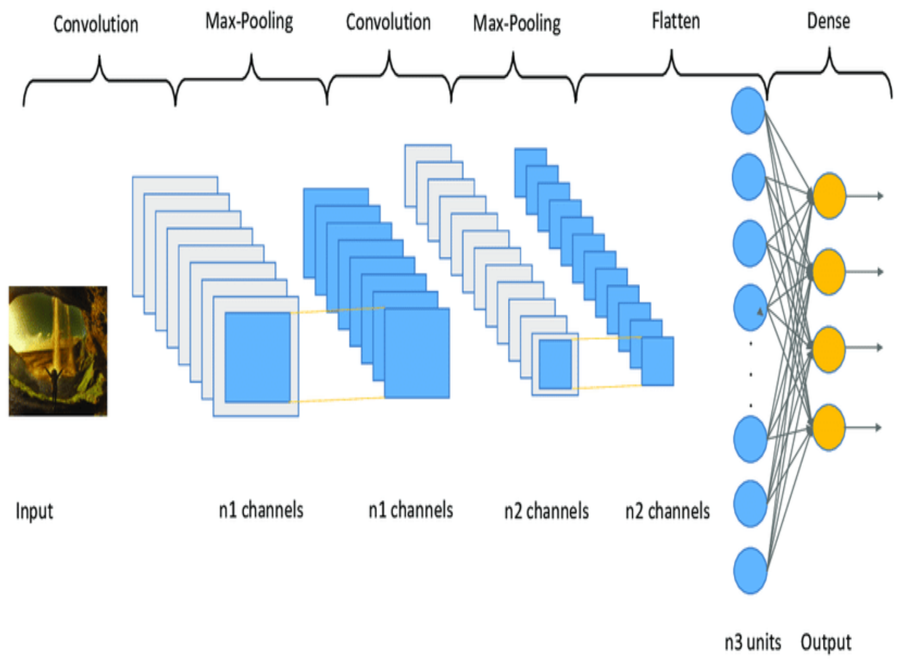


ME-504

STRUCTURAL OPTIMIZATION WITH DEEP LEARNING

RAHUL YADAV-2022MEB1334

SUMER BASSI-2022MEB1351

PROBLEM STATEMENT:

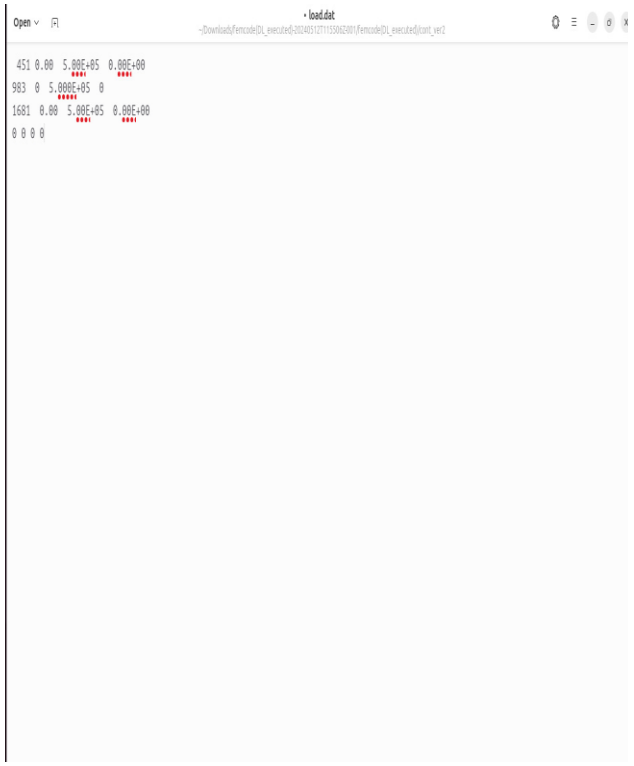
Developing a deep learning solution to predict maximum displacement in cantilever beams, utilizing randomly generated configurations and convolutional neural networks and using active learning to improve its performance.

