CP302 Project:
Machine Learning Aided Modelling of Granular Materials

Ananya Bansal (2022CHB1041)
Supervisor: Dr. Saikat Roy

Indian Institute of Technology, Ropar

May 14, 2025

CONTENTS

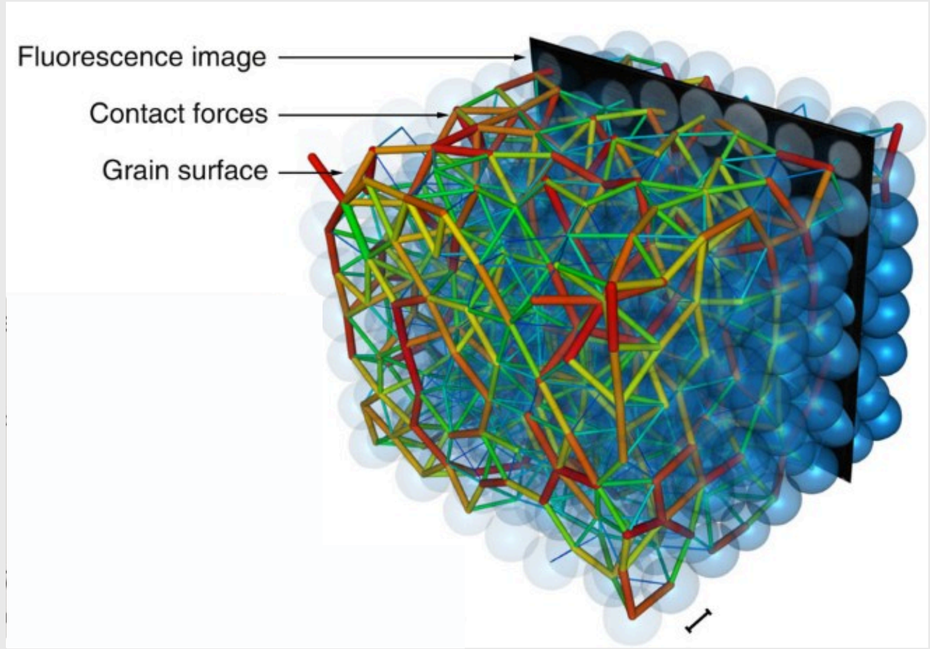
OVERVIEW

What are Granular Materials?

Materials composed of discrete particles (e.g., sand, powders).

- They consist of densely packed solid particles, sometimes within a surrounding gas or liquid, forming a multiphase system.
- Found in industries like pharmaceuticals, mining, and construction.

Understanding **force transmission** is crucial for applications in engineering and manufacturing.



Traditional simulation methods are computationally expensive. So we explore the possibility of doing better with data driven methods.

PROBLEM STATEMENT & OBJECTIVES

The Problem	Predicting normal contact forces in granular assemblies under uniaxial compression.
Current Challenge	Complete DEM simulations require significant computational resources.
Objective	Develop a Graph Neural Network (GNN) model for efficient and accurate force prediction, based on the work of Cheng and Wang (2022).
Additional Research	Explore how this can be applied to colloidal systems.

BRIEF RECAP TO GNN

A graph neural network (GNN) is a type of artificial neural network that processes graph-structured data, ideal for systems with relational dependencies.

- A graph is represented as G = (V, E)
- **V (Nodes):** Represent entities
- **E (Edges):** Represent relationship
- **Basic Working of GNNs:**
  - Encode-process-decode architecture
  - Message passing between connected nodes
  - Iterative updates refine predictions

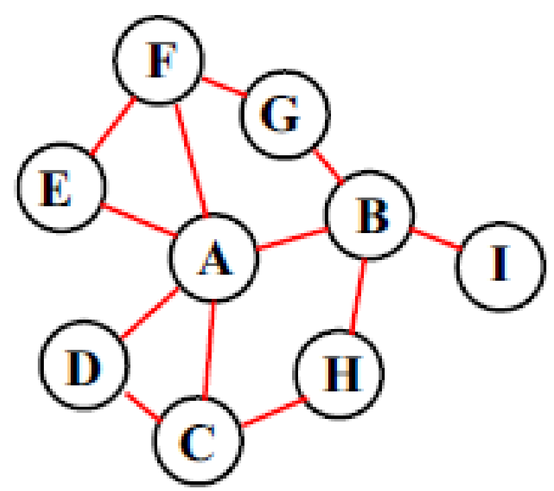


Fig 1: A basic graphical structure

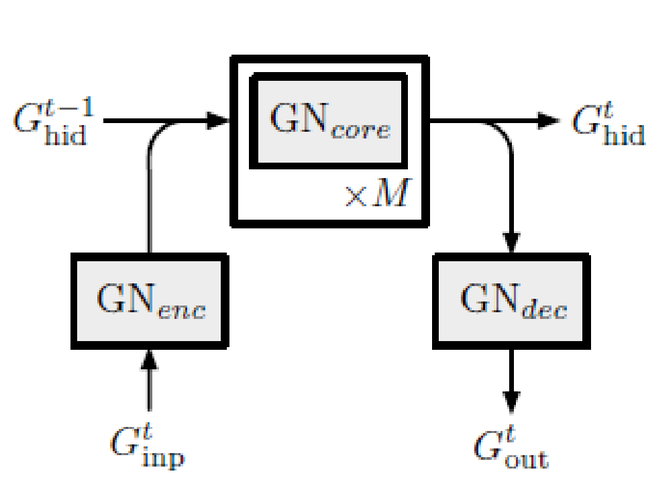


Fig 2: Recurrent GN Encode-process-decode Architecture

BRIEF RECAP TO GNN

WHY GNNS FOR GRANULAR MATERIALS?

- Granular materials can be naturally represented as graphs.
  - Nodes = **Particle properties** (size, velocity, position)
  - Edges = **Contact properties** (normal & tangential force, distance, relative velocities)
- GNNs capture both **local and global force interactions** efficiently.
- Unlike traditional ML, GNNs **preserve spatial relationships** in the data.

HOW DO GNNS WORK?

- **Message Function:** Nodes send messages to neighbors based on their features, neighbor features, and edge features.
- **Aggregation Function:** Each node collects and combines messages from all its neighbors.
- **Update Function:** Nodes update their features using previous state and aggregated messages.

