AI 4 Science Discovery Network+

Group: 2
Challenge: Task 3 - Detect Defects in Electron Microscopy Images
AI4SD ML Summer School Report
20-24th June 2022

Project Team: Ross J. Urquhart (University of Strathclyde), Chris Woodley (University of Liverpool), Katerina Karoni (University of Edinburgh), Jan Elsner (UCL), Daniel York (Swansea University)

Report Date: 15/07/2022

Group: 2
Challenge: Task 3 - Detect Defects in Electron Microscopy Images
AI4SD-SummerSchool-Series:Report-2
Report Date: 15/07/2022
DOI: 10.5258/SOTON/AI3SD0245
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	2
Challenge Name	Task 3 - Detect Defects in Electron Microscopy Images
Project Dates	20-24th June 2022

2 Project Team

2.1 Project Student

Name and Title	Ross J. Urquhart
Employer name / University Department Name	University of Strathclyde
Work Email	ross.urquhart@strath.ac.uk
Details	First Year PhD student at University of Strathclyde. Studying the use of Machine Learning to accelerate catalyst design for Hydrogen Isotope Exchange reactions.

Name and Title	Chris Woodley
Employer name / University Department Name	University of Liverpool
Work Email	sgcwoodl@liverpool.ac.uk
Details	Postdoctoral researcher at the university of Liverpool. Using computational chemistry to support drug discovery projects, and developing interpretable machine learning models to predict the properties of low molecular weight gels.

Name and Title	Katerina Karoni
Employer name / University Department Name	University of Edinburgh
Work Email	s2090086@ed.ac.uk
Details	Second year of the MAC-MIGS CDT Programme at the University of Edinburgh. Studying Sampling methods and Machine Learning for Molecular Dynamics applications.

Name and Title	Jan Elsner
Employer name / University Department Name	UCL
Work Email	jan.elsner.19@ucl.ac.uk
Details	Third year PhD at University College London. Using methods from computational physics and chemistry to study electron transport in organic semiconductors.

Name and Title	Daniel York
Employer name / University Department Name	Swansea University
Work Email	983045@swansea.ac.uk
Details	First Year PhD student at Swansea University. Studying the use of different computational chemistry methods for the valorisation of biomass into functional bio-based materials.

2.2 Challenge Description

Challenge 3 surrounds the detection of defects in electron microscopy image. Given 48x48 pixel images, the group was tasked with identifying defects in the image based on imperfect C6 rings. The group was given a set of 'perfect' patches and a set of 'defect' patches, Figure 1, and told to measure success with the appropriate metric(s).

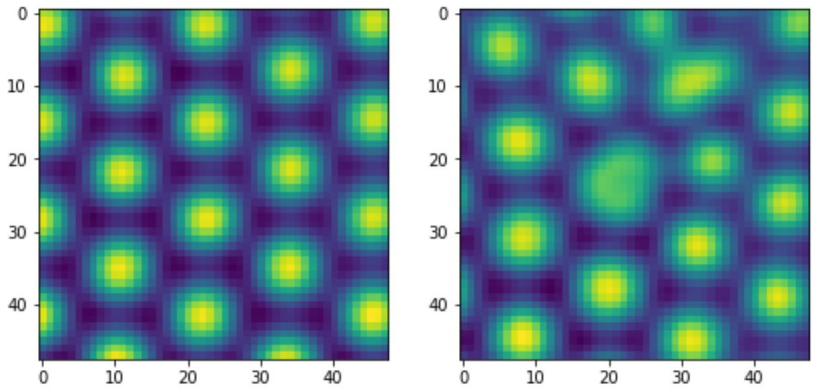

Figure 1: Example of perfect patch (left) and patch with defect (right)