- ## Introduction:

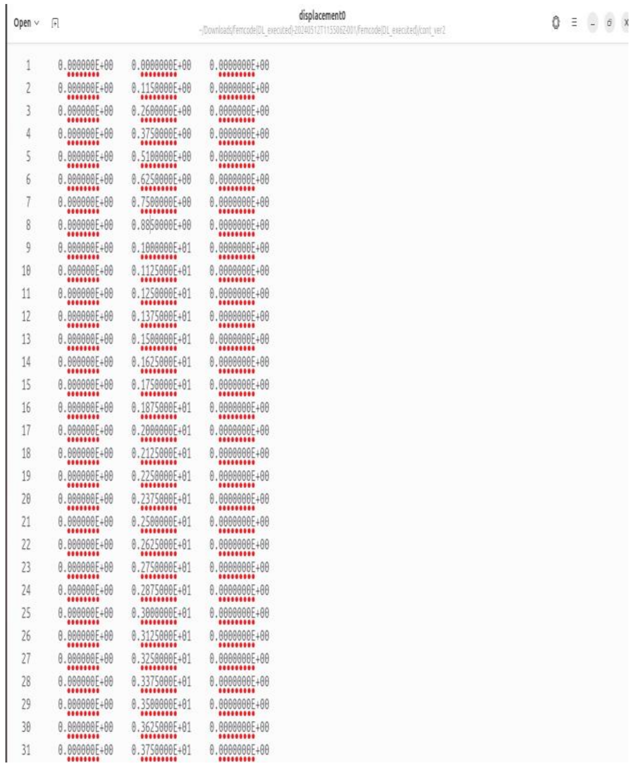
The problem statement outlines the task of predicting the maximum displacement of a cantilever beam given specific configurations. This task is important in various engineering applications, as it helps in understanding the structural behavior and performance of materials under different conditions. In this project, we aim to develop a deep learning model using convolutional neural networks (CNNs) to accurately predict the maximum displacement. The following project roadmap outlines the steps we will take to achieve this goal."

PROJECT ROADMAP:

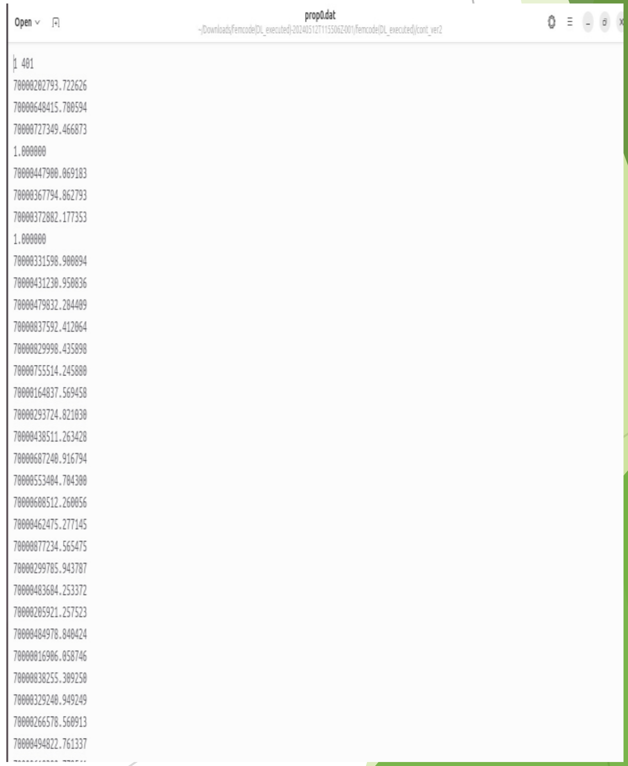
1. **Data Preparation:**
   - Prepare Labelled Dataset


Open
• load.dat
~/Downloads/femcode(DL_executed)-20240512T115506Z-001/femcode(DL_executed)/cont_ver2

```
 451 0.00  5.00E+05  0.00E+00
983  0  5.000E+05  0
1681  0.00  5.00E+05  0.00E+00
0 0 0 0
```

