mathematics

Article

Anomaly Detection in the Molecular Structure of Gallium Arsenide Using Convolutional Neural Networks

Timothy Roche *, Aihua Wood *, Philip Cho and Chancellor Johnstone

Department of Mathematics & Statistics, Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, OH 45433, USA
* Correspondence: timothy.roche@afit.edu (T.R.); aihua.wood@afit.edu (A.W.)

Abstract: This paper concerns the development of a machine learning tool to detect anomalies in the molecular structure of Gallium Arsenide. We employ a combination of a CNN and a PCA reconstruction to create the model, using real images taken with an electron microscope in training and testing. The methodology developed allows for the creation of a defect detection model, without any labeled images of defects being required for training. The model performed well on all tests under the established assumptions, allowing for reliable anomaly detection. To the best of our knowledge, such methods are not currently available in the open literature; thus, this work fills a gap in current capabilities.

Keywords: electron microscope; convolutional neural networks (CNNs); anomaly detection; principal component analysis (PCA); machine learning; deep learning; neural networks; Gallium Arsenide (GaAs)

MSC: 68U10

check for updates

Citation: Roche, T.; Wood, A.; Cho, P.; Johnstone, C. Anomaly Detection in the Molecular Structure of Gallium Arsenide Using Convolutional Neural Networks. *Mathematics* **2023**, *11*, 3428. https://doi.org/10.3390/math11153428

Academic Editor: Huawen Liu

Received: 21 June 2023
Revised: 28 July 2023
Accepted: 2 August 2023
Published: 7 August 2023

1. Introduction

Electron microscopes use electrons to “take pictures” of incredibly small areas; these areas so small that one can see the atomic structure of the material. Gallium Arsenide (GaAs) is a semiconductive material extensively used in a wide variety of modern day technology, such as cell phones, solar panels, and much more [1]. Minor flaws and imperfections at the molecular scale can impact the properties of GaAs [2]. However, it is difficult to detect these defects, as they happen at an incredibly small scale. Ref. [3] developed a machine learning technique to detect defects within GaAs using simulated data, as real data were not available. This paper aims to extend the work of [3] with the construction of a machine learning classifier trained on actual images of GaAs.

The GaAs images used in this work were collected with a Titan 80–300 image spherical aberration-corrected transmission electron microscope operated at 300 kV, hereby referred to as “the Titan”. Similar to [3], we employ a combination of principle component analysis (PCA) and convolutional neural networks (CNNs) to detect defects within GaAs images. CNNs are a type of neural network that has been shown to perform well on defect detection tasks across a variety of fields [4–6].

PCA is a method frequently used to reduce the dimensionality of data [7]. In the data sciences, large data sets often have correlated or redundant factors. When utilizing PCA, the principle components are calculated with linear combinations of covariates; these principle components are then used as new features in the analysis. While there is a slight loss of information, increasing the number of principle components reduces the percentage of information lost. Once a PCA is trained, it can be used to calculate the principle components for a specific sample of data. From this point, a PCA reconstruction of an image can be created. The idea is that the PCA reconstruction resembles the general pattern of the original data while removing or ignoring anomalies in the data set [8].

The rest of the paper is organized as follows: First, we discuss the data used in this project, how they are obtained, and certain limitations that we must address. Section 3 explains the methodology implemented to create, train, and test the model, with the results being discussed in Section 4. The paper is concluded in Section 5.

2. Data

All the data used in the training of our model are created from a single picture taken by the Titan. This image is of a "defect free" sample of GaAs (hereby referred to as the substrate image). While not 100% defect-free, the sample was grown in a way that made it as close to defect-free as possible. For training, we use the 40-square-nanometer GaAs image shown in Figure 1. To provide a sense of scale, a single human hair is generally between 40,000- and 120,000-nanometer-thick.

