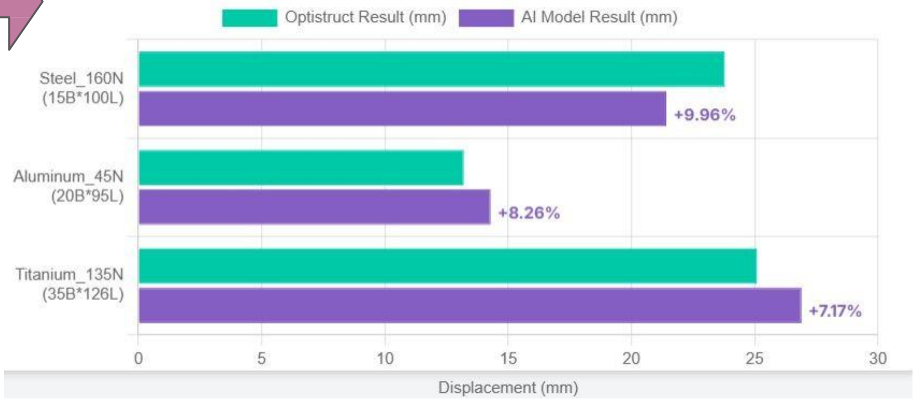
The model shows close agreement with the OptiStruct results for test cases within the training range of geometry, load, and material.

2.57% to 5.78%

Error Range

This indicates that the model generalizes well to new cases similar to the data it was trained on.

Out of range performance



Out-of-Range Performance

When tested on cases with +/- 15% variation from the training range, the model still delivers reasonable accuracy.

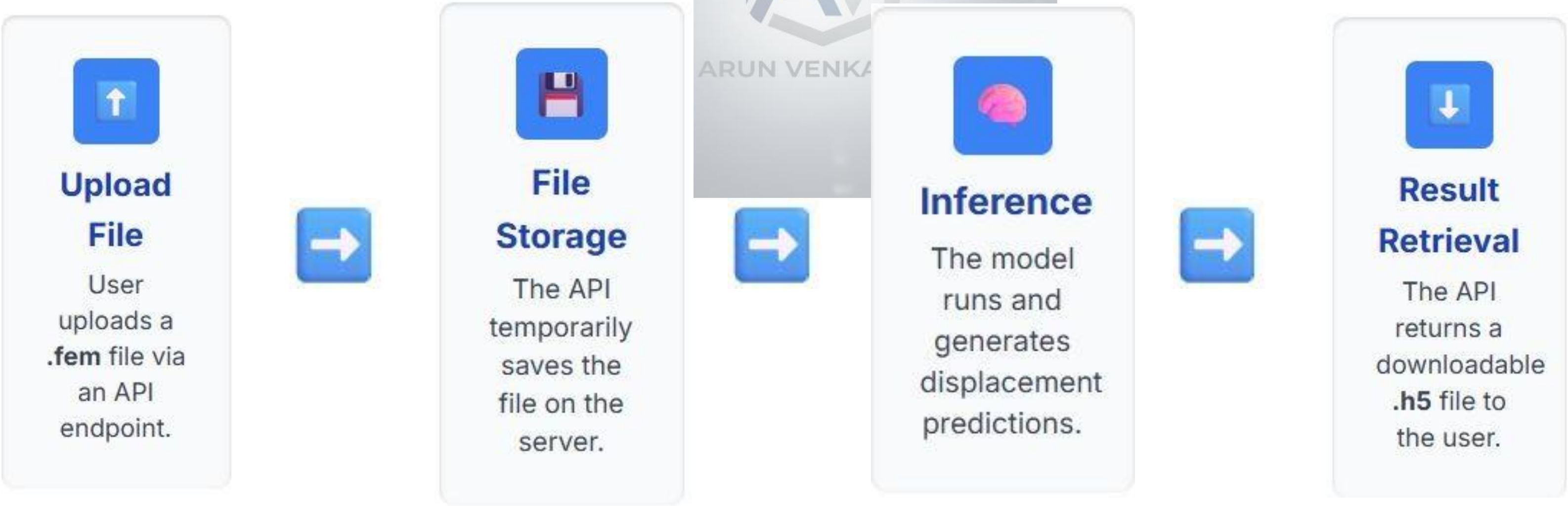
7.17% to 9.96%

Error Range

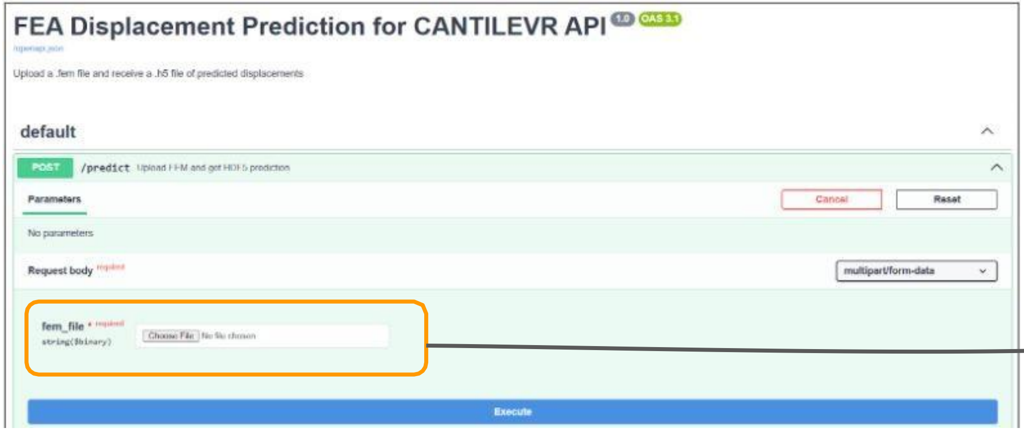
This demonstrates the model's ability to extrapolate beyond its training domain while maintaining acceptable engineering accuracy.

