AI 4 Science Discovery Network+

Group: 7
Challenge: 3 - Defect Detection in Graphene Sheets
AI4SD ML Summer School Report
20-24th June 2022

Project Team: James Osborne (University of Liverpool), Ellie Nelson (University of York), Edvin Mamo (University College Dublin), Shaoqi Zhan (University of Oxford), Steven Tendyra (University of Manchester)

Report Date: 05/07/2022

Group: 7
Challenge: 3 - Defect Detection in Graphene Sheets
AI4SD-SummerSchool-Series:Report-5
Report Date: 05/07/2022
DOI: 10.5258/SOTON/AI3SD0248
Published by University of Southampton

Network: Artificial Intelligence and Augmented Intelligence for Automated Investigations for Scientific Discovery
This Network+ is EPSRC Funded under Grant No: EP/S000356/1

Principal Investigator: *Professor Jeremy Frey*
Co-Investigator: *Professor Mahesan Niranjan*
Network+ Coordinator: *Dr Samantha Kanza*

Contents

1 Project Details

Group Number	7
Challenge Name	3 - Defect Detection in Graphene Sheets
Project Dates	20-24th June 2022

2 Project Team

2.1 Project Student

Name and Title	James Osborne
Employer name / University Department Name	University of Liverpool
Work Email	J.Osborne@liverpool.ac.uk

Name and Title	Ellie Nelson
Employer name / University Department Name	University of York
Work Email	efn509@york.ac.uk

Name and Title	Ellie Nelson
Employer name / University Department Name	University of York
Work Email	efn509@york.ac.uk

Name and Title	Edvin Mamo
Employer name / University Department Name	University College Dublin
Work Email	entvin.mamo@ucdconnect.ie

Name and Title	Shaoqi Zhan
Employer name / University Department Name	University of Oxford
Work Email	shaoqi.zhan@chem.ox.ac.uk

Name and Title	Steven Tendyra
Employer name / University Department Name	University of Manchester
Work Email	steven.tendyra@postgrad.manchester.ac.uk

2.2 Challenge Description

Challenge 3 was focused on the identification of defects present within graphene sheets. Provided with electron microscopy images of sheets of graphene, the data-set was partitioned into a sample of perfect patches (regions of an image which do not contain defects), defect patches (regions of an image which contain a defect) and a data-set of images which are not edited or partitioned into smaller sections of analysis. The full image is 256 x 256 patches (Figure 1a). The blue (high electron density) corresponds to atoms and the green corresponds to background. In the full image patches, there is a perfect 48 x 48 patches and a defect 48 x 48 patches. By selecting and training an appropriate machine learning model, the goal was the identification of defect regions contained within a whole electron microscopy image of a graphene sheet.

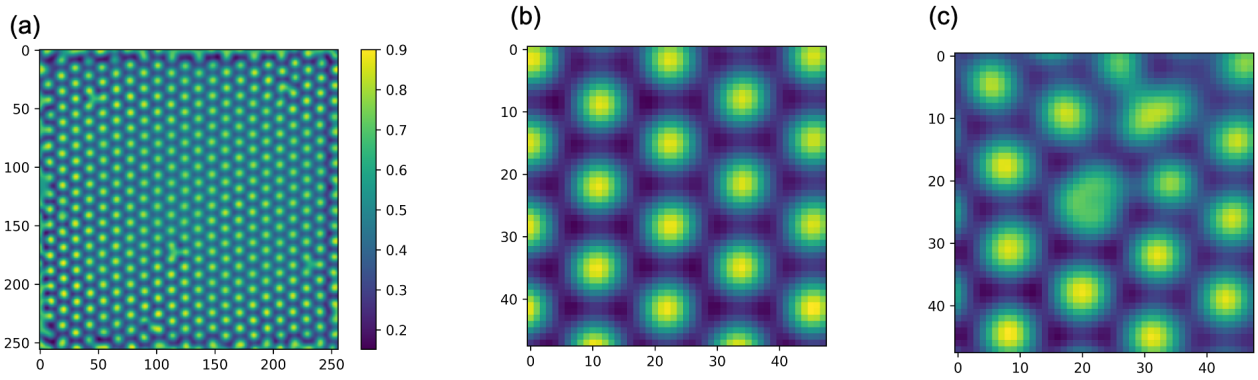

Figure 1: **The image of graphene sheets.** (a) The full image (256 x 256 patches), (b) The perfect 48 x 48 patches; (c) The defect 48 x 48 patches

3 Lay Summary

For the design of useful materials, structure is an important factor in determining material properties which are known a the structure-property relationship. The presence of disorder can negatively impact the properties of a desired material. Commonly, electron microscopy is used to capture a "picture" of the material structure. Using this, areas of disorder known as defects can be identified. Due to the large images captured during this process, automated techniques using machine learning and deep learning can be used to automatically identify defects electron microscopy images. Here we were given images for graphene, a popular material for it's electronic properties and flexibility. Here we used a one-class support vector machine (OneClassSVM), which is a popular technique for identifying outliers in a data-set, which is exactly what defect detection requires. By training the model to recognise defects, large images can be screened and defects can be identified, accelerating materials design processes through automated means.

