AI 4 Science Discovery Network+

Group: 6
Challenge: Task 3 - Detect Defects in Electron Microscopy Images
AI4SD ML Summer School Report
20-24th June 2022

Project Team: Robert Dickson, Ben Honore, Hai Lin, Rachael Pirie

Report Date: 01/07/2022

Group: 6
Challenge: Task 3 - Detect Defects in Electron Microscopy Images
AI4SD-SummerSchool-Series:Report-4
Report Date: 01/07/2022
DOI: 10.5258/SOTON/AI3SD0247
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	6
Challenge Name	Task 3 - Detect Defects in Electron Microscopy Images
Project Dates	20-24th June 2022
Website	https://github.com/RPirie96/AI4SD_group6.git

2 Project Team

2.1 Project Student

Name and Title	Robert Dickson
Employer name / University Department Name	University of Liverpool
Work Email	rd712@liverpool.ac.uk
Website Link	https://www.linkedin.com/in/robert-dickson-0007207b/ https://github.com/robertcdickson

Name and Title	Ben Honore
Employer name / University Department Name	University of Bristol
Work Email	bh17536@bristol.ac.uk
Website Link	https://github.com/Benhonore

Name and Title	Hai Lin
Employer name / University Department Name	University of Liverpool
Work Email	hailin@liverpool.ac.uk
Website Link	www.linkedin.com/in/hai-lin-82a077224/ https://github.com/hailin1991

Name and Title	Rachael Pirie, Miss
Employer name / University Department Name	Newcastle University
Work Email	r.pirie2@newcastle.ac.uk
Website Link	www.linkedin.com/in/rmepirie https://github.com/RPirie96

2.2 Project Supervisor

Name and Title	Jeremy Frey, Prof.
Employer name / University Department Name	University of Southampton
Work Email	J.G.Frey@soton.ac.uk
Website Link (if available)	https://www.southampton.ac.uk/chemistry/about/staff/jgf.page

2.3 Challenge Description

Graphene is an exotic nanomaterial consisting of a single layer of carbon atoms arranged in a two-dimensional honeycomb lattice (Figure 1). Since its rediscovery and isolation from graphite by tape in 2009, [1] graphene has aroused intensive interest in the worldwide and in numerous research areas, due to its extraordinary mechanical, thermal, optical, electronic, and other properties. The discoverer Geim and Novoselov were awarded the Nobel Prize in Physics in 2010.

Although graphene samples with high perfection have such outstanding performance, the defects in graphene are inevitable during the growth and processing, and can significantly affect the properties. Thus, it is important to figure out the amount, position, size and type of defects in a graphene sample. Machine learning, an emerging data-driven approach, offers a highly efficient solution to learning hidden patterns or classifying anomalies from complex data.

In this work, we applied machine learning to detect the defects in the electronic microscopic images of graphene samples and compared the performance of several different machine learning algorithms. As the data set contained significantly fewer defect examples than perfect ones, only methods suitable for use with imbalanced data were considered.