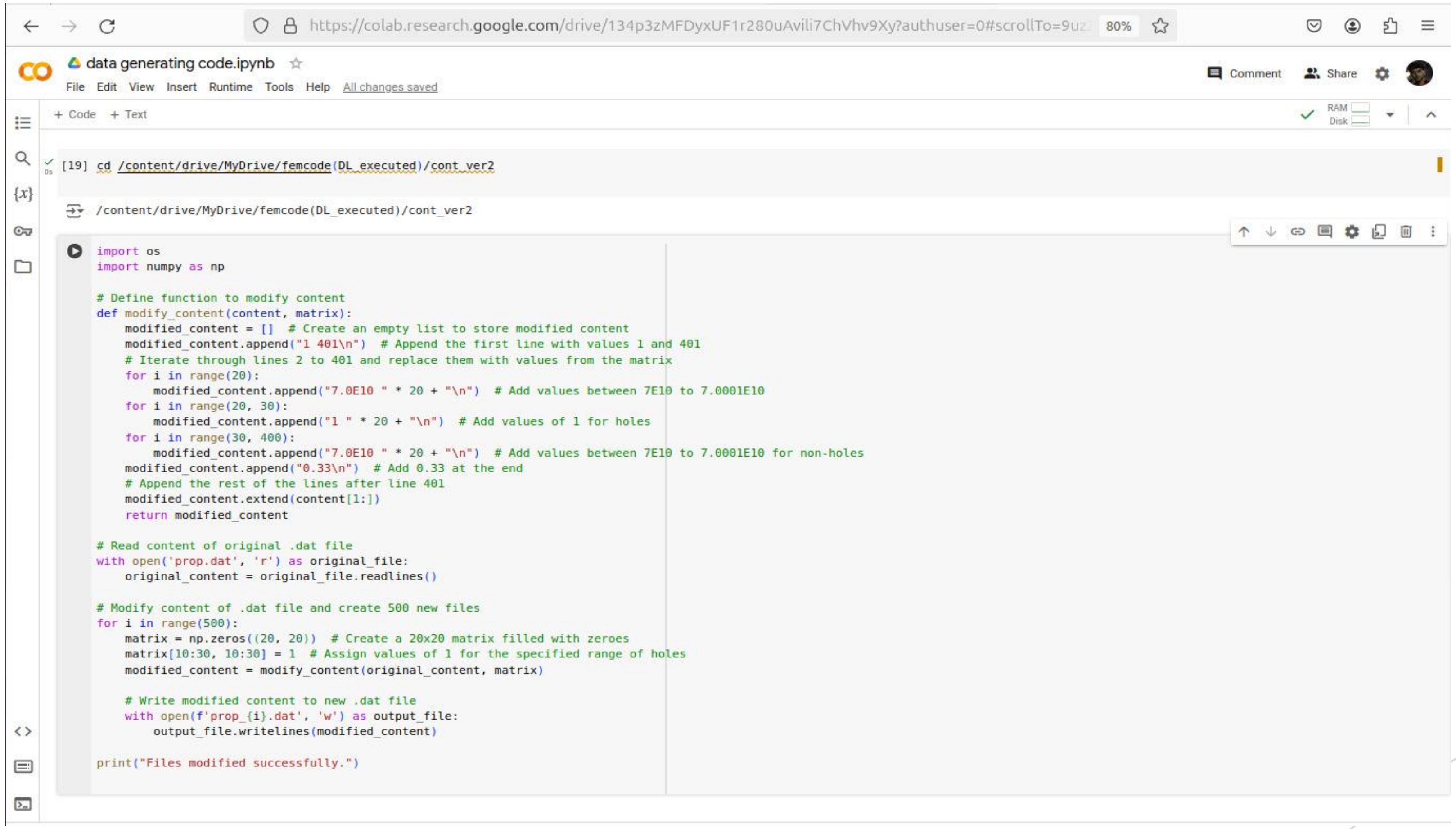

Open
displacement0
~/Downloads/femcode(DL_executed)-20240512T115506Z-001/femcode(DL_executed)/cont_ver2

```
  1    0.000000E+00   0.0000000E+00   0.0000000E+00
  2    0.000000E+00   0.1150000E+00   0.0000000E+00
  3    0.000000E+00   0.2600000E+00   0.0000000E+00
  4    0.000000E+00   0.3750000E+00   0.0000000E+00
  5    0.000000E+00   0.5100000E+00   0.0000000E+00
  6    0.000000E+00   0.6250000E+00   0.0000000E+00
  7    0.000000E+00   0.7500000E+00   0.0000000E+00
  8    0.000000E+00   0.8850000E+00   0.0000000E+00
  9    0.000000E+00   0.1000000E+01   0.0000000E+00
 10    0.000000E+00   0.1125000E+01   0.0000000E+00
 11    0.000000E+00   0.1250000E+01   0.0000000E+00
 12    0.000000E+00   0.1375000E+01   0.0000000E+00
 13    0.000000E+00   0.1500000E+01   0.0000000E+00
 14    0.000000E+00   0.1625000E+01   0.0000000E+00
 15    0.000000E+00   0.1750000E+01   0.0000000E+00
 16    0.000000E+00   0.1875000E+01   0.0000000E+00
 17    0.000000E+00   0.2000000E+01   0.0000000E+00
 18    0.000000E+00   0.2125000E+01   0.0000000E+00
 19    0.000000E+00   0.2250000E+01   0.0000000E+00
 20    0.000000E+00   0.2375000E+01   0.0000000E+00
 21    0.000000E+00   0.2500000E+01   0.0000000E+00
 22    0.000000E+00   0.2625000E+01   0.0000000E+00
 23    0.000000E+00   0.2750000E+01   0.0000000E+00
 24    0.000000E+00   0.2875000E+01   0.0000000E+00