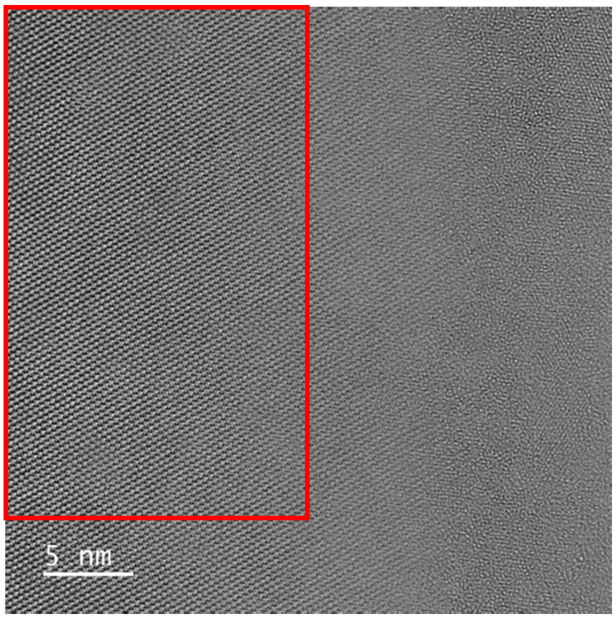


Figure 1. Example of image from the Titan. The section used to generate data is highlighted in red.

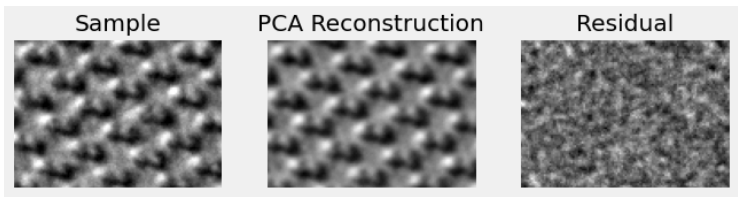
A myriad of issues arise in the electron microscope images of GaAs samples. Even the "defect free" samples created are not fully defect-free; the number of defects is far lower than what would usually be seen, but it is likely that there are still defects present in the samples. Other issues can be caused when the samples are prepared for imaging. The samples must be ground down to very small thickness, and this process can cause a minor warping of the samples.

Yet another issue is alignment. In simulated data, the alignment of samples can be fully controlled and aligned to a precise, preferable angle. With real-world images, this level of control is not possible. It is important to remember that we are dealing with images of atoms; while the structure of GaAs is quite consistent, there is very little indication of the alignment of the atoms before the images are captured. This often leads to the data being slightly rotated or skewed when compared with the simulated data.

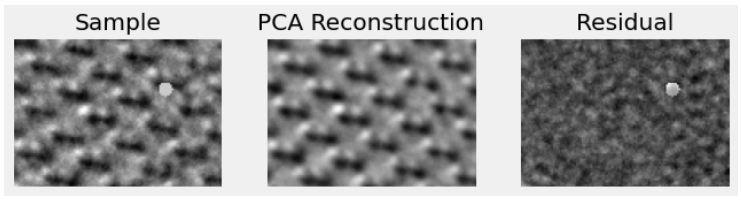
The state of the Titan also impacts image quality, as the characteristics of the electron beam affect the quality of the image. Also, due to the small scale, the Titan is very sensitive to outside interference. People walking nearby and even the operator breathing can affect the quality of the image. Due to these factors, obtaining high-quality images is difficult.

These issues render parts of the training image unusable, with warping from the sample preparation process and the scale in the corner being the main contributors. Therefore, only part of the full image is used. The 1800-by-1024-pixel section identified with the red bounding box in Figure 1 is the substrate section used for the remainder of this paper.

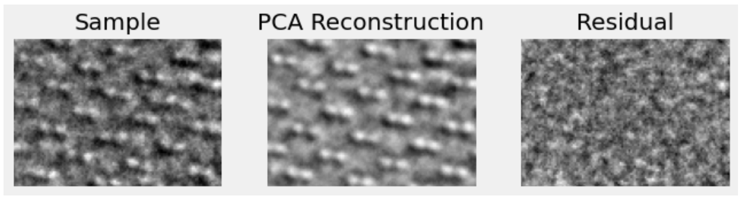
Samples from this subsection are then taken to create the data set used to train the model. These samples are 118-by-84-pixel rectangles chosen at random from the 1800-by-1024 substrate section, allowing over 1.5 million unique samples to be created from a single electron microscope image. An example of one of these samples is shown in the top-left corner of Figure 2.



(**a**) Defect-free sample, reconstruction, and residual



(**b**) Simulated defect sample, reconstruction, and residual



(**c**) Actual defect sample, reconstruction, and residual

Figure 2. Examples of the data used in training and testing the model. From right to left: section taken from microscope image, PCA reconstruction of section, and residual.

During the training of the model, these samples are chosen to be either a defect sample or a non-defect sample, with a specified probability. For defects, a simulated defect is added to the section at a random location. The process for simulating defects and further data preparations are discussed in further detail in Section 3.