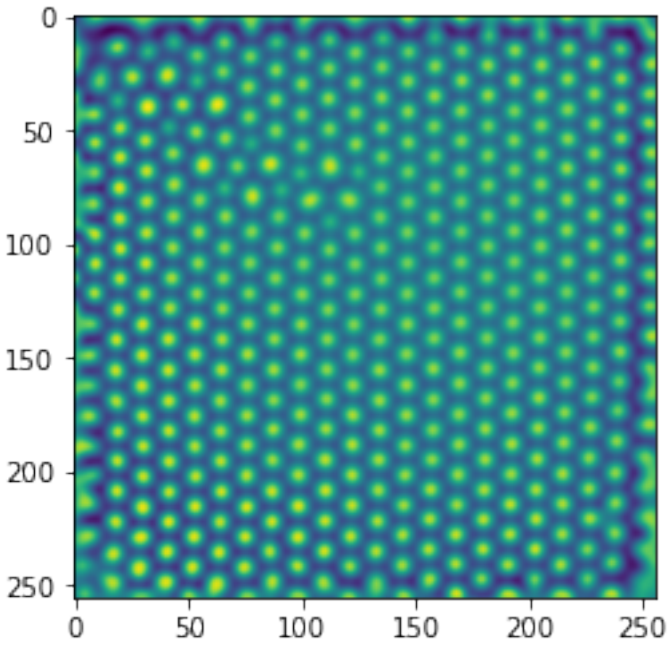

Figure 1: An electronic microscope image of graphene

3 Lay Summary

The challenge of detecting sections of images containing defects is an example of an *outlier detection* problem, where the model used looks for points that do not conform to the behaviour of the rest of the available data. The model is "shown" a set of examples, and learns what a perfect patch and a defect patch look like. It then uses this information to predict whether a new patch contains a defect. Typically, machine learning models need a similar number of perfect and imperfect examples to perform well. However the data set we have here has far more perfect examples than imperfect, which needs to be accounted for in the model chosen. Four different machine learning methods were investigated to classify whether an image patch contains a defect or not: a one class Support Vector Machine, which learns only what a perfect example looks like, then for unseen examples anything that does not resemble this is labelled as an outlier; an Isolation Forest, a method which aims to construct a tree of features of the data, where an outlier will sit closer to the beginning of the tree; a dense neural network, trained on both perfect and imperfect examples, which learns by passing information through a series of decision points; and a convolutional autoencoder, trained to break down and reconstruct perfect images, where the reconstruction of an imperfect example will be worse, allowing it to be identified.

4 Methodology

4.1 One Class SVM

Support vector machines (SVM) and a class of algorithms that can separate data in to two distinct classes by forming a hyperplane in N-dimensional space. Although many hyperplanes that correctly separate the training data may be found, the goal of the SVM is to find a hyperplane that *maximises* the margains between the two classes. While these algorithms are in general a form of supervised learning, the one-class SVM is an unsupervised outlier detection algorithm that, given a biased training set of inliers and outliers, defines a non linear classification of the perfect and imperfect data points (See Figure 2).

In this work, we apply the one-class SVM to the problem of identifying defects in graphene

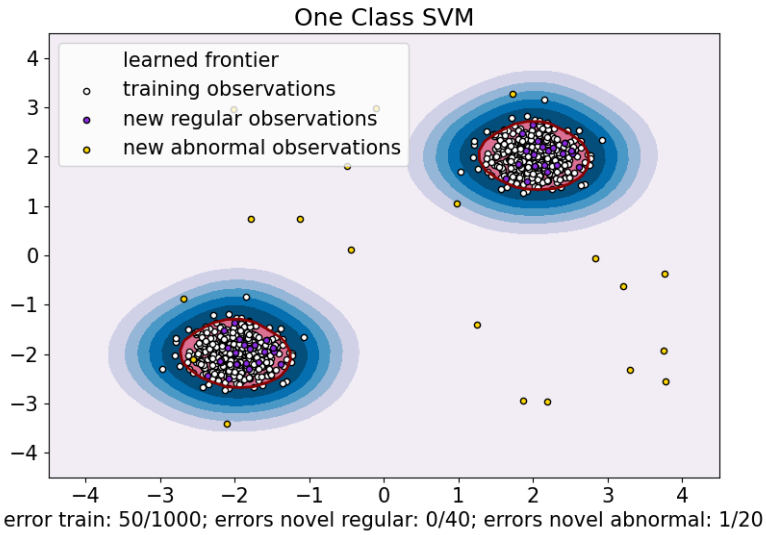

Figure 2: An example of One Class SVM.

electron microscopy images. We use the one-class SVM as implemented in the scikitlearn [2] python package using the methodology of Schölkopf and coworkers. [3]

4.2 Isolation Forest

Isolation forest is an anomaly detection algorithm that explicitly isolates anomalies instead of profiles normal points. [4] Due to the anomalies are "few and different", they are more susceptible to isolation than normal points. A typical example is presented in Figure 3. In this Gaussian distribution (135 points), all data points are isolated by repeating recursively partition in a data-induced random tree. During this process, an anomaly x_o in (b) always possesses a shorter path length than a normal point x_i in (a). That is because the fewer instances of anomalies result in a smaller number of partitions, and the more distinguishable instances are more likely to be separated in early partitioning. This isolation characteristic of tree forms the basis of this isolation forest method to detect anomalies.