3 Lay Summary

This project is based around the use of AI (artificial intelligence) and ML (machine learning) to identify whether a graphene patch contains defects or not. The classical problem is that of training a ML algorithm to identify whether an image is a cat or a dog, known as an image classification problem. However, the scope of these ML methods are applicable far beyond this, relatively, simple problem and current uses of image classifiers include; classification of arteries and veins in retinal blood vessels [1], classification of land use/land cover from high-res satellite images [2] and for the classification of blood smear images to help diagnose a range of diseases

(e.g. leukaemia and AIDS) [3]. These are but a few examples but they highlight the importance and usefulness of these methods as they allow for machines to perceive images in a similar way to humans by identifying key features (e.g. animal face shapes, blood vessel location and size, etc.). The ML algorithms learn from the initial data they are presented with and this allows for the classification of future data into specific classes (e.g. cat or dog). This is critical for advancement in a range of scientific fields as ML methods and algorithms can provide a quicker and more accurate classification than otherwise possible. For example, in this project, it is much quicker to send thousands of graphene images through a ML algorithm than identifying if a patch is defective or not by manually observing the images.

3.1 Learning types

Before ML algorithms can be utilised to identify defective/perfect graphene patches, they must first be trained. There are a few key types of learning; supervised, unsupervised and reinforcement.

Supervised learning requires labelled input data (see fig. 2) and works by extracting features from each image and subsequently determining different combinations of these features to allow for classification of different images after the initial learning phase. In summary, this type of learning identifies features in a perfect/defective graphene patches and knows that these features belong to the perfect/defective graphene patches since the initial images are labelled (as perfect/defective).

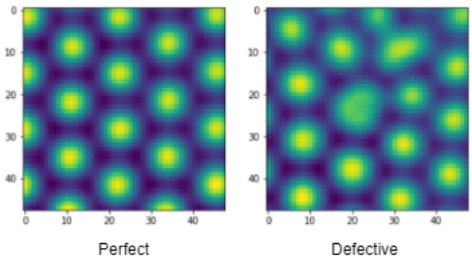

Figure 2: Labelled images for supervised learning.

Unsupervised learning uses unlabelled data and instead groups images in a couple of different ways. One way of grouping the unlabelled data is via 'clustering', this method identifies features in images and then groups the images based on patterns that are consistent among images and attempts to classify images this way (e.g. images of cats and dogs will have different features and these can be separated into different groups even if the algorithm hasn't been initially told whether an image is a cat or dog). Another grouping method is called 'association', this method attempts to identify rules within the data. In the context of a dog this could mean something such as 'if this type of ear is present, typically this type of nose is also present' and a similar rule would also be generated for cats but the features (e.g. ear, nose, etc.) are different for each animal. When applying this graphene, it could be though as: a perfect patch has only a repeating pattern but a defective patch has this pattern but also deviates from it when a defect is present.

Both supervised and unsupervised learning have been used to tackle the problem (section.**??**); however, it is also worth briefly mentioning reinforcement learning. This is where an algorithm is able to correct itself based on inputs, essentially this takes the action of the algorithm and can provide a reward to positive actions which allows for continuous improvement and strengthening of the classification algorithm - this is known as positive reinforcement. Negative reinforcement is also utilised and this stops any action that results in negative effect or doesn't give the correct values/classification, meaning these steps are known to be inherently bad and are subsequently

avoided. Reinforcement learning has been applied in a chemical context to optimize chemical reaction conditions to better improve the performance of physical experiments [4].

3.2 Artificial neural networks

A range of different artificial neural networks were explored to tackle the problem (section.**??**) with the development of an autoencoder, variational autoencoder and convolutional neural network (CNN) all attempted (the theory behind these is explained in more detail in sections.4.0.1, 4.1.1 and 4.2.1, respectively). To briefly introduce these methods; a CNN is fundamentally a series of filters that aims to extract high level features from images (using supervised learning as described in section.3.1) allowing the CNN to identify and classify future images. Autoencoders utilise unsupervised learning and contain and input and output CNN as well as a bottleneck layer in between these layers (fig.3). The basis of this method is to reduce the dimensionality (size compression) of the image information from the input layer into the bottleneck and train the network so that the output layer can reconstruct the image from this compressed image representation as best as possible.