The reliance on simulated defects is due to the fact that we do not have any labeled images of actual defects. The best we have is an image that we know has defects in it but with no idea how many or where they are. Not only does this force us to generate simulated data for training the model, but it also makes validating the model nearly impossible, because there is no way to tell if the predictions are correct.

To overcome this issue, the assumption followed for this paper is that each 84-by-118 section from the defect image has at least one defect in it. Based on information gathered when the sample was grown, the subject matter experts are confident this assumption is true. An example of one of these defect samples can be seen in the bottom left of Figure 2.

Testing Data

Once the model is trained, we perform three different tests, each using a different data set. The first of these tests is a model validation test and uses defect-free, or substrate, samples with and without simulated defects. Generators are used to create this test set, and about 30% of the data contain simulated defects.

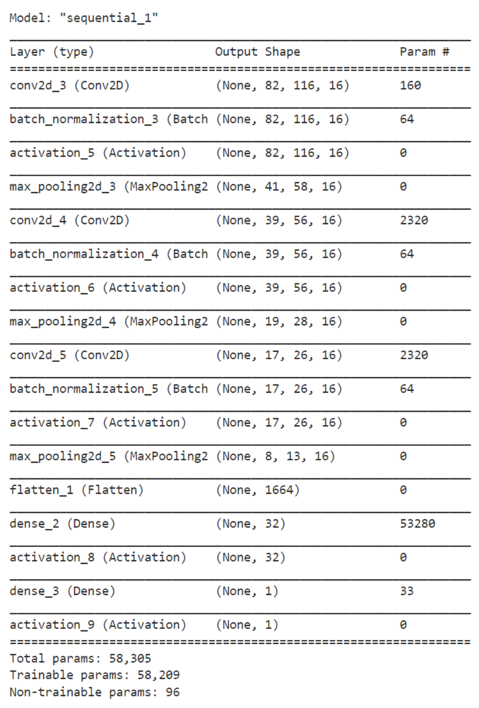
The second test is to ensure that the model actually detects defects and does not just detect samples taken from a different image. The data for this test do not contain any defects, neither simulated nor real. Half the samples for this are taken from the original data set, and the other half is taken from a different picture of "defect free" GaAs. Ideally, the models predict most of these as defect-free, but they may find a few defects, as even the “defect free” images may still contain some defects.

The third test is to determine if the model can find actual defects. We currently do not have a way of detecting and labeling real defects in images, so we do not have a labeled set of defects for this test. Instead, we have an image of GaAs that definitely contains defects, we just do not know where. Based off information gathered when the GaAs sample was created, the subject matter experts are confident that each 84-by-118 section from this image contains at least one defect. These images comprise half of the data for the third test, and ideally, the model predicts all of these samples as containing defects. The other half of the data are original defect-free samples without simulated defects.

3. Methodology

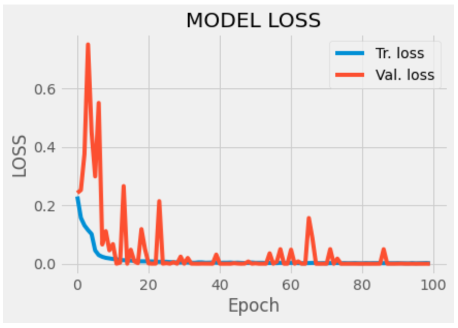
The prior work on this project using simulated data [3] performed well but was never applied to real-world data. The actual data collected with the electron microscope suffered minor distortion from sample preparation and other sources. This caused the models trained on the simulated data to detect this distortion as defects rather than random noise. The models had to be retrained using real data to compensate for these other non-defect-related imperfections. Regardless, the methodology is quite similar to the prior work by [3], with changes to incorporate the recently available data.

The three main pieces are a CNN, PCA, and a data generator. The CNN used is a standard AlexNet CNN [9], which is a type of neural network commonly used for defect detection. A summary of the model can be found in Figure 3a, with loss and accuracy graphs from training in Figure 3b and Figure 3c, respectively. While this style of CNN is commonly used for defect detection tasks, our CNN is trained on an unusual set of data. The samples discussed in the previous section are not directly used in training. Before the samples can be used in training, they go through a process involving a PCA reconstruction.