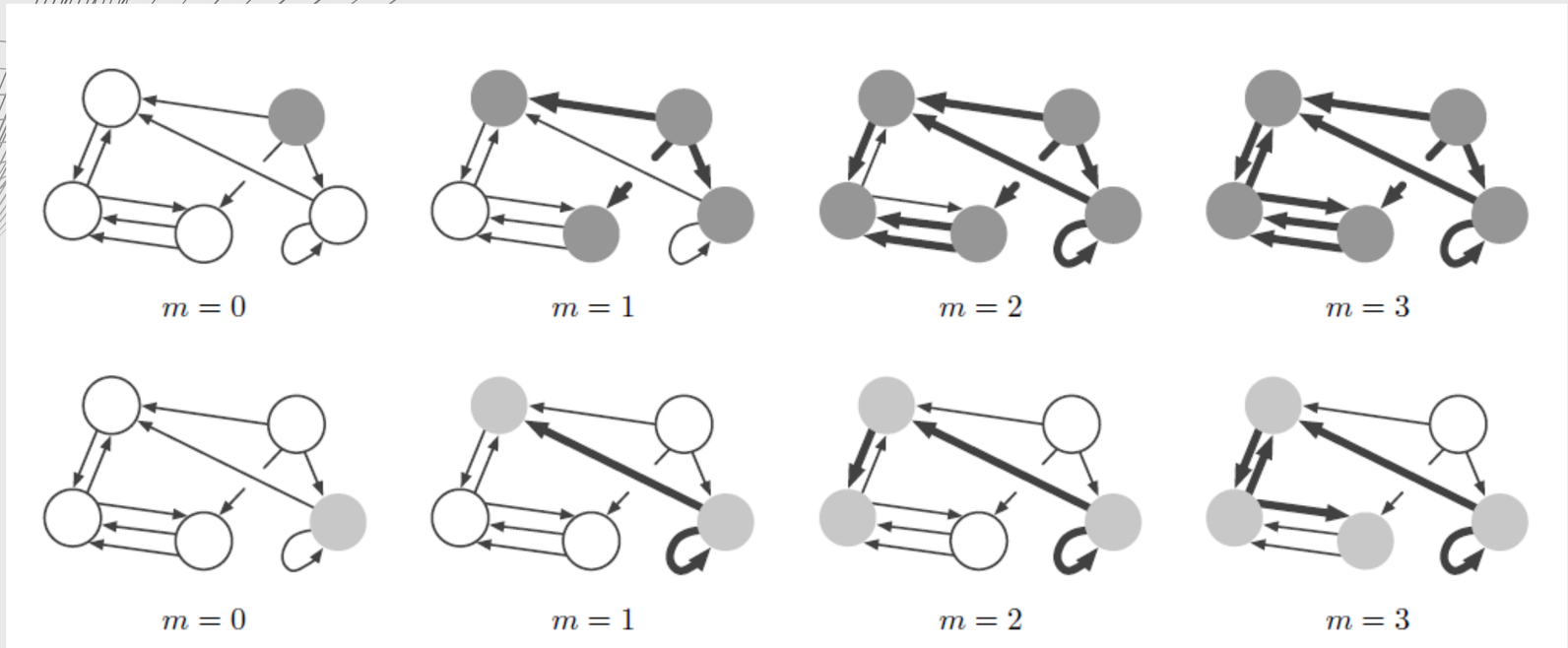
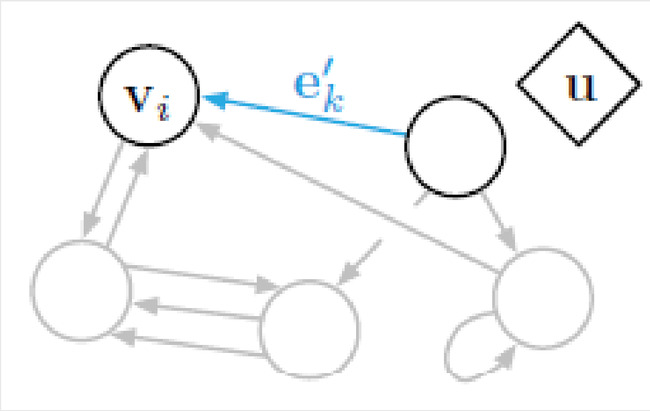
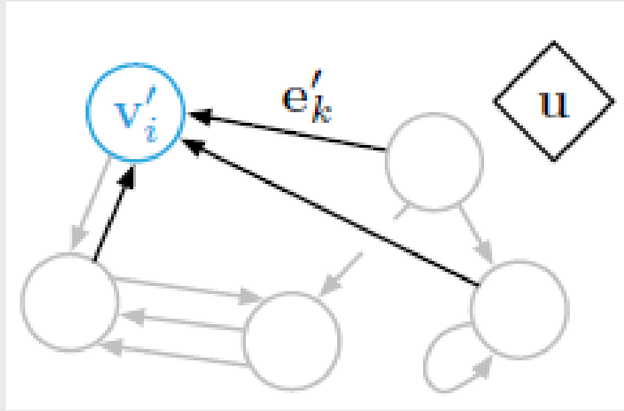


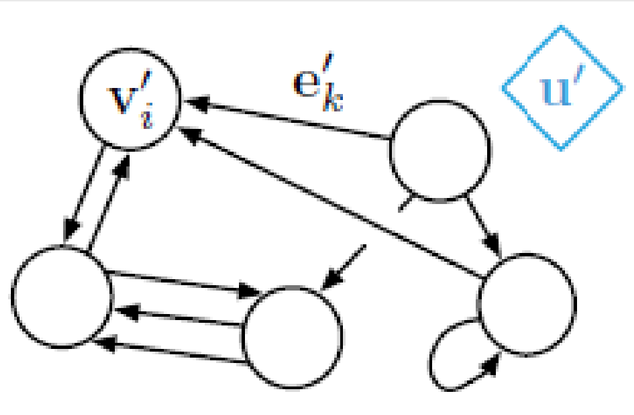
Fig 3: Example of message passing



(a) Edge Update



(b) Node Update



(c) Global Function Update

Fig 4: Updates in a GN block

PRE-MIDSEM WORK & LEARNINGS

Initial Approach:

- Data: Single granular assembly
- Model trained on a subset of particles within the assembly (70%) and the remaining particles (30%) used for model validation.
- Initial data did not take into account velocity vectors; assumed a static configuration.