In this work, since the defect cases in the training dataset are much fewer and distinguishable than perfect ones, the isolation forest model is very useful to detect the defects as anomalies.

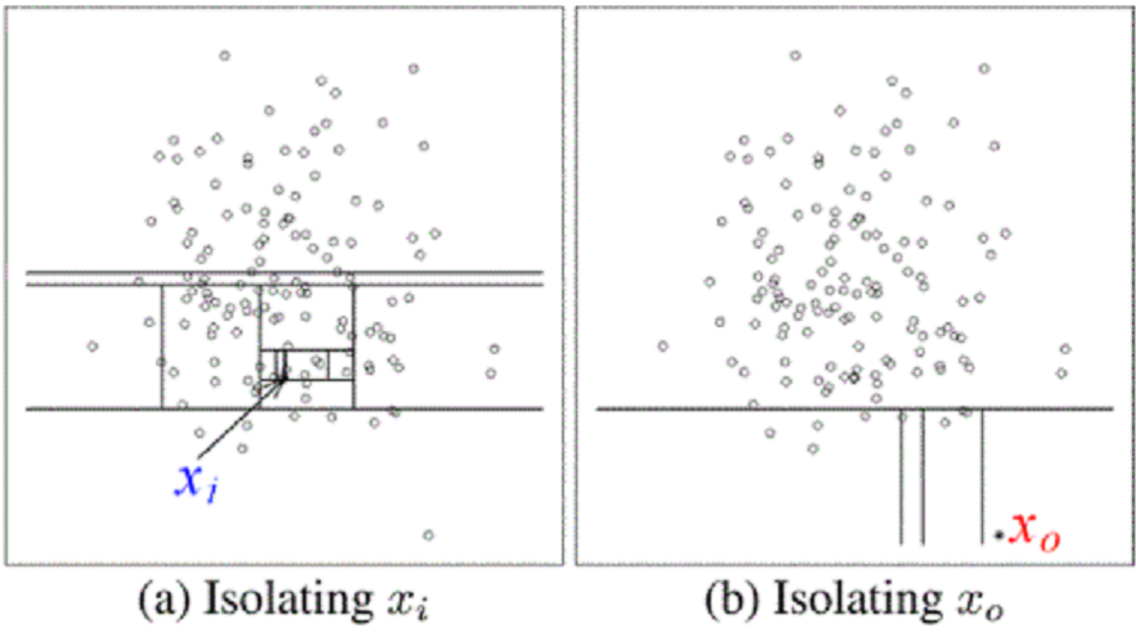

Figure 3: An example of Isolation Forest. (a) Normal data points x_i are less susceptible to isolation than (b) anomalies.

4.3 Dense Neural Network

Neural networks are a form of machine learning architecture made up of a series of layers containing multiple nodes. Information is passed between nodes in a neural network to generate an output via at least one hidden layer. At each node, weighted inputs are received from the previous layer and combined, put through an activation function and the output is passed on to the next layer (Figure 4).

In supervised learning, neural networks are optimised through training. At each training iteration, a loss is calculated between the model output and the target value. This loss metric depends on the task: for example, mean squared error (MSE) is popular for regression tasks where an exact value is output, whereas cross entropy is common for classifiers. The calculated

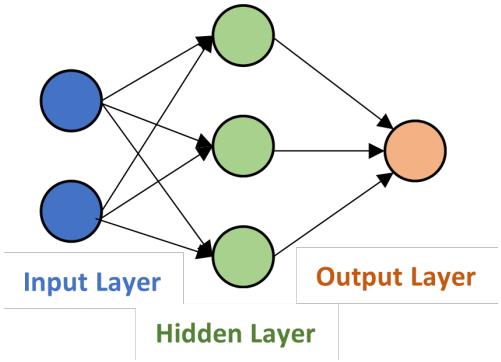

Figure 4: Basic architecture of a neural network

loss is then backpropagated through the network, tuning the parameters within the network to minimise the loss. Over a series of training iterations, the training loss steadily decreases. [5]

Keras is a deep learning API, available within the TensorFlow library, that can be used to build, train and trial neural networks with minimal programming. [6] Two fully connected hidden layers were implemented with 3000 and 1000 nodes in that order. This was perhaps excessive for the given dataset size but efficient optimisation and activation methods were used to afford more time to a complex model. The output layer contains ten nodes passed through a sigmoid activation function to return ten output values between zero and one. These outputs effectively vote to obtain a final value of zero (no defects present) or one (defects present). The overall architecture of the implemented model is shown in Figure 5.