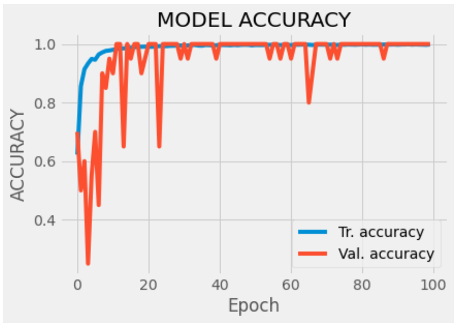


```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 82, 116, 16)       160
_________________________________________________________________
batch_normalization_3 (Batch (None, 82, 116, 16)       64
_________________________________________________________________
activation_5 (Activation)    (None, 82, 116, 16)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 41, 58, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 39, 56, 16)        2320
_________________________________________________________________
batch_normalization_4 (Batch (None, 39, 56, 16)        64
_________________________________________________________________
activation_6 (Activation)    (None, 39, 56, 16)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 19, 28, 16)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 17, 26, 16)        2320
_________________________________________________________________
batch_normalization_5 (Batch (None, 17, 26, 16)        64
_________________________________________________________________
activation_7 (Activation)    (None, 17, 26, 16)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 13, 16)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1664)              0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                53280
_________________________________________________________________
activation_8 (Activation)    (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
_________________________________________________________________
activation_9 (Activation)    (None, 1)                 0
=================================================================
Total params: 58,305
Trainable params: 58,209
Non-trainable params: 96
```

(**a**) CNN: summary



(**b**) Loss by epoch



(**c**) Accuracy by Epoch

Figure 3. Neural network structure and training results.

The PCA used for this reconstruction is trained on the same 84-by-118-pixel samples that are discussed in the previous section of this paper. Once trained, this PCA model has

200 components and accounts for over 90% of the variance (see Figure 4). This PCA is then used in the last piece: the data generator.

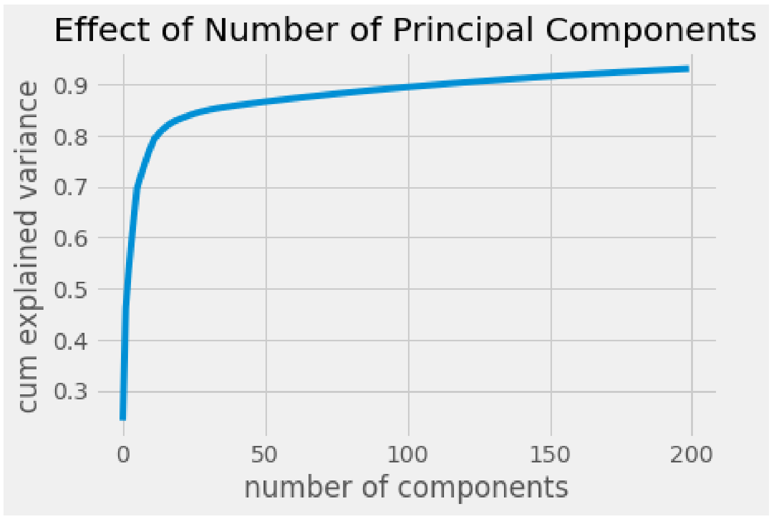


Figure 4. Training PCA: variance explained per number of components.

The data generator is the piece that produces the data used to train the model. This is where the simulated defects are added and a PCA reconstruction is used to generate a batch of training samples. Section 3.1 covers the data generation process in further detail.

The rest of this section is divided into two subsections. First, we will discuss the training of the neural network and how the simulated defects are created. The second subsection will cover how the model is tested and validated.

3.1. Model Training

The neural network is trained for 100 epochs containing 100 batches of 100 samples. Unlike many neural networks, our CNN is not trained using a conventional training set. Instead, a data generator is used, due to the nature of the data available. Pseudo-code for the generator is provided in Algorithm 1 and is discussed further below.