Model Performance:

- Loss: 9.67
- R2 Score: 0.588
- MSE: 11.25

Limitations:

- Data lacked generalizability
- Limited dataset size for training & testing.
- Static assumption ignored particle dynamics.

Proposed Improvements:

- Use DEM data from multiple assemblies.
- Incorporate velocity vectors and dynamics.
- Train on 6 assemblies, test on 4.

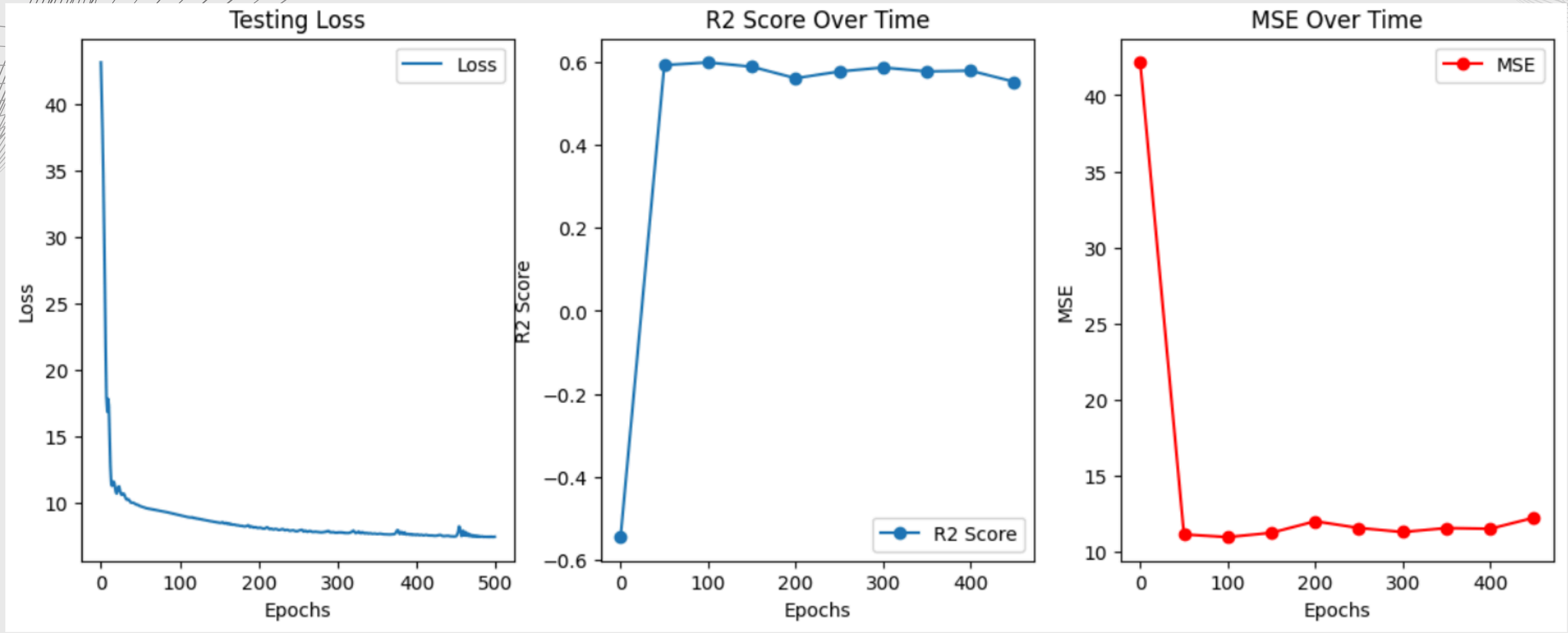


Fig 5: Model Performance over time

Results obtained were not favorable, so a different approach was worked upon and discussed for work after the midsems.

POST-MIDSEM IMPLEMENTATION

I. Data Procurement & Pre-processing

Data Procurement:

- 10 granular assemblies, each with ~4000 particles.
- ~200 node and edge files per assembly (~4000 files total).
- Data size: ~10 million data points.

Pre-processing:

- Converted 4000 files to CSV format & cleaned the data.
- Replaced displacement with corrected values in all node files.
- Added relative distances and velocities to edge files, by mapping through node files.
- Removed unnecessary headers & text.

DATA PREPROCESSING

```
\\data_pairf_disp\\data_pairf_disp\\conf10\\disp\\23500000.displace
✅ All files (.data, .displace, .pairf) processed and saved in: C:\Users\UIET\Desktop\Ananya\IIT\Sem 6\BTP\post midsem data\data_pairf_disp\data_pairf_d
isp\processed_data
```

```
            # Replace displacement columns in .data with corrected ones
            df_data["c_10[1]"] = df_disp["corrected_dx"]
            df_data["c_10[2]"] = df_disp["corrected_dy"]

            # --- Read .pairf file ---
            pairf_path = os.path.join(pairf_folder, f"{step}.pairf")
            df_pairf = pd.read_csv(pairf_path, sep='\s+', names=pairf_columns, on_bad_lines='skip')

            # --- Save both as CSV ---
            df_data.to_csv(os.path.join(output_dir, f"{conf}_{step}_nodes.csv"), index=False)
            df_pairf.to_csv(os.path.join(output_dir, f"{conf}_{step}_edges.csv"), index=False)

        except Exception as e:
            print(f"⚠️ Error processing {conf}, step {step}: {e}")

print(f"✅ All files processed and saved in: {output_dir}")
```

```
✅ All files processed and saved in: C:\Users\UIET\Desktop\Ananya\IIT\Sem 6\BTP\post midsem data\data_pairf\processed_data
```

```
ITEM: TIMESTEP
23400000
ITEM: NUMBER OF ATOMS
4000
ITEM: BOX BOUNDS pp ff pp
0.0000000000000000e+00 1.5000000000000000e+02
0.0000000000000000e+00 4.0000000000000000e+01
-5.0000000000000000e-01 5.0000000000000000e-01
ITEM: ATOMS id type diameter x y c_10[1] c_10[2] vx vy c_1 omegaz
```

Fig 6: Data preprocessing & unnecessary text

II. First Approach

Approach:

- Used all 10^7 data points across 10 configurations.
- Training using 6 assemblies, testing using 4.
- Incorporated **particle dynamics**, including velocity vectors.

Challenges faced:

- Variable edge file sizes caused reading issues.
  Solution found: Implemented a code to get data size for all edges files and run the model iteratively.
- Large data size led to model crashes → Model failed to train effectively due to computational constraints.
  Solution found: Shift to a single strain step approach to reduce data size.