The Rectified Linear Units (ReLU) activation function returns values as they were unless they were below zero, in which case it returns zero. This near linearity means that ReLU is fast to calculate and it works well in practice but the fact that it does not distinguish values below zero means that the slope is not always smooth during backpropagation. The optimisation method used was stochastic gradient descent (SGD). This determines the gradients based only on one randomised training instance which makes training much faster and better able to handle large datasets. A binary categorical cross entropy loss function was used, as is popular in classification tasks where the answer takes one of two values. [5, 7]

4.4 Convolutional Auto Encoder

The final approach tested was that of a Convolutional Autoencoder implemented in Keras, [6] presented by Wells. [8] This is an example of a deep learning approach, where a neural network is trained on perfect examples only. The autoencoder first deconstructs the images by compressing the data into fewer and fewer dimensions. The goal is then to reconstruct the images from the compressed data as accurately as possible. When the network is fed an imperfect example, it will try to reconstruct something that looks like a perfect example, resulting in a larger error and thus allowing defects to be identified. The theory here is that because the model is not trained on explicit imperfect examples, it should be able to detect them even when we don't know what they look like.

The architecture of the autoencoder is shown in Figure 6. The model is constructed from an input layer, several hidden layers that compress the data (encoder part), several hidden layers that expand the data (decoder) and an output layer. Dimensionality reduction of the data is achieved using 2x2 max pooling, [9] where 4 pixels containing a RGB colour value are reduced to a single pixel. The likelihood that an image belongs to the perfect class is modelled using a technique known as kernel density estimation. [10] This measures how densely

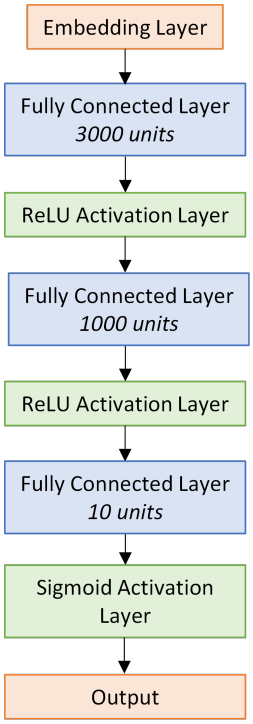

Figure 5: Architecture of the implemented Dense Neural Network

the training data occupies each part of the vector space (constructed from the pixels in each image). For a new observation, an estimation of the density is made by considering the distance of the point to the training data. It is assumed that imperfect images will occupy areas of the latent space seen less frequently in the training data, resulting in a lower density. The assumption here is that anomalous images will occupy areas of the latent space that were seen less frequently in the normal training data and therefore have lower density. Classification can then be made on a threshold reconstruction error or probability density, the values of which are selected based on the task. For the Fruits 360 data set used in the original article these were set to higher than 95% reconstruction error or probability density lower than 99% of the normal images. The code for the method was adapted from Well's implementation (https://github.com/JudeWells/keras_anomaly_detection).

5 Results

The accuracy of the one-class SVM was found to be 97.9 %. This is unsurprising as the dataset is heavily biased towards the perfect graphene images and so classifying most images as perfect will give a very high accuracy. A better measure of performance, the receiver operating characteristic (ROC) curve is presented in Figure 7. This gives an area under the curve (AUC) of 0.63. As an uninformative classifier will have a AUC of 0.5 and a perfect classifier an AUC of 1.0, we can note that the one-class SVM classifies the data better than an uninformative classifier, but still with relatively poor performance.

The isolation forest algorithm was employed here to build a model for the defect detection. The random state was set as 42 and the contamination was 1.4%, which is the ratio of defect patches in all training samples. The obtained accuracy of the isolation forest was found to be 99.1%, better than other models but still not convincing enough due to the highly unbalanced raw data. The ROC curve is shown in 8 and gives an AUC of 0.91. The high AUC indicates this isolation forest model has a high probability to detect the defects in the graphene images.