 25    0.000000E+00   0.3000000E+01   0.0000000E+00
 26    0.000000E+00   0.3125000E+01   0.0000000E+00
 27    0.000000E+00   0.3250000E+01   0.0000000E+00
 28    0.000000E+00   0.3375000E+01   0.0000000E+00
 29    0.000000E+00   0.3500000E+01   0.0000000E+00
 30    0.000000E+00   0.3625000E+01   0.0000000E+00
 31    0.000000E+00   0.3750000E+01   0.0000000E+00
```


Open
prop0.dat
~/Downloads/femcode(DL_executed)-20240512T115506Z-001/femcode(DL_executed)/cont_ver2

```
1 401
70000202793.722626
70000648415.780594
70000727349.466873
1.000000
70000447980.069183
70000367794.862793
70000372882.177353
1.000000
70000331598.900894
70000431230.950836
70000479832.284409
70000837592.412064
70000829998.435898
70000755514.245880
70000164837.569458
70000293724.821030
70000438511.263428
70000687240.916794
70000553404.704300
70000608512.260056
70000462475.277145
70000877234.565475
70000299785.943787
70000483684.253372
70000205921.257523
70000484978.840424
70000016986.058746
70000838255.309250
70000329240.949249
70000266578.560913
70000494822.761337
```

Combining Configuration Files


https://colab.research.google.com/drive/134p3zMFDyxUF1r280uAvili7ChVhv9Xy?authuser=0#scrollTo=9uz
80%
data generating code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

```
[19] cd /content/drive/MyDrive/femcode(DL_executed)/cont_ver2
```

```
/content/drive/MyDrive/femcode(DL_executed)/cont_ver2
```