III. Second Approach: Single Strain Step

Approach:

- Selected step value 32,700,000 for realistic strain/stress.
- This effectively reduced data to ~100,000 points → more manageable.
- GNN model trained on 6 assemblies, tested on 4.

	A	B	C	D	E	F	G	H
1	Step Value	strain	Sigmayy_1	Sigmayy_2	Sigmayy_3	Sigmayy_4	Sigmayy_5	
95	32700000	33.53846	301.3347	573.0749	304.6477	318.2667	820.6369	

Fig 7: Corresponding strain & stress values for 5 configurations

Preliminary Results:

- Train Loss decreased from 219.26 to 197.90 over 100 epochs.
- Subpar performance (as shown in Fig 8) that needed to be improved significantly.

```
Train Metrics: {'L1': 199.70047, 'L2': 306.6432, 'MSE': 94030.05, 'R2': 0.3016948098618706, 'Pearson': 0.5960169731446764}
Test Metrics: {'L1': 294.75748, 'L2': 444.13477, 'MSE': 197255.7, 'R2': 0.19475749475613924, 'Pearson': 0.5799930113916326}
```

Fig 8: Training and Testing metrics for inital model

SECOND APPROACH & IMPROVEMENTS

Steps suggested to improve model performance:

- Normalized features (scaled to 0-1) to align with contact force magnitude.
- Increased number of epochs for better convergence.
- Used Adam optimizer (as per Cheng and Wang, 2022).
- Tuned hyperparameters (e.g. hidden layers).
- Trained on 5 assemblies, validated on 1, and used 4 for testing for most optimal tuning.

```python
num_epochs = 100
train_losses = []

for epoch in range(num_epochs):
    loss = train(model, train_loader)
    train_losses.append(loss)
    if epoch % 10 == 0:
        print(f"Epoch {epoch} | Train Loss: {loss:.4f}")
```

```
Epoch 0 | Train Loss: 219.2582
Epoch 10 | Train Loss: 209.0618
Epoch 20 | Train Loss: 205.7987
Epoch 30 | Train Loss: 200.9814
Epoch 40 | Train Loss: 199.7545
Epoch 50 | Train Loss: 199.1387
Epoch 60 | Train Loss: 198.3401
Epoch 70 | Train Loss: 197.9677
Epoch 80 | Train Loss: 198.0829
Epoch 90 | Train Loss: 197.8968
```

Fig 9: Loss over 100 epochs

IV. Improvements & Final Results

All of the changes mentioned in the previous slide were incorporated into the final model, leading to a drastic improvement in the results, showing successful implementation of the GNN model.

Final Results:
Train Loss: 0.0019 | Val Loss: 0.0000
Pearson Correlation Coefficient: 0.999
Strong agreement with DEM simulations

Significance:

- Near-perfect accuracy validates GNN for force prediction.
- Reduced computational load compared to DEM.

```
Epoch 0 | Train Loss: 0.8010 | Val Loss: 0.4048
Epoch 10 | Train Loss: 0.0461 | Val Loss: 0.0305
Epoch 20 | Train Loss: 0.0115 | Val Loss: 0.0018
Epoch 30 | Train Loss: 0.0062 | Val Loss: 0.0008
Epoch 40 | Train Loss: 0.0064 | Val Loss: 0.0021
Epoch 50 | Train Loss: 0.0038 | Val Loss: 0.0004
Epoch 60 | Train Loss: 0.0031 | Val Loss: 0.0002
Epoch 70 | Train Loss: 0.0030 | Val Loss: 0.0001
Epoch 80 | Train Loss: 0.0028 | Val Loss: 0.0002
Epoch 90 | Train Loss: 0.0025 | Val Loss: 0.0001
Epoch 100 | Train Loss: 0.0024 | Val Loss: 0.0002
Epoch 110 | Train Loss: 0.0023 | Val Loss: 0.0000
Epoch 120 | Train Loss: 0.0021 | Val Loss: 0.0000
Epoch 130 | Train Loss: 0.0021 | Val Loss: 0.0000
Epoch 140 | Train Loss: 0.0021 | Val Loss: 0.0000
Epoch 150 | Train Loss: 0.0019 | Val Loss: 0.0000
Epoch 160 | Train Loss: 0.0020 | Val Loss: 0.0000
Epoch 170 | Train Loss: 0.0020 | Val Loss: 0.0000
Epoch 180 | Train Loss: 0.0019 | Val Loss: 0.0000
Epoch 190 | Train Loss: 0.0020 | Val Loss: 0.0000
Epoch 200 | Train Loss: 0.0019 | Val Loss: 0.0000
Epoch 210 | Train Loss: 0.0020 | Val Loss: 0.0000
Epoch 220 | Train Loss: 0.0019 | Val Loss: 0.0000
Epoch 230 | Train Loss: 0.0018 | Val Loss: 0.0000
Epoch 240 | Train Loss: 0.0019 | Val Loss: 0.0000
```