Algorithm 1: Data Generator

```
function Generator($substrate\_array, pca$);
Input   : 1800 by 1024 section from substrate image, trained PCA
Output  : Batch of 100 training images
for $i = 1$ to $100$ do
    sample = Random 84 by 118 section of substrate image;
    if $rand() > defect\_rate$ then
        reconstruction = PCA reconstruction of sample;
        residual = sample - reconstruction;
        Add residual to data set, labeled as “non-defect”;
    else
        Add simulated defect in a random location on the sample;
        reconstruction = PCA reconstruction of sample;
        residual = sample - reconstruction;
        Add residual to data set, labeled as “defect”;
    end
end
Yield batch of 100 training samples
```

The generator requires inputs of the substrate section and the training PCA and yields a batch of 100 samples. First, a random 84-by-118-pixel sample is taken from the substrate image. From there, the sample is randomly chosen to be either a defect image or a defect-free image. The probability of choosing a defect can be changed but is set at 50% for training. If the sample is chosen to be defect-free, its PCA reconstruction is created and subtracted from the sample, providing the residual, which is then added to the training set and labeled as defect-free.

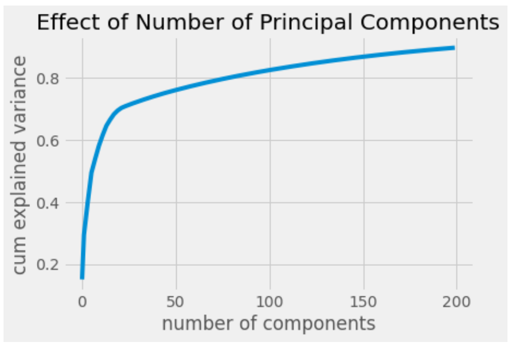
However, if the sample is chosen to contain a defect, a simulated defect is added at a random location. For this model, the simulated defects are circles of three to five pixels in radius, with a gray-scale color value of 0.8 (light gray), but the size, shape, and color of the defects can be changed at will. A PCA reconstruction of the simulated defect sample is then created and subtracted from the simulated defect image. The resulting residual is then added to the training set and labeled as a defect. A sample with a simulated defect, its PCA reconstruction, and the residual can be seen in the middle row of Figure 2. These residual images are what is used to train the CNN.

This process is repeated 100 times to create a batch of 100 residual images. For each epoch of training, 100 of these random batches are generated, and the model is trained for 100 epochs. The validation data are generated in the same way, except the validation batch size contains 20 samples.

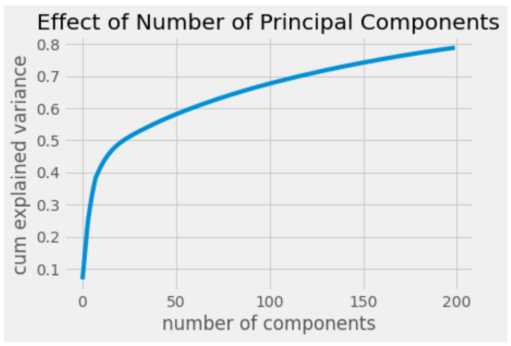
3.2. Model Testing and Validation

Three different criteria are used to evaluate the performance of the model, using data from three separate pictures taken with the Titan. The first test is to ensure the model trains properly. This test uses data simulated in the same way as the training set. The second test uses data from a different image of the substrate (defect-free) GaAs to ensure that other defect-free samples are not predicted as containing defects. The final test uses samples from an image of GaAs containing defects, under the assumption that each sample taken from the image contains at least one defect. This is the test to determine whether or not we can detect actual defects. The results of these tests are discussed in the next section.

Originally, the PCA calculated on the training substrate image was used to create reconstructions for all the data used in the testing of the model. However, when the models were tested, all samples taken from any set other than the training set were predicted to contain defects. This indicated that the training PCA was not applicable to the other data sets. In response to this discovery, a PCA model is trained for each different set of data used in the testing process. The second substrate PCA explains about 90% of the total variance, while the defect PCA explains about 80%. The variance explained versus the number of components graphs for these additional PCA models are shown in Figure 5.



(**a**) Defect-free test PCA

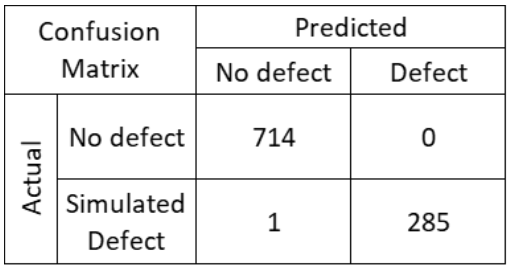


(**b**) Defect test PCA

Figure 5. Test PCAs: variance explained per number of components.