4 Methodology

Scientific basis. Graphene is an important material in the development conductive components used to develop energy storage, optoelectronics, flexible electronics, robotics and textiles. [1, 2] Analysis of the material structure is an important aspect of rationalising material properties in devices. Here, we have electron microscopy images of graphene (a 2D nanomaterial formed from a sheet of graphitic carbon) which detail the atomic structure of the material. Using these images, defects within the structure can be identified and by using AI-based tools, these defects can be identified automatically and routinely. Here we adopt the use of two methods: One-Class Support Vector Machines (OneClassSVM) and Convolutional neural networks (CNNs) as two possible approaches to classify defects within electron microscopy images of graphene.

OneClassSVM. Is an unsupervised learning algorithm that is commonly deployed in outlier detection processes since it identifies data points within a range of the class with a 1 and all outliers with a -1 value. Once trained, new data can be accepted by the model and can be assigned a 1 or -1 value, allowing for the detection of further outliers in larger data-sets, post-training. The boundary defined by the SVM model is formed from a (n-1) dimensional hyperplane, where n is the dimensionality of the vectors which define each data point. The boundary defined by this hyperplane denotes whether a data point belongs to the class present in the dataset, or lies outside it.

The practical implementation of this model was performed using Python 3.10 using the scikit-learn, OneClassSVM model. This model accepts a number of parameters: kernel (the type of kernel method used to compute the hyperplanes defining the boundary), gamma (defines the scale of the kernel coefficient), tol (the tolerance of the stopping criterion) and nu (the percentage out outliers provided in the training dataset). Selection and balancing of these parameters is an important factor to consider when optimising a machine learning model for use on a problem and is typically problem dependent.

CNN Approach. In the convolutional neural network (CNN) approach, a simple image classifier CNN was implemented along with the use of TensorFlow (TF). TensorFlow is an end-to-end open source platform for machine learning (ML) applications. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. So, firstly the SEM perfect and defect graphene patches were loaded in the python script to preprocess the data. Since the pixel values fall in the range of 0 to 255, scaling these values to a range of 0 to 1 through dividing the values by 255 before feeding them to the neural network model. Right after that and in an effort to create a robust training data for the developing model, all the data was concatenated. Labels were put in the data and tuples were created and tuples were shuffled. The next step is to build a machine learning model, using the tf.keras.Sequential command by stacking layers. Before starting the training process, configure and compile the model with the use of Keras Model.compile command is essential. The optimizer class was set to 'adam', the loss set to sparse categorical entropy as it proved to have better results. Last for the evaluation of a metric for the model the metrics parameter was set to accuracy. The Model.fit model was used to adjust the model and minimize the loss with the split of the data to a validation and test set.

U-Net Approach. An additional CNN-based approach, this time based around the U-Net architecture [3] was considered. Whilst it is generally accepted that training deep networks requires many thousands of instances of labelled training data, the U-Net architecture and training strategy is presented as an alternative that relies strongly on data augmentation to use the

available training data more efficiently. U-Net has been widely utilised for semantic segmentation within the biomedical imaging domain, and widely adapted for other disciplines. One of its key advantages is that it can be modified for small datasets whilst still maintaining high levels of accuracy [4]. However, building a complete implementation tailored for this problem would likely be out of the scope of this project. A U-Net-like architecture was built using PyTorch [5], an open source machine learning framework. This consisted of blocks, an encoder, decoder, and the various classes required to complete the implementation. Blocks were defined to have Conv2D operations with ReLU activation between them. The encoder, the contractive part of the architecture, utilised 2x2 MaxPooling and stride 2 downsampling. The decoder consisted of an upsampling followed by convolutions, cropping of the feature map, and ReLU.

5 Results

OneClassSVM. For this approach we employed the scikit-learn library in Python. [6] As OneClassSVM is an unsupervised learning method, to train the model both the datasets containing the perfect patches and the defect patches were concatenated, and randomly sorted. The data was then reshaped from a 3d array to a 2d array in order to be compatible with the scikit-learn OneClassSVM function.