Fig 10: Loss over 250 epochs

FINAL MODEL

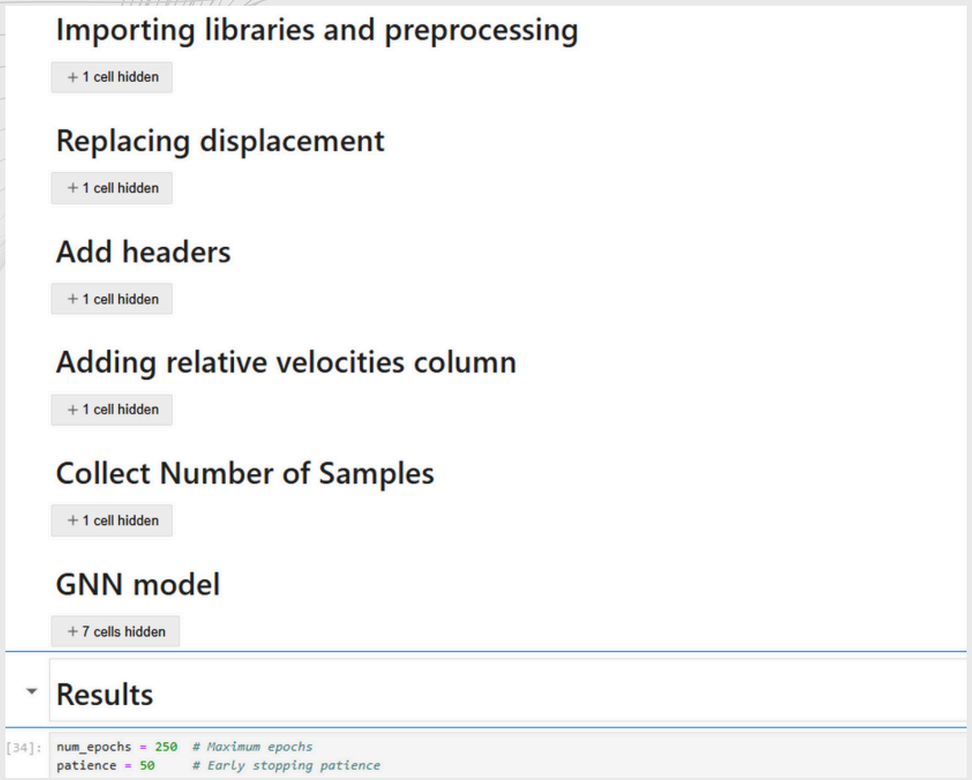


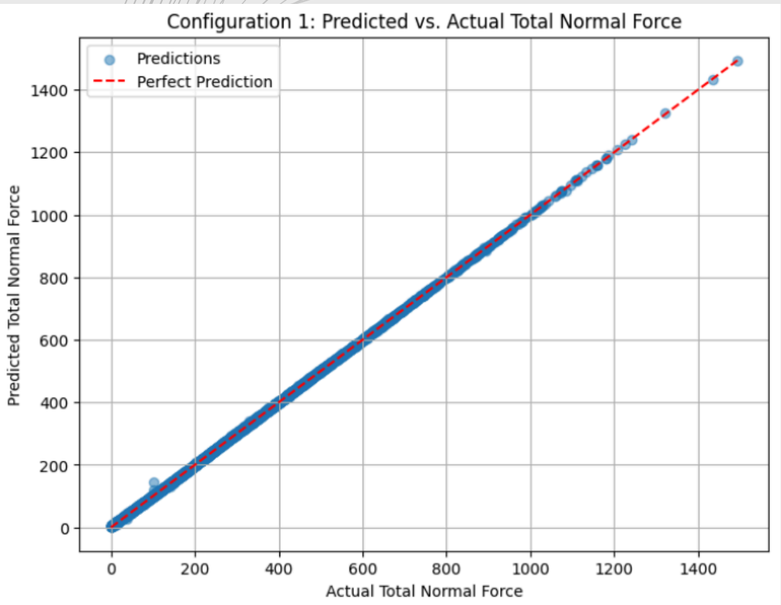
Fig 11: Consolidated python notebook for implementation of model

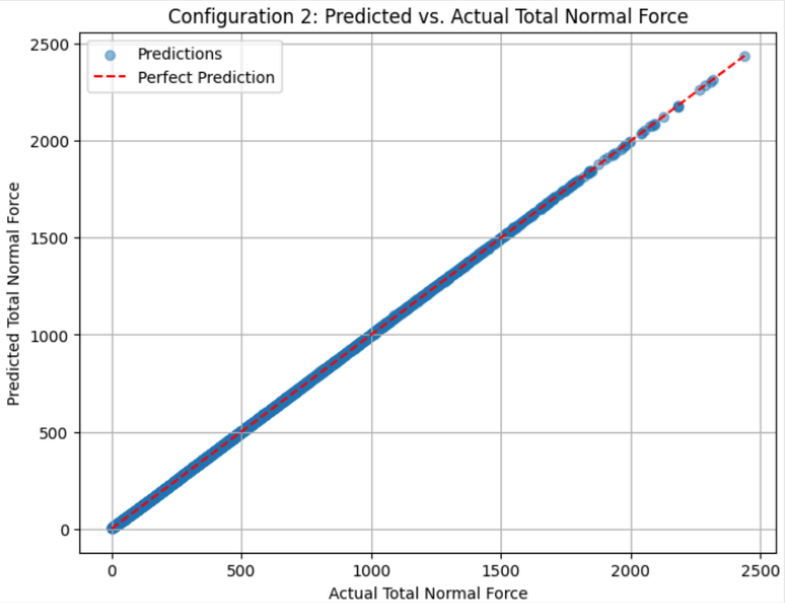
Fig 12: Plot of Loss (normalized) over epochs

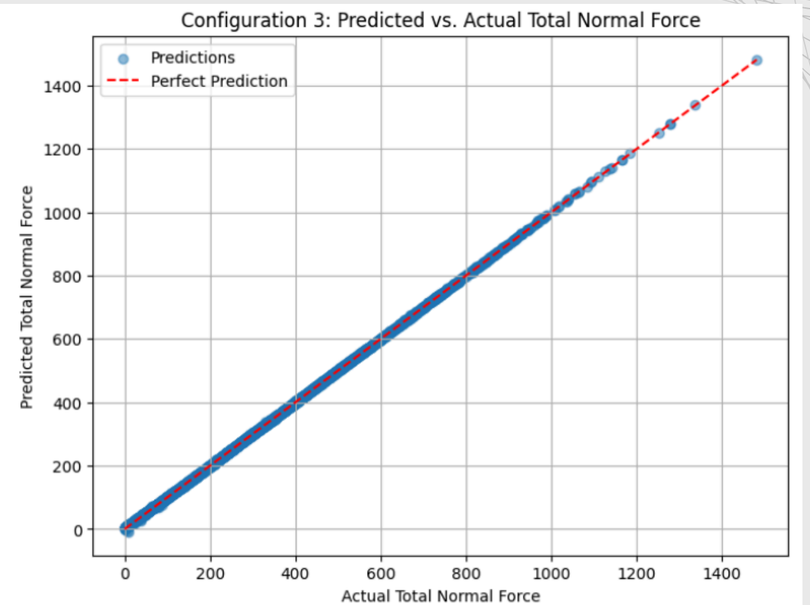
```
Train Metrics: {'L1': 1.349599, 'L2': 1.7624816, 'MSE': 3.1063414, 'R2': 0.9999790282688872, 'Pearson': 0.9999897258112725}
Validation Metrics: {'L1': 1.5005449, 'L2': 1.9418949, 'MSE': 3.7709556, 'R2': 0.9999046697315994, 'Pearson': 0.999953226305069 4}
Test Metrics: {'L1': 1.6077521, 'L2': 2.4920688, 'MSE': 6.2104063, 'R2': 0.9999746477135163, 'Pearson': 0.9999873445787801}
```