4 Methodology

subsectionAutoencoder

4.0.1 Theory

Autoencoders are a specific type of feedforward neural network used for unsupervised learning. The aim of an autoencoder is to learn a low-dimensional representation of its input data. An autoencoder consists of three components, the encoder, the code and the decoder. The encoder and decoder can have one or more layers. The layer between the encoder and decoder is called the code (hidden layer) and contains the desired lower dimensional representation of our input data. The neural network is designed so that there are less nodes in the hidden layer than in the input and output layers which are of the same size. Thus there is a "bottleneck" in the network, which is crucial for the feature extraction we wish to perform. Specifically, the input is compressed into a lower-dimensional "code" and then the output is reconstructed from this representation. The process of learning a low-dimensional representation of our data is called feature extraction. The network is trained so that the output layer best approximates the input layer.

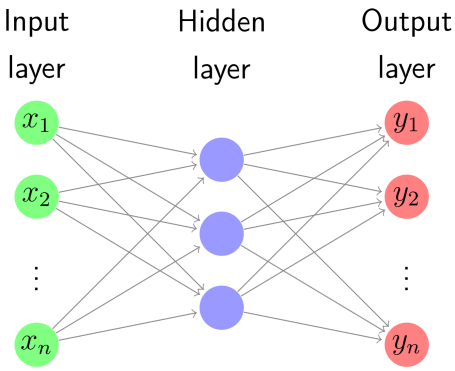

Figure 3: Autoencoder architecture

For data $X = \{x_1, ..., x_n\}$, we let $\mathcal{X} \subseteq \mathbb{R}^D$ be our data space and let $\mathcal{H} \subseteq \mathbb{R}^d$ be our bottleneck (hidden layer) space. The autoencoder consists of two parts, the encoder and the

decoder which can be defined as transition functions $F_{enc} : \mathcal{X} \to \mathcal{H}$ and $F_{dec} : \mathcal{H} \to \mathcal{X}$. The optimal transition functions F^*_{enc}, F^*_{dec} are defined as follows:

$$F^*_{enc}, F^*_{dec} = \arg \min_{F_{enc}, F_{dec}} \sum_{x \in X} ||x - (F_{dec} \circ F_{enc})x||_2^2, \tag{1}$$

where F^*_{enc} and F^*_{dec} are optimised over the weights and biases and X is our training data set. Our loss function L is defined as

$$L(x, F_{enc}, F_{dec}) = \sum_{x \in X} ||x - (F_{dec} \circ F_{enc})x||_2^2.$$

We can then use $F^*_{enc} : \mathcal{X} \to \mathcal{H}$ as a feature map / low-dimensional representation of our data. Figure.1 shows a simple autoencoder with only one layer in the encoder and decoder. However, in the general setting the encoder and decoder can consist of more than one layers. Autoencoders can be trained using standard gradient descent based optimisation algorithms.

4.0.2 Application

We next train an autoencoder on our defective and perfect patches data. As mentioned earlier, this will result in the hidden layer encoding a lower dimensional representation of the data passed into the autoencoder. By 'removing' the decoder part of the network after training and looking at the features of the code layer we can obtain the following histogram of features for defective and perfect patch data. Note: The feature plotted here is the output of the first node of the code layer (figure.4). We can see that the network can - to an extent - differentiate between defective and perfect patches. However, looking at the reconstructed patches we see that even though the network has learned to output an image similar to the input one, the reconstruction for defective and perfect patches is the same (figure.5). Therefore, the trained network cannot differentiate well between the two classes of data. This is partly because we are dealing with image data and pixel connectivity information is lost when using a normal autoencoder. Using a convolutional autoencoder enables us to perform a much more faithful reconstruction of perfect and defective patches (figure.6).