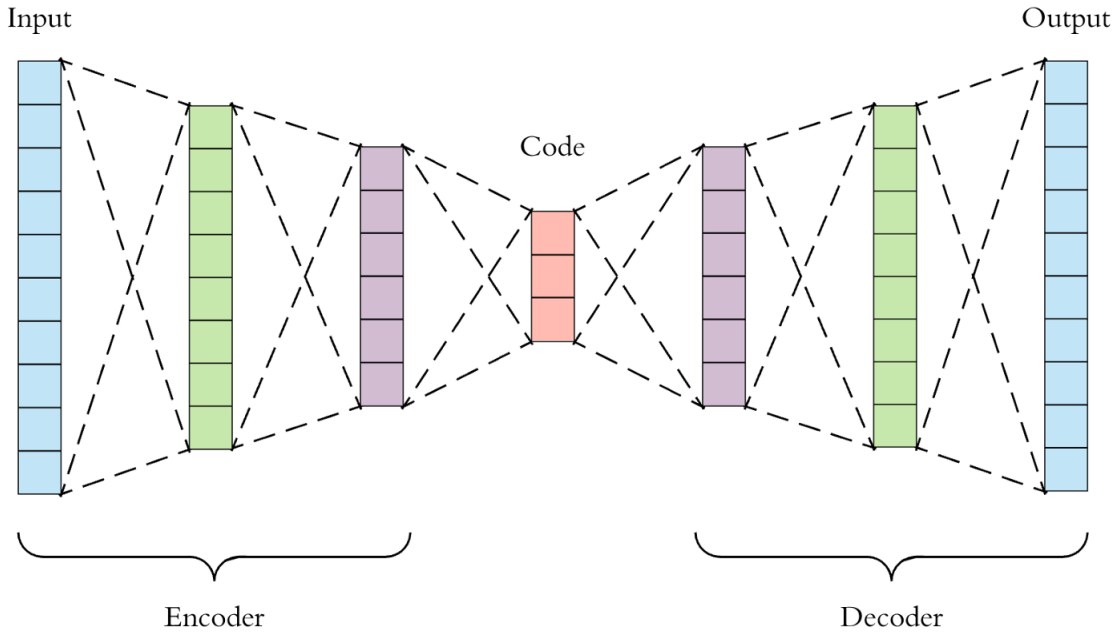

Figure 6: Architecture of the feed-forward neural network deployed to classify images, where each square represents a single image pixel in the input and output layers, and a fully connected node in the middle layers. Reproduced from [8]

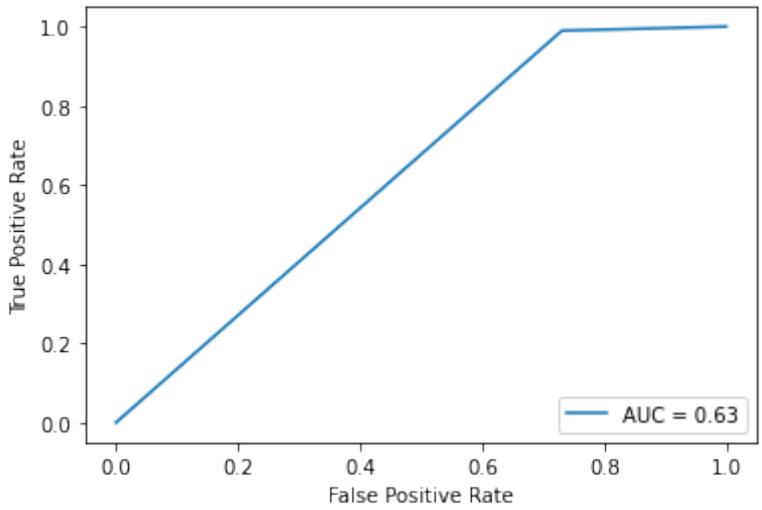

Figure 7: Receiver Operating Characteristic curve for the one-class SVM algorithm.

The method was then tested on a small set of patches from two examples of unseen images - one containing a large number of defects, and one containing no defects. In both cases, compared to visual inspection, the model misclassified all of the patches. This suggests that the model has been overfit to the training data.