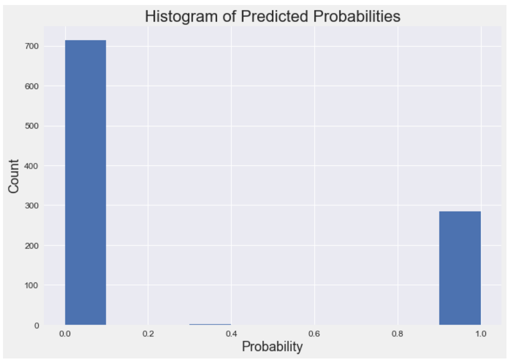
4. Results

As discussed in Section Testing Data, three tests were performed on the model, with each test using a different data set and checking different things. The first test was a model validation test. Data similar to the training data, that is, data with simulated defects, were fed into the trained model. The purpose of this test was to ensure that the model trained properly. The defect ratio was also lowered for this test set to ensure that it had no impact on model accuracy. As seen in the confusion matrix shown in Figure 6a, the model performed well in this test, predicting only one simulated defect as not a defect. The distribution of the predicted defect probabilities is shown in Figure 6b.



Confusion Matrix		Predicted	
		No defect	Defect
Actual	No defect	714	0
	Simulated Defect	1	285

(**a**) Validation test: confusion matrix



(**b**) Validation test: prediction distribution

Figure 6. Outputs from the validation test. (**a**) Confusion matrix: This test had 99.9% accuracy, with only one simulated defect being classified incorrectly. The cause of this is not clear, but it is not a major concern. (**b**) Distribution of predicted probabilities: A histogram showing the number of samples with the predicted probability of a defect. Probabilities above 0.5 were predicted as defects.

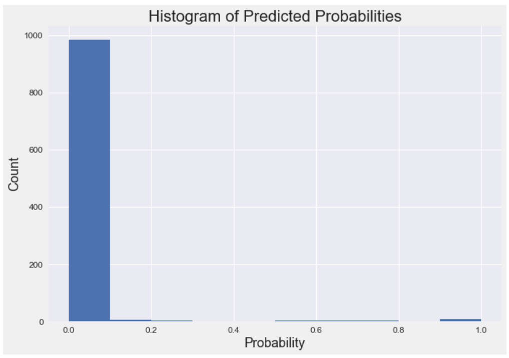
The next test checked for false positives. The data for this test were from a different substrate sample, meaning that there should have been few to no defects in the sample. This test was to check if the model actually predicted defects or if it detected something else. Thus, the "confusion matrix" (shown in Figure 7a) for this test is not a typical confusion matrix. The top row concerns samples from the original "defect free" substrate image used in training, while the second row concerns samples from a different "defect free" image. The model performed well in this test, only predicting eleven samples from the second substrate image as having defects. These predictions could even be true, since the sample, while as close to perfect as possible, may still have contained some defects. The distribution of the predicted probabilities is shown in Figure 7b.

We note that in this test, there were multiple instances of high estimated probabilities of defects, i.e., estimated probabilities greater than 0.5. The exploration of the substrate images resulting in these high estimated probabilities is of interest to us. However, this is out of the scope of the current work.

The final test was performed on samples from the test image containing defects. Recall that the assumption for this test set was that each sample contained at least one defect. Under this assumption, the model performed perfectly. To confirm this, the test was performed ten times, each on a different random set of samples. All ten tests had 100% accuracy. A confusion matrix and a prediction distribution histogram from one of these tests are shown in Figure 8.

Confusion Matrix		Predicted	
		No defect	Defect
Actual	Training Substrate	489	0
	Test Substrate	491	11

(**a**) Substrate test: confusion matrix

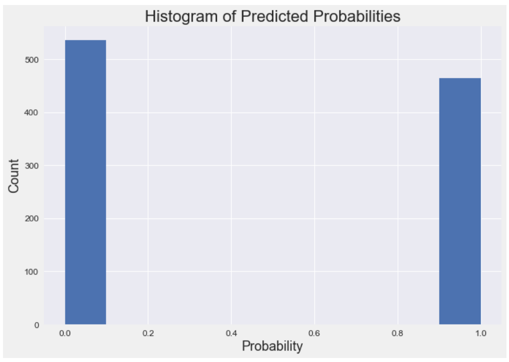


(**b**) Substrate test: prediction distribution

Figure 7. Results of the substrate test. (**a**) Confusion matrix: The expected result was no defects for both sets. However, it is likely that both "defect free" images had some defects, so the predicted defects could be true defects. No defects were detected in the training substrate image because it was used in training. (**b**) Distribution of predicted probabilities: A histogram showing the number of samples with the predicted probability of a defect. Probabilities above 0.5 were predicted as defects.