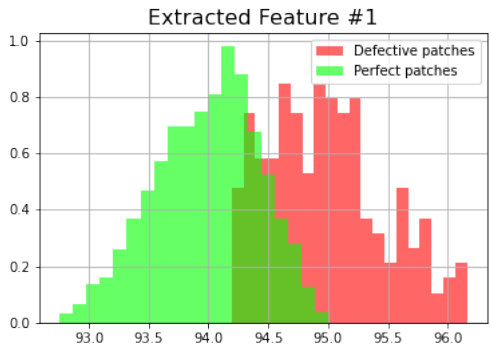

Figure 4: Reconstruction of perfect patches.

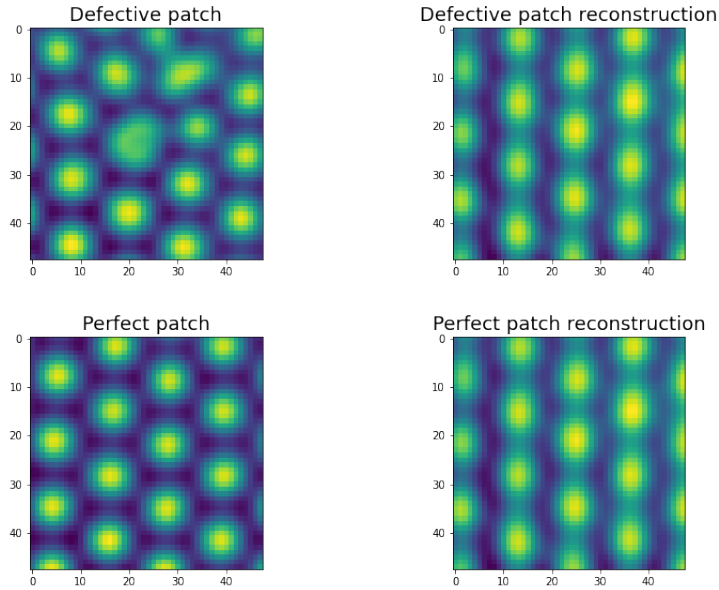

Figure 5: Reconstruction of defective and perfect patches.

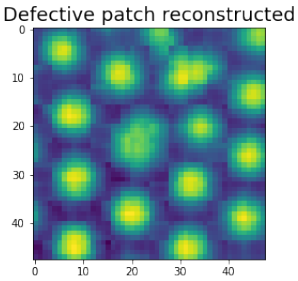

Figure 6: Reconstruction of defective patch using convolutional autoencoder.

4.1 Variational Autoencoder

4.1.1 Theory

In this model, the latent code, $\mathbf{z}$, has a prior distribution $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z})$. We aim to maximize:

$$\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}) = \int \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z})\mathrm{d}\mathbf{z}. \tag{10}$$

This integral is generally intractable. One approach to proceed could be to approximate the integral using Monte Carlo methods. This would involve sampling $\{\mathbf{z}_i\}_{i=1}^N$ from the prior $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z})$, modelling $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with a neural network and approximating:

$$\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_i). \tag{11}$$

However, the high dimensionality of $\mathbf{x}$ means that many samples are needed to get a reasonable approximation: it is also intractable to compute $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ for every $\mathbf{z}$. Instead, a recognition model is used, $\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, which is trained to give high probability $\mathbf{z}_i$ values, such that fewer samples are needed. We are modelling the true (unknown) posterior, $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, with $\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, which in turn may be modelled as a Gaussian with mean, $\boldsymbol{\mu}$, and variance, $\boldsymbol{\sigma}^{\mathbf{2}}$, which are

determined by the outputs of a neural network. Overall, we wish to maximize $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x})$ while simultaneously ensuring that $\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ accurately approximates $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. This is achieved by maximizing the evidence lower bound, defined by:

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \log(\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x})) - \text{D}_{\text{KL}}(\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})), \tag{12}$$

where D_{KL} is the Kullback—Leibler divergence, defined for continuous random variables P and Q with probability densities $p(x)$ and $q(x)$ respectively as being:

$$\text{D}_{\text{KL}}(Q||P) = \int_{-\infty}^{\infty} q(x) \log(\frac{q(x)}{p(x)}) \text{d}x. \tag{13}$$

Intuitively, the Kullback—Leibler divergence measures how similar two distributions are. To see where ELBO comes from and how it may be maximized, we may use Bayes' theorem to write

$$\log(\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) = \mathbb{E}_{z\sim\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}[\log(\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}))] - \text{D}_{\text{KL}}(\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z})) + \text{D}_{\text{KL}}(\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})). \tag{14}$$

The first term on the right hand side may be computed by sampling (decoder network), the second term has a closed form solution and the third term is intractable since $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is intractable. However, using the property that the Kullback—Leibler divergence is always greater than 0, we obtain a tractable lower bound on $\log(\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))$, which is what we wish to maximize overall. Rearranging Eqn. 14, we obtain

$$\begin{aligned}\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \log(\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) - \text{D}_{\text{KL}}(\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) \\ &= \mathbb{E}_{z\sim\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}[\log(\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}))] - \text{D}_{\text{KL}}(\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z})).\end{aligned} \tag{15}$$

The second line of Eqn. 15 is tractable and is therefore something we can maximize. We wish to find $\boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$ such that

$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \text{argmax}_{\boldsymbol{\theta},\phi} \sum_{i=1}^{N} \text{ELBO}(\mathbf{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}), \tag{16}$$

Note that using a Gaussian prior $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{z}) \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ and recognition model $\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^{\mathbf{2}}\mathbf{I}))$, the Kullback—Leibler divergence may be expressed analytically as

$$-\text{D}_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^{\mathbf{2}}\mathbf{I}))||\mathcal{N}(\mathbf{0}, \mathbf{I})) = \frac{1}{2}\sum_{i=1}^{l}(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2), \tag{17}$$

where l is the number of latent dimensions.

One problem that arises is that sampling from $\mathbf{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is not a differential operation, therefore we cannot use standard gradient descent. To fix this, we introduce the reparameterization trick, which allows us to preserve the probability distribution of $\mathbf{z}$ whilst allowing us to back-propagate through a deterministic node. This involves reparameterizing the latent code as

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}, \tag{18}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ is the Hadamard (element-wise) product. Sampling is now from the standard Gaussian, $\boldsymbol{\epsilon}$, and gradients are able to propagate through $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, since these are now deterministic.

4.1.2 Application

Our idea was to learn the latent distribution of perfect samples, $\mathbf{z}_{\text{perfect}}$, using a variational autoencoder. By then running all of the data through our encoder network, we should be able to classify perfect vs non-perfect samples but checking whether the resulting latent distribution conforms to $\mathbf{z}_{\text{perfect}}$.

We implemented a convolutional variational autoencoder in PyTorch. The encoder and decoder consisted of 5 convolutional layers and a fully connected network map to the latent space. We successfully trained our network on perfect samples. By epoch 20-30, our network was able to successfully reproduce training images (figure.7).