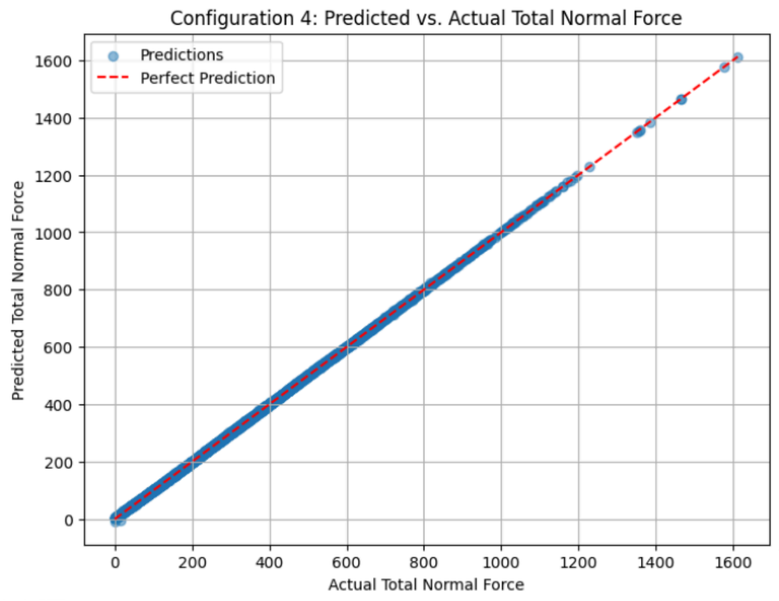
Fig 13: Train, Validation & Test Metrics for Model performance

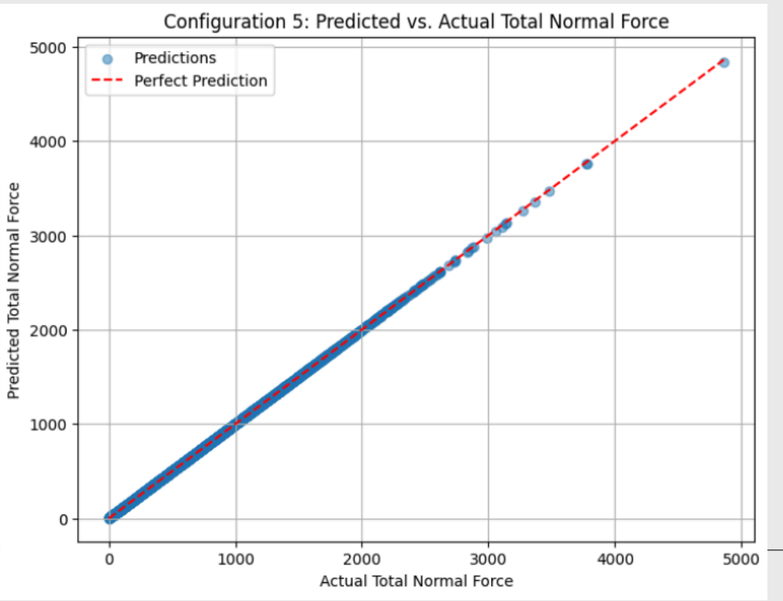
TRAINING CONFIGURATION PLOTS (ACTUAL VS PREDICTED)