Step 5 : Model Deployment

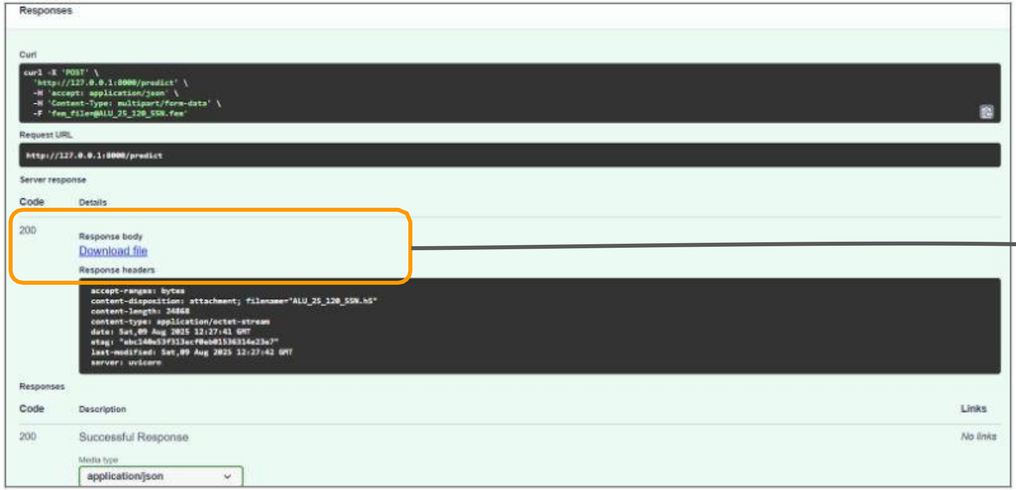
- **Deployment :** The trained model was deployed as a RESTful API using the FastAPI framework in Python. This approach allows the model to run on a server, where it can be accessed by various clients (e.g., web applications, scripts, or other engineering software).
- The API is served using Uvicorn, a fast ASGI server. This setup allows the model to be accessed from any machine on the network
- The key steps of the API are as follows:



Step 5 : Model Deployment



❖ Interactive API for testing, show the option to upload .fem file



❖ In response, we can download the predicted displacement result in .h5 format

Conclusion and Future Work

❖ **Conclusion :** This dissertation presented the design, development, and validation of a Physics-Informed Neural Network framework for predicting FEA results in 3D models.

A Complete End-to-End Pipeline

A key achievement was the successful implementation of a full data and model pipeline, bridging the gap between traditional CAE inputs and AI-driven predictions.

1. **Automated Data Preparation**

   Automated processing of OptiStruct **.fem** and **.pch** files to create supervised datasets.

2. **Model Development**

   A multi-layer perceptron (MLP) based PINN with a hybrid loss function that combines data and physics constraints.

3. **HDF5 Export**

   Implementation of an HDF5 export module for seamless visualization in Altair HyperView.

4. **RESTful API Deployment**

   Deployment as a FastAPI REST service for practical, on-demand inference.

Robust Performance

In-Range Error	Out-of-Range Error
2.57% - 5.78%	7.17% - 9.96%

The model demonstrates **high accuracy** for in-domain data and **robust extrapolation** for out-of-domain conditions.

Overall, the framework bridges the gap between AI-driven prediction models and traditional CAE workflows.

Future Work

Non-Linear Models & Complex Geometries

The current model is limited to linear elastic behavior and simple cantilever beams. A key next step is to expand the model's capabilities to handle more advanced scenarios.

- **Non-Linear Materials:** Incorporate material models like **plasticity** into the PINN by updating the governing partial differential equations (PDEs) within the loss function.
- **Complex Geometries:** Expand the training dataset to include a wider variety of 3D geometries, enhancing the model's ability to generalize to intricate, real-world components.

Future Work

Advanced Prediction & Assembly-Level Simulation

To provide a more complete analysis, the model's output can be expanded beyond displacement to handle more complex engineering scenarios.

Extension to Stress and Strain

Expand the output layer to predict the six independent components of both the strain tensor (ϵ) and the stress tensor (σ).

Assembly-Level Predictions

Extend the framework to model interactions between multiple parts, such as contact, friction, and bolted connections.

- **Closing comments :** the proposed framework demonstrates the potential to evolve into a scalable, solver-independent, and computationally efficient simulation tool and in certain contexts even serve as a partial substitute—ultimately delivering significant value to industrial product development cycles

Thank You!

For your attention