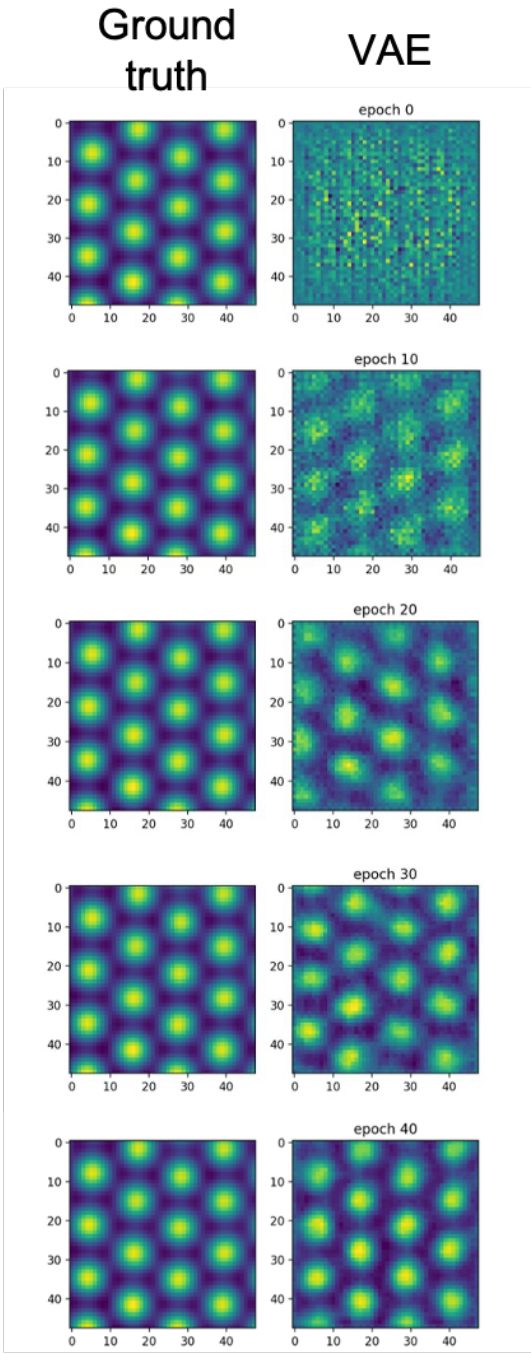

Figure 7: VAE reproduction of a perfect patch at different training epochs.

Unfortunately, we did not have time to check the latent distributions of our data.

4.2 CNN Binary Classifier

4.2.1 Theory

Convolutional Neural Networks (CNNs) are a subset of neural networks that are designed to work with grid-like data such as images. Pictures on a screen can be thought of as broken up into a grid where each pixel has a value related to the brightness and colour.

A convolutional network is comprised of convolutional layers. each of these layers passes a convolutional filter over a matrix. The filter performs a dot product of a subset of values from the previous layer. This means convolutional layers are smaller than the input image but richer in the detail they contain. For example, a 3x3 convolutional filter can take a 3x3 subset of the image and perform a dot product, producing one value. This means if the image is a 5x5 grid, a 3x3 filter will compress the image into a 3x3 matrix. The filter, or kernel, is passed across the input image building an image representation as described above. Pooling serves to reduce the size of the convolutional layer and thus decrease computational cost. It is also good at identifying important features. Max pooling, used below, returns the maximum value of the kernel space. *E.g.* if the kernel had values [32, 16, 8, 4] max pooling would return 32.

CNNs also make use of fully connected linear layers and activation functions as these help to map the representation between input and output.

Uses of CNNs lie in object detection, image captioning and semantic segmentation applications.

4.2.2 Application

Alongside unsupervised approaches we investigated the use of a supervised binary classifier to distinguish between perfect and defect samples. We opted to train a CNN binary classifier owing to their widespread use in image classification problems. The work on this approach focused on the design of CNN architecture for the given dataset.

The combined labelled dataset was split into training and testing sets in a ratio of 0.8:0.2. We implemented two CNN architectures to extract features from the labelled perfect and defect patches: A) three convolutional layers (conv2d and MaxPool2d), and three fully connected layers from a pre-existing architecture; B) a CNN with 2 convolutional layers separated by a maxpooling layer and followed by three fully connected linear layers.

The CNN based on architecture A was exposed to the training data for 50 epochs and evaluated on the test data. Neither CNN architecture was exposed to an augmented dataset due to time constraints.

5 Results

5.1 CNN Binary Classifier

We evaluated the CNN built on architecture A on the test-set data. We opted to use the AUC-ROC scoring metric as opposed to accuracy; due to the massive imbalance in the data accuracy would provide an overestimation of the quality of our model. After training for 50 epochs the training-set AUC-ROC score was 0.5 suggesting that the produced model was no better than chance at detecting defect patches (figure.8).