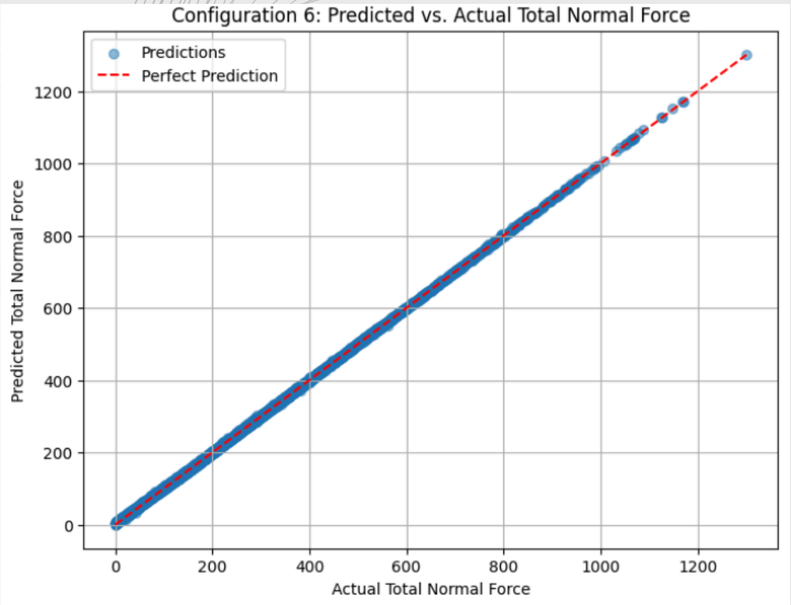


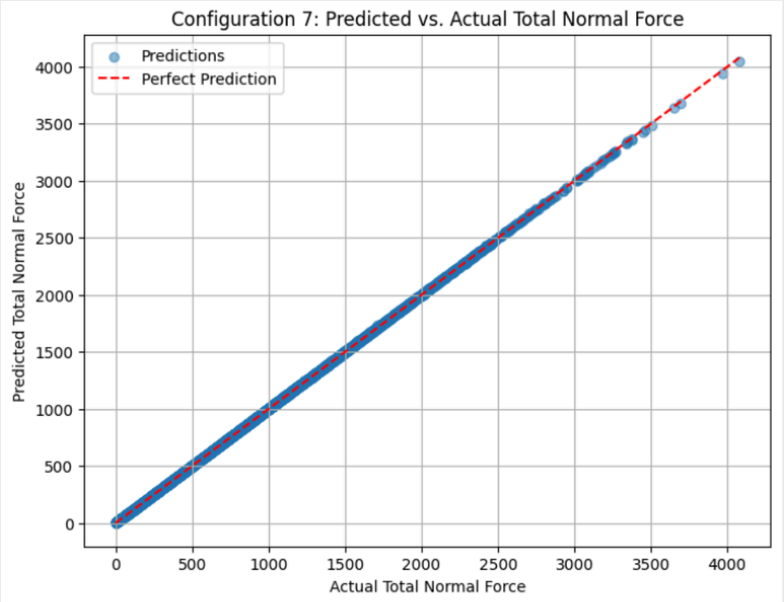


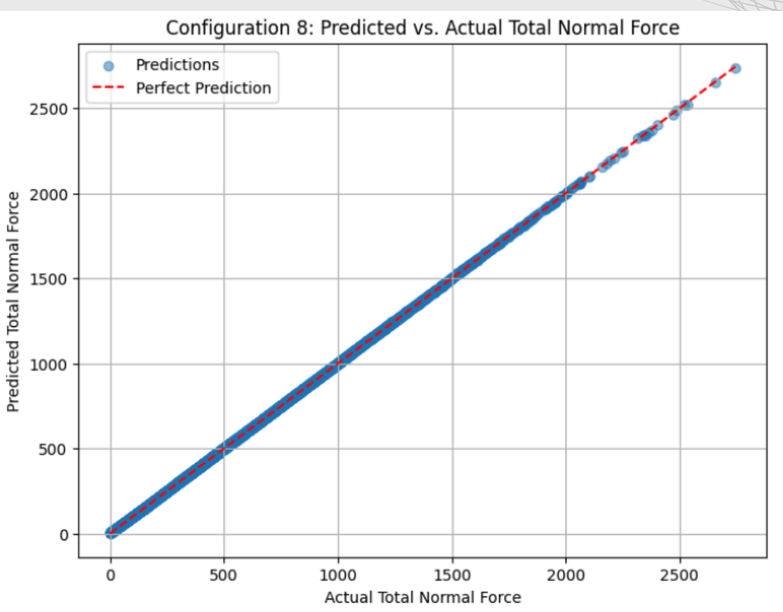


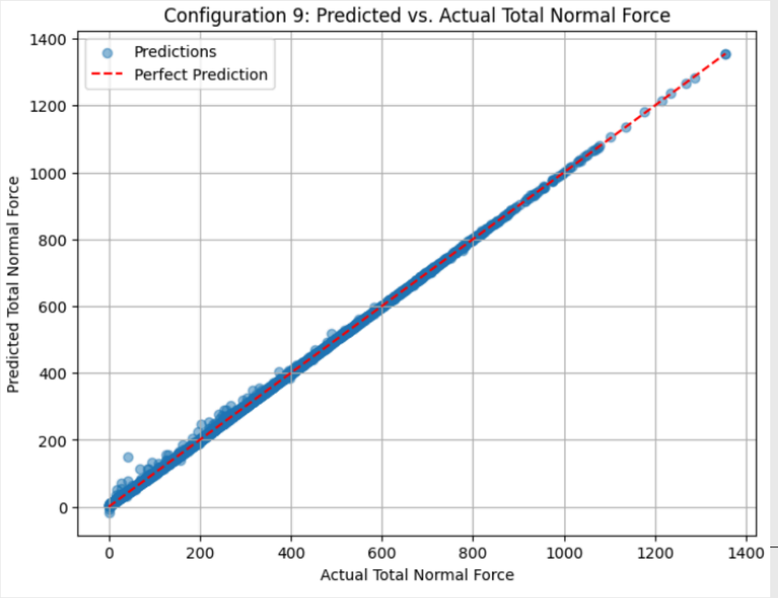


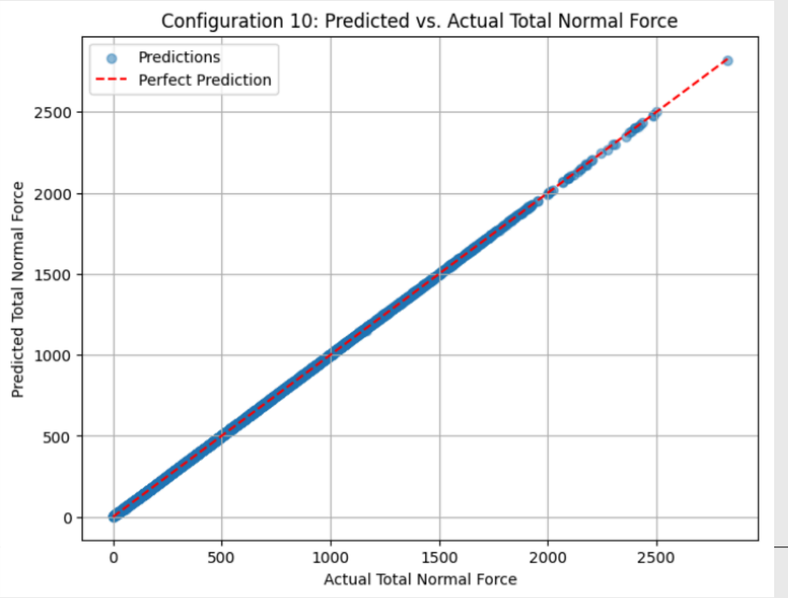
VAL & TESTING CONFIGURATION PLOTS (ACTUAL VS PREDICTED)











COMPARISON WITH PLOTS SHOWN BY CHENG, WANG (2022)