Four different kernel methods were tested to discern which method produced the best model fit. These included: a radial basis function (RBF), polynomial, sigmoid and linear. Each kernel method was tested and the radial basis function as it yielded the most sensible prediction for the number of errors held within the data. The RBF kernel provided the lowest error of 1.38%, which we perceive as an acceptable error in labelling the data and was the lowest error we achieved during parameter tuning. We then aimed to test our model on the full-stack dataset provided. However, as the dataset contained was not the same size as the training data, we struggled to find a way to resize the data so it could be tested with our model. The full code for this project can be found in the Jupyter notebook present on our team GitHub under *Graphene_SVM.ipynb*.

CNN approach After the machine learning model was run, outputs were utilized for evaluation. It was found that the graphene image classifier was trained with an accuracy of 93% on this data-set. Test accuracy was found around 70%. The plots of training accuracy and validation accuracy are off by large margins, and the model has achieved only around 70% accuracy on the validation set. the training accuracy is increasing linearly over time, whereas validation accuracy stalls around 71% in the training process. Also, the difference in accuracy between training and validation accuracy is noticeable—a sign of overfitting. When there are a small number of training examples, the model sometimes learns from noises or unwanted details from training examples—to an extent that it negatively impacts the performance of the model on new examples. This phenomenon is known as overfitting. It means that the model will have a difficult time generalizing on a new dataset. There are multiple ways to fight overfitting in the training process.

U-Net Approach Due to the inherent complexity in tailoring and implementing CNN-like architectures from scratch for a given problem, only the training stage was reached by the end of the challenge. A functional U-Net architecture was implemented, but segmenting data appropriately and running it through the architecture proved challenging.

6 Conclusions & Future Work

By the end of the day, we had not produced a method that was able to detect defects within the dataset provided. In our group, much of the morning was spent researching methods appropri-

ate to this problem, and the afternoon was spent building the code to train the OneClassSVM model. Should we have more time to spend on this problem in the future we would aim to resize the individual images within the full-stack dataset to be the same size as the training data. We would then build a program that could iterate through the dataset to detect which individual images contain the defects.

Concerning the CCN method, it seems that overfitting was the main problem due to small data provided for training. As a result this approach may not be the perfect one, although there possible ways to overcome this problem. One quite considerable way to avoid overfitting is data augmentation. Data augmentation takes the approach of generating additional training data from existing examples by augmenting them using random transformations that yield believable-looking images. These random transformations can be random flip, random zoom or random rotation of the existing images.This helps expose the model to more aspects of the data and generalize better. Also as the SEM images are grayish, another potential solution can be the extraction of pixels of every image and the calculation of average value of them along with SME's in an effort to effectively classify the perfect and defect patches.

Concerning the U-Net architecture, whilst perhaps out of the scope of this project or challenge, such an approach should be able to be tailored to this particular problem with suitable modifications such as an appropriate loss function, dropout layers, or modifications to the dataset such as adding Gaussian noise or augmentation. Whilst not necessarily the most straightforward approach, it would serve to demonstrate the applicability of U-Net within other scientific domains.

7 Outputs, Data & Software Links

Code for this project, along with the relevant datasets, can be found in the following GitHub Repository: https://github.com/J-Osb/AI4SD_summer_school_2022_group7.

References

[1] Farjadian F, Abbaspour S, Sadatlu MAA, Mirkiani S, Ghasemi A, Hoseini-Ghahfarokhi M, et al. Recent Developments in Graphene and Graphene Oxide: Properties, Synthesis, and Modifications: A Review. ChemistrySelect. 2020;5(33):10200-19. Available from: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/slct.202002501.

[2] Razaq A, Bibi F, Zheng X, Papadakis R, Jafri SHM, Li H. Review on Graphene-, Graphene Oxide-, Reduced Graphene Oxide-Based Flexible Composites: From Fabrication to Applications. Materials. 2022;15(3). Available from: https://www.mdpi.com/1996-1944/15/3/1012.

[3] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv; 2015. Available from: https://arxiv.org/abs/1505.04597.

[4] Prasad P, Elle OJ, Lindseth F, Albregtsen F, Kumar RP. Modifying u-net for small dataset: a simplified u-net version for liver parenchyma segmentation. In: Drukker K, Mazurowski MA, editors. Medical Imaging 2021: Computer-Aided Diagnosis. SPIE; 2021. Available from: https://doi.org/10.1117/12.2582179.

[5] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. p. 8024-35. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[6] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30.

List of Figures