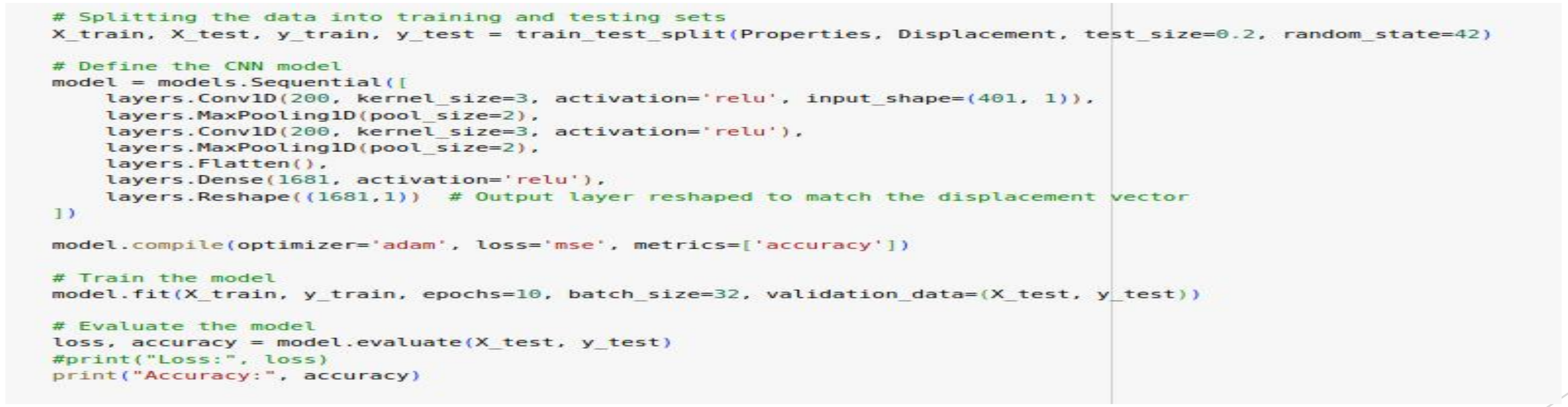
```python
import os
import numpy as np

# Define function to modify content
def modify_content(content, matrix):
    modified_content = []  # Create an empty list to store modified content
    modified_content.append("1 401\n")  # Append the first line with values 1 and 401
    # Iterate through lines 2 to 401 and replace them with values from the matrix
    for i in range(20):
        modified_content.append("7.0E10 " * 20 + "\n")  # Add values between 7E10 to 7.0001E10
    for i in range(20, 30):
        modified_content.append("1 " * 20 + "\n")  # Add values of 1 for holes
    for i in range(30, 400):
        modified_content.append("7.0E10 " * 20 + "\n")  # Add values between 7E10 to 7.0001E10 for non-holes
    modified_content.append("0.33\n")  # Add 0.33 at the end
    # Append the rest of the lines after line 401
    modified_content.extend(content[1:])
    return modified_content

# Read content of original .dat file
with open('prop.dat', 'r') as original_file:
    original_content = original_file.readlines()

# Modify content of .dat file and create 500 new files
for i in range(500):
    matrix = np.zeros((20, 20))  # Create a 20x20 matrix filled with zeroes
    matrix[10:30, 10:30] = 1  # Assign values of 1 for the specified range of holes
    modified_content = modify_content(original_content, matrix)

    # Write modified content to new .dat file
    with open(f'prop_{i}.dat', 'w') as output_file:
        output_file.writelines(modified_content)

print("Files modified successfully.")
```

2. CNN Model Building:

- Implementation

3. Training

- Train the CNN model using the labeled dataset.



```python
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(Properties, Displacement, test_size=0.2, random_state=42)

# Define the CNN model
model = models.Sequential([
    layers.Conv1D(200, kernel_size=3, activation='relu', input_shape=(401, 1)),
    layers.MaxPooling1D(pool_size=2),
    layers.Conv1D(200, kernel_size=3, activation='relu'),
    layers.MaxPooling1D(pool_size=2),
    layers.Flatten(),
    layers.Dense(1681, activation='relu'),
    layers.Reshape((1681,1))  # Output layer reshaped to match the displacement vector
])

model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
#print("Loss:", loss)
print("Accuracy:", accuracy)
```

- Incorporate active learning techniques to iteratively select the most informative samples for labeling, thus optimizing the training process.

