#### Introduction

The integration of two-dimensional (2D) materials with Artificial Intelligence (AI) has garnered significant attention due to the unique properties of 2D materials and their potential applications in various fields. Atomistic simulations play a vital role in understanding these materials properties and behaviours at the atomic scale. However, extracting meaningful insights from large volumes of simulation data poses a challenge. This project aims to address this by developing a Python-based code for post-processing atomistic simulation data and setting up a deep learning model for the inverse design of 2D material-based heterostructures.

#### Project Objectives

1. Develop a Python-based code for post-processing and analyzing atomistic simulation data.
2. Design and train a deep learning model for inverse design to predict optimal 2D material-based heterostructure parameters for specific target properties.

**Workflow**

**WP1. Deep Learning Model for Inverse Design**

* Designed a deep learning model using the TensorFlow

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_12 (Dense) (None, 256) 1792

dense\_13 (Dense) (None, 128) 32896

batch\_normalization\_3 (Bat (None, 128) 512

chNormalization)

dense\_14 (Dense) (None, 128) 16512

dropout\_3 (Dropout) (None, 128) 0

dense\_15 (Dense) (None, 128) 16512

dense\_16 (Dense) (None, 64) 8256

batch\_normalization\_4 (Bat (None, 64) 256

chNormalization)

dense\_17 (Dense) (None, 64) 4160

dropout\_4 (Dropout) (None, 64) 0

dense\_18 (Dense) (None, 64) 4160

dense\_19 (Dense) (None, 32) 2080

batch\_normalization\_5 (Bat (None, 32) 128

chNormalization)

dense\_20 (Dense) (None, 32) 1056

dropout\_5 (Dropout) (None, 32) 0

dense\_21 (Dense) (None, 16) 528

dense\_22 (Dense) (None, 8) 136

dense\_23 (Dense) (None, 4) 36

=================================================================

Total params: 89020 (347.73 KB)

Trainable params: 88572 (345.98 KB)

Non-trainable params: 448 (1.75 KB)

Results:

Mean Squared Error: 0.034

Root Mean Squared Error: 0.186

Mean Absolute Error: 0.102

* Utilising several optimization techniques to predict inputs based on outputs. Several optimization algorithms were employed to fine-tune the model parameters, ensuring the best performance in predicting target properties:

1. **Differential Evolution (DE):**
   * Global optimization strategy effective for complex parameter spaces.
   * Optimized parameters were rounded to ensure compatibility with the model.

MAE: 0.106658

MSE: 0.041315

RMSE: 0.203260

1. **Particle Swarm Optimization (PSO):**
   * Swarm intelligence technique used for exploring the parameter space efficiently.
   * Produced competitive results in terms of optimization accuracy and time.

MAE: 0.099933

MSE: 0.020561

RMSE: 0.143392

1. **Simulated Annealing (SA):**
   * Probabilistic technique to escape local minima and find global optimum.
   * Balanced exploration and exploitation during the search process.

MAE: 0.106061

MSE: 0.023226

RMSE: 0.152399

1. **Shgo (Simplicial Homology Global Optimization):**
   * Combined simplicial homology with iterative refinement for robust global optimization.

MAE: 0.110324

MSE: 0.044614

RMSE: 0.211221

1. **Grid Search:**
   * Exhaustive search over a predefined grid of parameters.
   * Provided a baseline comparison for other optimization techniques.

MAE: 0.101092

MSE: 0.040492

RMSE: 0.201227

1. **Bayesian Optimization:**
   * Probabilistic model-based optimization.
   * Efficiently navigated the parameter space by balancing exploration and exploitation

MAE: 0.106494

MSE: 0.023240

RMSE: 0.152446

#### Generative Adversarial Network (GAN) for Inverse Design

In addition to the above methods, a Generative Adversarial Network (GAN) was employed to generate input parameters based on the desired output parameters. The GAN consists of a generator model that produces input parameters from random noise and a discriminator model that evaluates the authenticity of the generated samples. The GAN training involved alternating between updating the discriminator and the generator, ultimately enabling the generator to produce realistic input parameters corresponding to the target output.