Confusion Matrix		Predicted	
		No defect	Defect
Actual	No defect	536	0
	Defect	0	464

(**a**) Defect test: confusion matrix



(**b**) Defect test: prediction distribution

Figure 8. Results of the defect test. (**a**) Confusion matrix: This test had 100% accuracy, correctly predicting all samples from the defect sample as defects and all samples from the defect-free image as no defects. (**b**) Distribution of predicted probabilities: A histogram showing the number of samples with the predicted probability of a defect. For this test, all non-defect samples had a predicted probability under 0.1, and all defect samples had a predicted probability above 0.9.

4.1. Discussion

The results from this set of tests are promising, and the model performed as we hoped. However, there are certain gaps in our validation process that we cannot currently fill. The first issue is the lack of a baseline model. As this is the first attempt to accomplish this specific task, there does not exist another model to use as a comparison. As work continues in this area, this model can be used as a baseline in the future. The other issue is the lack of labeled data, thus certain assumptions must be made, as must be done for most real-world applications. The two main assumptions we made were that the substrate images were defect-free and that other images of GaAs contained many defects. The subject experts are confident that our assumption that every 118-by-84 section from the "defect" image contains at least one defect is correct.

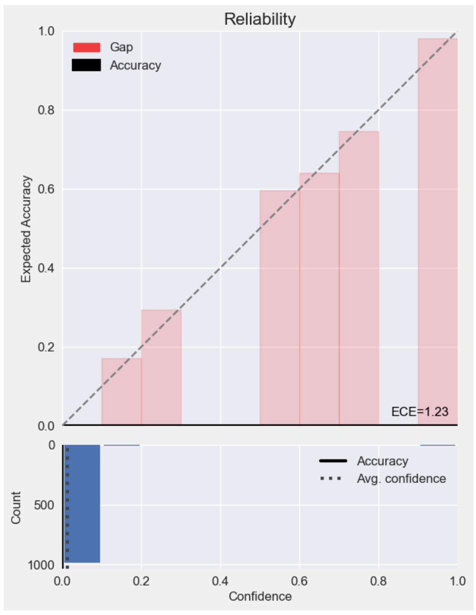
That being said, the model performed well in all tests under our assumptions. It is still possible that the model detected something other than defects in the defect test data, leading to these results. As we acquire more data, this will allow for more options for testing and validating the model. Other validation methods are currently being explored and will be reported in a subsequent paper.

4.2. Calibration

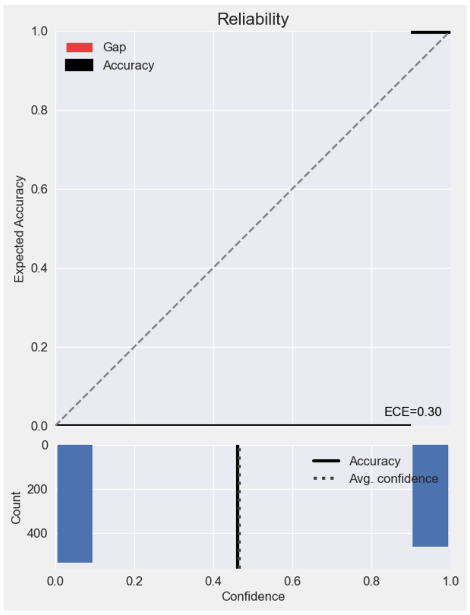
Given that the output of our CNN is an estimated probability of a defect on a particular substrate image, it is also helpful to explore the performance of these probability estimates further than with accuracy only (or even other metrics like sensitivity, specificity, or F1). Ideally, probability estimates should be well *calibrated*. Perfect calibration occurs when

$$\mathrm{E}_{\hat{p}}\left[\left|\mathbb{P}\left(\hat{z}=z|\hat{p}=p\right)-p\right|\right]=0, \quad (1)$$

where E_X is the expectation of some random variable X and $\mathbb{P}$ is a (conditional) probability mass function. In our case, z is the observed class, i.e., defect-free or defected, while $\hat{z}$ is the predicted class. $\hat{p}$ is a probability estimate for the predicted class, and p is the true class probability. With Equation (1), the implication for defect identification in our substrate images is that our estimated probabilities are reflective of the true underlying probabilities of a defect. As an example, we should observe defects, say, in 10% of our observations ("relative frequency") where we estimate a 10% defect probability. We can assess calibration in practice with the use of metrics like empirical calibration error (ECE) or with the use of reliability plots. We point the interested reader to [10] for more details on these approaches. We include reliability plots for our two test sets in Figure 9.