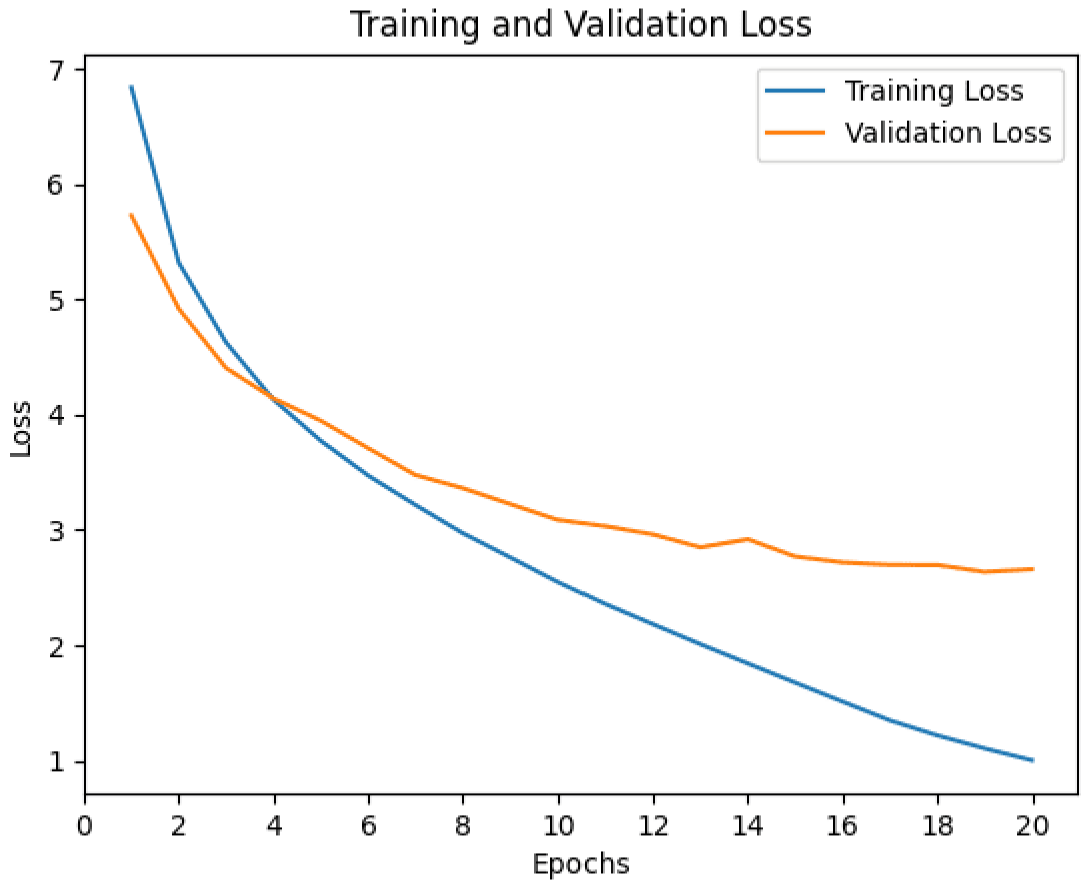
WHAT IS ACTIVE LEARNING?

Active learning is a machine learning approach where a model iteratively selects the most informative data samples to be labeled by an oracle (human annotator or existing labeled data) for training. Instead of passively using randomly selected data for training, active learning algorithms actively query the data points that are expected to provide the most learning value, thereby reducing the annotation effort and improving the efficiency of the learning process.

ALGORITHM:

1. Start with a small set of labeled data.
2. Train a model with this data.
3. Use the model to predict labels for unlabeled data.
4. Select the most uncertain or informative data points for labeling.
5. Label these data points and add them to the labeled set.
6. Retrain the model with the updated labeled data.
7. Repeat steps 3-6 until a stopping criterion is met.

3. **Evaluation**
4. **Prediction**



DIFFICULTIES FACED:

1. **Data Generation**:
   1. Explanation: Generating realistic data for training a CNN can be challenging, especially when it involves creating multiple configurations with variations in material properties, load distributions, and displacement patterns.
   2. Solution: We used random data generation techniques to simulate different configurations of the cantilever beam. By defining appropriate ranges for material properties, load magnitudes, and displacement patterns, we ensured that the generated data covered a diverse range of scenarios.
2. **Combining Data from Different Files**:
   1. Explanation: Integrating data from separate files (prop.dat, load.dat, displacement files) into a unified format suitable for CNN input can be complex, especially when the data structures and formats differ.
   2. Solution: We developed a script to read data from each file, apply any necessary transformations or adjustments, and combine them into a single input file for the CNN. By carefully managing the data conversion process, we ensured consistency and compatibility across all configurations.
3. **Segmentation Fault while Generating Displacement Files**:
   1. Explanation: A segmentation fault typically occurs due to memory access violations, often caused by issues such as accessing uninitialized memory or exceeding memory limits.
   2. Solution: We carefully reviewed the code responsible for generating displacement files and identified potential memory-related issues such as buffer overflows or incorrect memory allocations. By debugging the code and implementing appropriate error-checking mechanisms, we resolved the segmentation fault issue and ensured stable data generation.