Model: "generator"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=============================================================

dense\_93 (Dense) (None, 64) 320

batch\_normalization\_16 (Ba (None, 64) 256

tchNormalization)

dropout\_25 (Dropout) (None, 64) 0

dense\_94 (Dense) (None, 128) 8320

batch\_normalization\_17 (Ba (None, 128) 512

tchNormalization)

dropout\_26 (Dropout) (None, 128) 0

dense\_95 (Dense) (None, 256) 33024

batch\_normalization\_18 (Ba (None, 256) 1024

tchNormalization)

dropout\_27 (Dropout) (None, 256) 0

dense\_96 (Dense) (None, 256) 65792

batch\_normalization\_19 (Ba (None, 256) 1024

tchNormalization)

dropout\_28 (Dropout) (None, 256) 0

dense\_97 (Dense) (None, 512) 131584

batch\_normalization\_20 (Ba (None, 512) 2048

tchNormalization)

dropout\_29 (Dropout) (None, 512) 0

dense\_98 (Dense) (None, 6) 3078

=============================================================

Total params: 246982 (964.77 KB)

Trainable params: 244550 (955.27 KB)

Non-trainable params: 2432 (9.50 KB)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Model: "discriminator"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=============================================================

dense\_81 (Dense) (None, 1024) 11264

dropout\_16 (Dropout) (None, 1024) 0

dense\_82 (Dense) (None, 512) 524800

dropout\_17 (Dropout) (None, 512) 0

dense\_83 (Dense) (None, 256) 131328

dropout\_18 (Dropout) (None, 256) 0

dense\_84 (Dense) (None, 128) 32896

dropout\_19 (Dropout) (None, 128) 0

dense\_85 (Dense) (None, 64) 8256

dense\_86 (Dense) (None, 1) 65

=============================================================

Total params: 708609 (2.70 MB)

Trainable params: 0 (0.00 Byte)

Non-trainable params: 708609 (2.70 MB)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Model: "gan"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

=============================================================

input\_4 (InputLayer) [(None, 4)] 0 []

sequential\_12 (Sequential) (None, 6) 246982 ['input\_4[0][0]']

tf.concat\_2 (TFOpLambda) (None, 10) 0 ['sequential\_12[0][0]',

'input\_4[0][0]']

sequential\_13 (Sequential) (None, 1) 708609 ['tf.concat\_2[0][0]']

=============================================================

Total params: 955591 (3.65 MB)

Trainable params: 244550 (955.27 KB)

Non-trainable params: 711041 (2.71 MB)

#### Post-Processing of Generated Samples

After generating samples with the GAN, a post-processing step ensured the generated parameters remained within feasible bounds. This step involved rounding specific parameters and clipping them within predefined limits.

**WP2. Python-based Post-processing Module and Dataset**

* **Post-processing Code**:
  + Developed Python scripts for data generation for 2D heterostructure.
  + Also calculating parameters like defect density, no of defects etc.

**Results**

The optimization algorithms provided various sets of parameters that minimized the loss function, predicting the target output as accurately as possible. The results from each optimization method were compared to evaluate their effectiveness and efficiency. The results indicate that the chosen model architecture and loss function are effective for inverse design. The GAN model, with its ability to generate input parameters for given outputs, adds an additional robust tool for the inverse design process.

**Expected Outcomes**

1. A robust Python-based tool for efficient post-processing of atomistic simulation data.
2. A trained deep learning model capable of accurately predicting the optimal parameters of 2D material-based heterostructures for given target properties.

**Research Significance**

This project addresses key challenges in the inverse design of 2D material-based heterostructures by combining AI and atomistic simulations. The developed tools and models facilitate the exploration of the vast design space of 2D materials, paving the way for novel applications in various technological domains.

Link To GitHub code Repo: <https://github.com/KaxitPandya/InverseDesign.git>