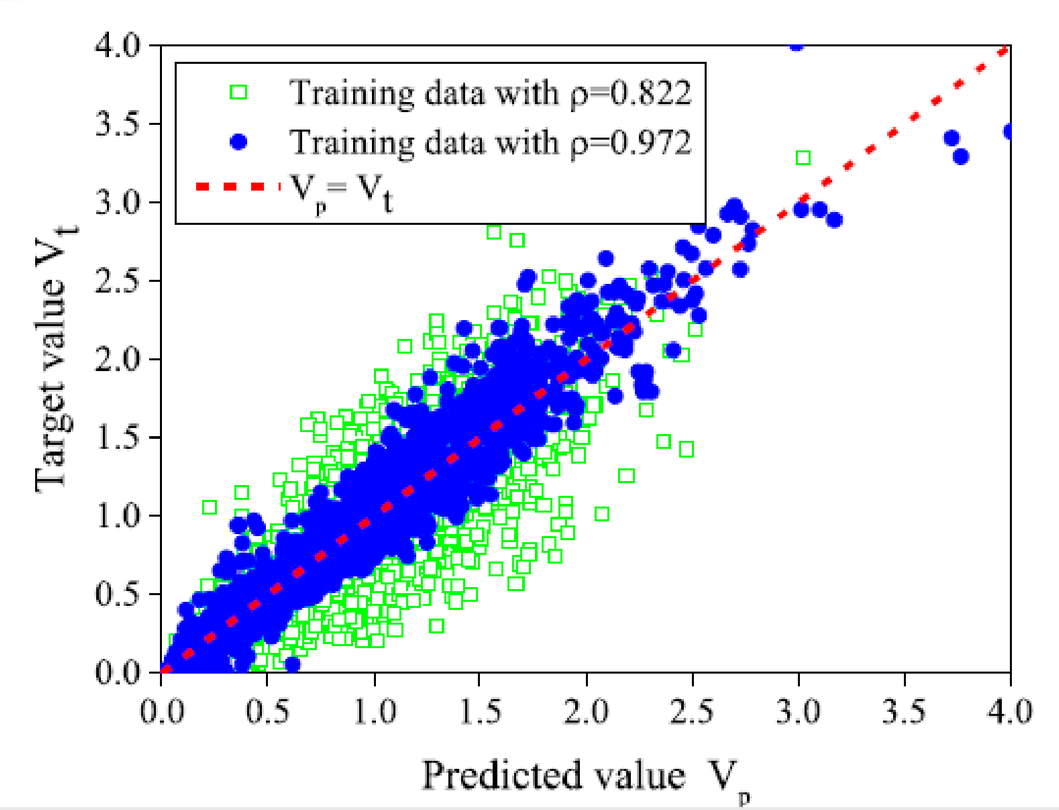


Fig 14: Results shown in Cheng & Wang's paper for normalized normal contact forces

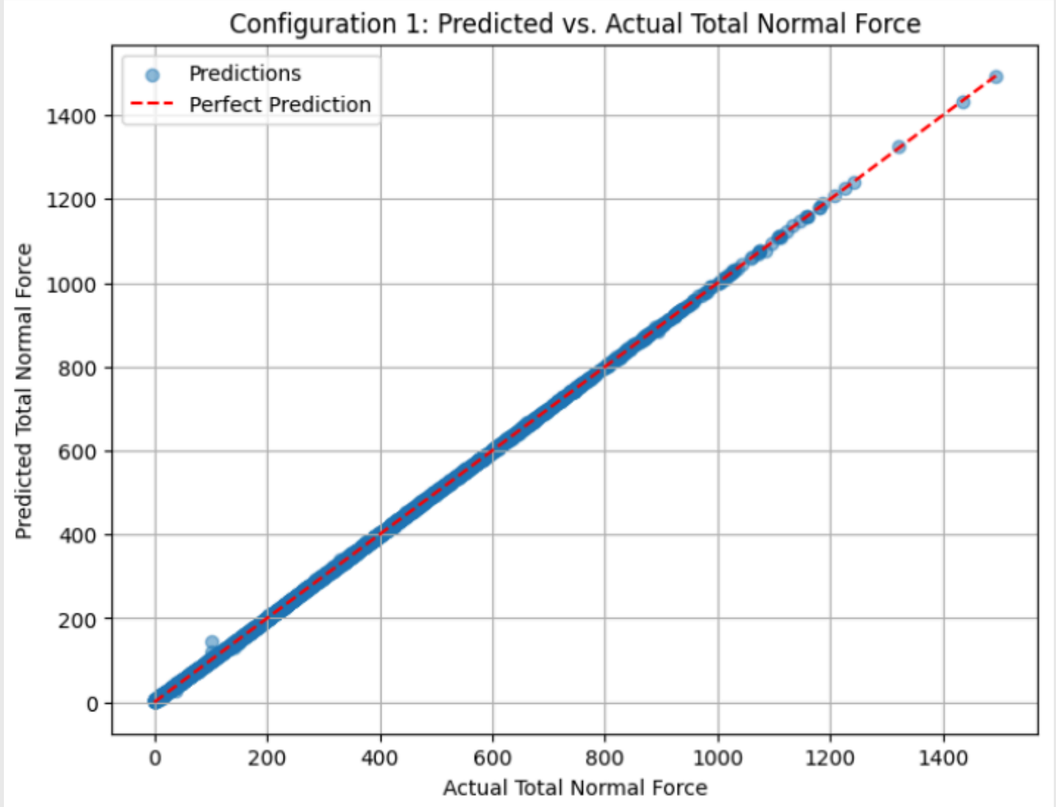


Fig 15: Results obtained by my model for normal contact forces

ADDITIONAL RESEARCH

Can such a model be applied to colloidal systems?

Yes, Graph Neural Networks (GNNs) can be effectively applied to colloidal systems to **predict contact forces or any other target interaction.** GNNs can learn the relationship between particle positions and force chains, enabling accurate predictions of contact force locations.

Aminimajd, Maia, Singh (2025) provides the evidence for this, implementing GNN to predict **frictional contact networks in suspensions.**

Potential:

- Colloidal systems share similarities with granular materials (particle interactions).
- GNNs could predict forces in complex colloidal suspensions which have graph like structure.

Challenges:

- Additional inter-particle forces (e.g., van der Waals, electrostatic).
- Need for experimental data validation.

CONCLUSION

- Developed a GNN model for normal contact force prediction in granular materials.
- Overcame initial limitations through different data and approaches.
- Achieved near-perfect accuracy (Pearson 0.999) with optimized model.
- Validated GNN as a fast, accurate alternative to DEM.
- Established a workflow for particulate systems.
- Applications in chemical, civil, and materials engineering.
- Can further be extended to experimental data and colloidal systems.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my supervisor, Dr. Saikat Roy, for his invaluable guidance and time and for providing me with the opportunity to work on such a meaningful and insightful project. This project has truly been a challenging yet incredibly rewarding journey, and pushed me to step out of my comfort zone and learn something new, which I'm thankful for. I also acknowledge the efforts of Mr. Divas Singh Dagur, who helped me with the procurement of data required for this project.

REFERENCES

- Battaglia et. al, 2018. Relational inductive biases, deep learning, and graph networks.

- Zhuang Cheng, Jianfeng Wang, 2022. Estimation of contact forces of granular materials under uniaxial compression based on a machine learning model.

- Rituparno Mandal, Corneel Casert, Peter Sollich, 2022. Robust prediction of force chains in jammed solids using graph neural networks. Nature Communications 13.

- Armin Aminimajd , Joao Maia and Abhinendra Singh, 2025. Scalability of a graph neural network in accurate prediction of frictional contact networks in suspensions

THANK YOU