The dense neural network was trained using an 80:20 train:test split. Following 200 training iterations, the model achieved the maximum ROC-AUC score of 1.0 on the held-out testing portion of the data (Figure 9). This result means that the model successfully identified all graphene structures containing defects without any false positives. In this case that was seven defect patches out of 455.

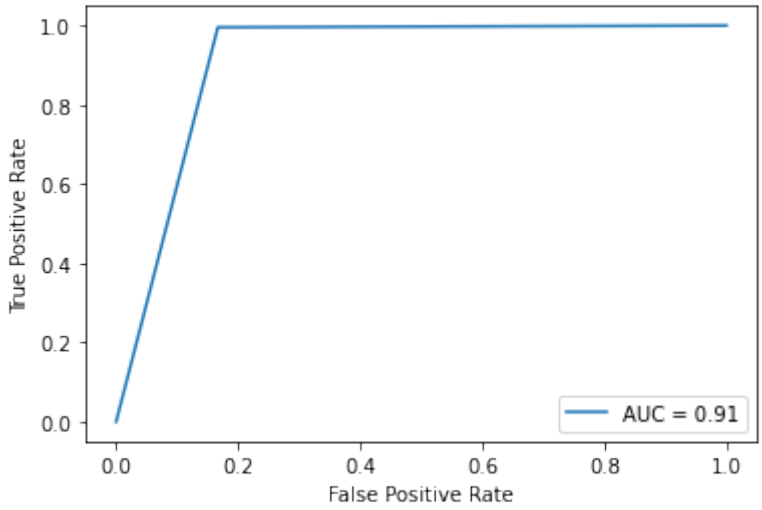

Figure 8: ROC-curve for the Isolation Forest algorithm.

In theory this is a very strong performance and suggests that the model can competently distinguish graphene structures containing defects from those without. However, this result is almost suspiciously good and there is a distinct possibility that overfitting is a factor here. Overfitting emerges as a result of overtraining or an overly complex model. It involves a relationship being fitted too specifically between variables in a certain data space and means that the learned relationship cannot and does not generalise well to external testing sets. This is entirely possible given a fully connected neural network was used and the large number of units included in layer. In other words, the network had such a wide range of different connections available to it, that it may well have optimised to a highly specific and non-generalisable function.

A solution that is commonly used in deep learning is dropout. This is a regularisation technique that involves randomly silencing different units of the network during each round of training, forcing the model to optimise to a more general solution than it would without dropout. This has been shown to improves the performance of neural networks. The proportion of nodes switched off during dropout can be varied and tuned as a hyperparameter to determine the optimum. [11]

Unfortunately, due to time constraints, the model was not tested on a separate external dataset

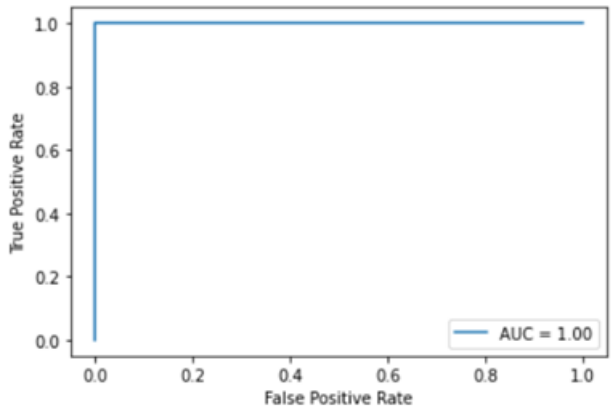

Figure 9: ROC-curve for the Dense Neural Network Implementation

which is the next logical step to take. When such an investigation can be carried out, the hypothesis will be that the model will not generalise well to new data and will perform much more poorly than it does in the results presented here. If that proves to be the case, regularisation will then be tested with the expectation that it will improve model performance.

To train the convolutional autoencoder, 2247 of the perfect examples were used. The remaining 32 were set aside for validation along with the 32 imperfect examples to give a balanced validation set. Due to time and resource constraints the model was trained with 100 epochs (the original method used 600). While the reported loss was low (0.07), the model failed to correctly reconstruct the perfect example images (Figure 6). Tuning of this parameter to permit correct reconstruction, whilst avoiding overfitting, would be required to test the capabilities of this method for classification.