This may be due to the CNN being unable to pick out features to classify, or more likely is that the dataset requires augmentation to sufficiently learn features of defect patches.

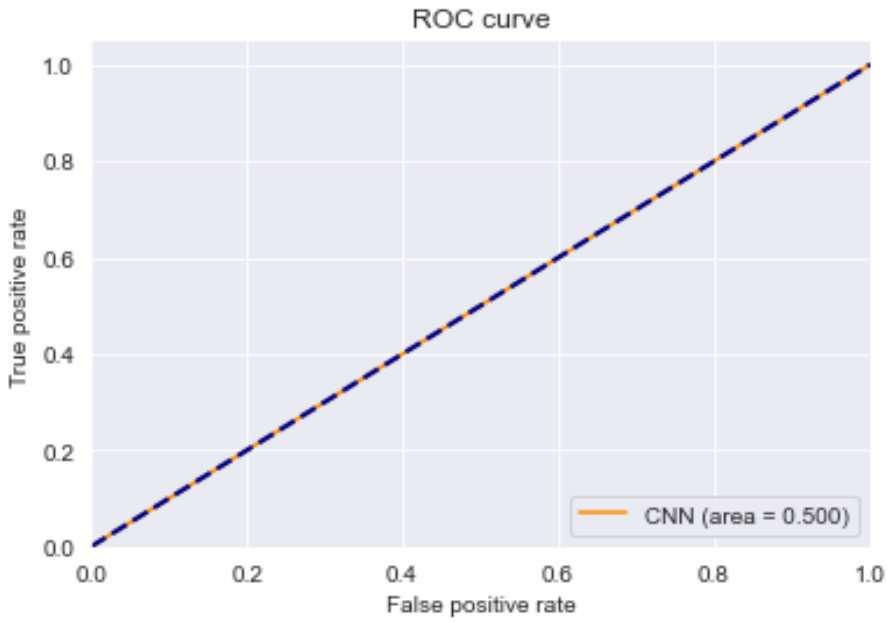

Figure 8: ROC curve of the classification performance of CNN A on the test-set of patches.

Architecture B was unable to produce any results due to constraints time constraints.

6 Conclusions & Future Work

6.1 Conclusions

In conclusion, we attempted to use supervised and unsupervised machine learning methods for defect detection in electron microscopy images of graphene sheets. Our supervised approach involved the development of CNN binary classifiers, while our unsupervised approaches involved the development of an autoencoder, a convolutional autoencoder and a variational autoencoder. Evaluation of the CNN binary classifier determined that this was no better than chance at detecting defects. Though not applied to classification, the autoencoders were able to produce faithful reproductions of graphene patches. Analysis of extracted features from one autoencoder does show a difference in features extracted from perfect patches and defect patches.

6.2 Future Work

Future work on this project would include full augmentation of the dataset in order to present a more rounded view of the data. Once augmented, it is hoped that the full dataset could be used to train the CNN architecture to full effect.

References

[1] Mookiah MRK, Hogg S, MacGillivray TJ, Prathiba V, Pradeepa R, Mohan V, et al. A review of machine learning methods for retinal blood vessel segmentation and artery/vein classification. Medical Image Analysis. 2021 2;68:101905.

[2] Ramaswamy SK, Ranganathan MBA. Land use land cover classifcation using local mutiple patterns from very high resolution satellite imagery. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives. 2014;XL-8:971-6.

[3] Khan S, Sajjad M, Hussain T, Ullah A, Imran AS. A review on traditional machine learning and deep learning models for WBCs classification in blood smear images. IEEE Access. 2021;9:10657-73.

[4] Zhou Z, Li X, Zare RN. Optimizing Chemical Reactions with Deep Reinforcement Learning. ACS Central Science. 2017 12;3:1337-44.

List of Figures