(**a**) Substrate test: reliability plot



(**b**) Defect test: reliability plot

Figure 9. Reliability plots. (**a**) Reliability for substrate test. (**b**) Reliability for defect test.

We note the two plots in Figure 9 provide very contrasting results. The poor calibration performance of the substrate test set is inherently due to its lack of defects; we have a range of estimated probabilities but only defect-free substrate images. Thus, the images for which we estimate higher probabilities of defects are never defected, resulting in relative frequencies of zero. Figure 9b, corresponding to the defect test set, shows near-perfect calibration, i.e., the points lie exactly along the diagonal. This results from our perfect classification performance on this test set, as well as the high confidence on whether images were defect-free or defected; recall from Figure 8b that our collection of estimated probabilities for this test set were massed near zero and one.

5. Conclusions

We developed a machine learning method for detecting anomalies in the molecular structure of Gallium Arsenide. Additionally, this method can be successfully trained without access to a training set containing labeled defects. This allows us to create a model to find defects in a substance without any prior knowledge as to what a defect actually looks like. This method could also be effective in detecting defects in the atomic structure of other materials, with appropriate alterations to accommodate new data structures.

We note that our results are based on the assumption that each 118-by-84-pixel sample from the defect image contains at least one defect. Based on the conditions in which the samples are created, we are confident in this assumption, yet we currently do not have a way to verify this. With the understanding that a test set with 100%-accurate labels may never be possible, work on developing alternative ways to validate the results is underway and will be reported in a follow-on paper.

Although this work mainly constitutes the proof of concept, it demonstrates that it is possible to detect *if* a defect is present in a section of Gallium Arsenide, thus filling a capability gap. The natural next step is to actually locate where the defects are: the *where* problem. This will surely be more challenging in the absence of accurate test data and the lack of educated assumptions.

Author Contributions: Conceptualization, A.W. and P.C.; Methodology, T.R. and P.C.; Software, T.R. and P.C.; Validation, P.C. and C.J.; Resources, A.W.; Writing—original draft, T.R. and C.J.; Writing—review & editing, T.R., A.W., P.C. and C.J.; Visualization, P.C. and C.J.; Supervision, A.W.; Project administration, A.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Moss, S.J.; Ledwith, A. *Chemistry of the Semiconductor Industry*; Springer Science & Business Media: Cham, Switzerland, 1989.
2. McCluskey, M.D.; Haller, E.E. *Dopants and Defects in Semiconductors*; CRC Press: Boca Raton, FL, USA, 2018.
3. Cho, P.; Wood, A.; Mahalingam, K.; Eyink, K. Defect detection in atomic resolution transmission electron microscopy images using machine learning. *Mathematics* **2021**, *9*, 1209. [CrossRef]
4. Zhang, M.; Wu, J.; Lin, H.; Yuan, P.; Song, Y. The application of one-class classifier based on CNN in image defect detection. *Procedia Comput. Sci.* **2017**, *114*, 341–348. [CrossRef]
5. Chen, X.; Chen, J.; Han, X.; Zhao, C.; Zhang, D.; Zhu, K.; Su, Y. A light-weighted CNN model for wafer structural defect detection. *IEEE Access* **2020**, *8*, 24006–24018. [CrossRef]
6. Shi, J.; Li, Z.; Zhu, T.; Wang, D.; Ni, C. Defect detection of industry wood veneer based on NAS and multi-channel mask R-CNN. *Sensors* **2020**, *20*, 4398. [CrossRef] [PubMed]
7. Roweis, S. EM algorithms for PCA and SPCA. *Adv. Neural Inf. Process. Syst.* **1997**, *10*, 626–632.
8. Malagon-Borja, L.; Fuentes, O. Object detection using image reconstruction with PCA. *Image Vis. Comput.* **2009**, *27*, 2–9. [CrossRef]
9. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Van Esesn, B.C.; Awwal, A.A.S.; Asari, V.K. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv* **2018**, arXiv:1803.01164.
10. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1321–1330.