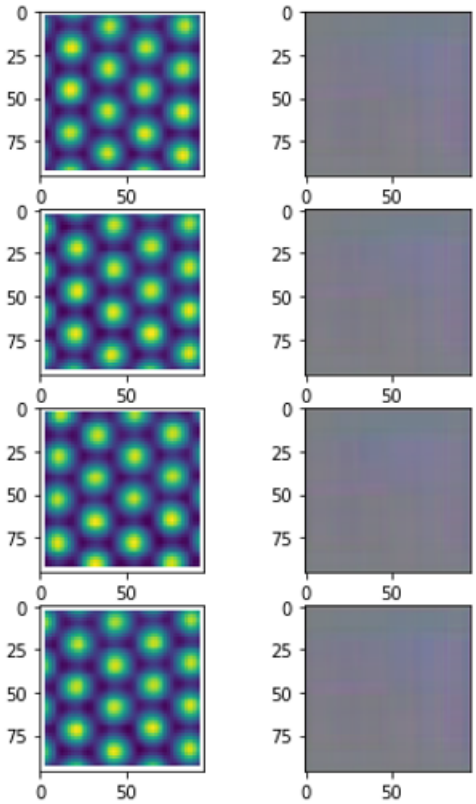

Figure 10: Sample of the original (left) and reconstructed (right) perfect example images for a model trained with 100 epochs.

6 Conclusions & Future Work

Overall, the best performance was given by the Isolation Forest for the witheld test data, with an AUC score of 0.91. However the method performed poorly on the classification of entirely unseen patches, suggesting that the method has been overfit to the training data and does not generalise well. In future this could be rectified by including methods to rectify the imbalance in the training examples. Both the one-class SVM and the isolation forest could potentially be improved by tuning the contamination parameter specifying what % of the data is expected to contain defects.

The dense neural network method was found to give a perfect AUC score of 1.0, which suggests

overfitting. If more time had been available, methods such as dropout could have been deployed to address the imbalance in the training data and improve the model performance.

The convolutional autoencoder initially seemed a promising approach due to its lack of reliance on prior knowledge of what the defect patches will look like. However further training on a higher powered computing resource and tuning of parameters to better suit the specific task at hand would be required to improve image reconstruction and therefore permit its use in classification.

7 Outputs, Data & Software Links

Group Presentation:	https://doi.org/10.5281/zenodo.6779015
GitHub Repository:	https://github.com/RPirie96/AI4SD_group6.git

References

[1] Novoselov KS, Geim AK, Morozov SV, Jiang De, Zhang Y, Dubonos SV, et al. Electric field effect in atomically thin carbon films. science. 2004;306(5696):666-9.

[2] Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, et al. API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning; 2013. p. 108-22.

[3] Schölkopf B, Williamson R, Smola A, Shawe-Taylor J, Piatt J. Support vector method for novelty detection. In: Adv. Neural Inf. Process. Syst.; 2000. p. 582-8.

[4] Liu FT, Ting KM, Zhou ZH. Isolation forest. In: 2008 eighth ieee international conference on data mining. IEEE; 2008. p. 413-22.

[5] Mater AC, Coote ML. Deep Learning in Chemistry. Journal of Chemical Information and Modeling. 2019;59(6):2545–2559.

[6] Chollet F, et al.. Keras. GitHub; 2015. Available from: https://github.com/fchollet/keras.

[7] Ramachandran P, Zoph B, Le QV. Searching for Activation Functions. arXiv; 2017. Available from: https://arxiv.org/abs/1710.05941.

[8] Wells J. Image Anomaly Detection / Novelty Detection Using Convolutional Auto Encoders in Keras Tensorflow 2.0. Medium; 2020. Available from: https://medium.com/@judewells/image-anomaly-detection-novelty-detection-using-convolutional-auto-encoders-in-keras-1c31321c10f2.

[9] Brownlee J. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks; 2019. Available from: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[10] Brownlee J. A Gentle Introduction to Probability Density Estimation; 2020. Available from: https://machinelearningmastery.com/probability-density-estimation/.

[11] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15(56):1929-58. Available from: http://jmlr.org/papers/v15/srivastava14a.html.

List of Figures