Unsupervised Electroluminescence Image Clustering for Automated Defect Detection in Photovoltaic Cells

Benjamin G. Pierce[1,2], Rounak Chawla[1], and Ellis Wright[1]
[1]Case Western Reserve University, Cleveland, OH, 44106
[2]Sandia National Laboratories, Albuquerque, NM, 87185

***Abstract*—Photovoltaics (PV) have become an increasingly inexpensive source of renewable energy in recent years, largely due to maturing technology and economies of scale. As the total amount of solar PV installed exceeds 1 terawatt (as of 2022), quantification of PV performance becomes crucial to manufacturers, insurers, and system operators. A commonly used form of nondestructive testing of PVs is electroluminescence (EL) imaging, where a bias is applied to a PV cell or module and the usable area emits light in the near-infrared spectrum, which is captured by a sensitive camera. These EL images are commonly used to manually and qualitatively evaluate modules by the failures or flaws they exhibit, commonly cracking, corrosion, or various manufacturing defects. However, it is difficult to do this at scale, especially in an assembly line fashion. Previous works have explored the possibilities of using convolutional neural networks (CNNs) to automatically classify EL images, but these CNNs require labeled training data in large quantities, and are vulnerable to the imbalanced class problem. In this work, we introduce a unsupervised convolutional auto-encoder (CAE) model, and compare it to a classical unsupervised image processing approach. We show that unsupervised approaches can attain very good accuracy scores on an EL image dataset without the need for costly training data with labels.**

***Index Terms*—electroluminescence imaging, defect detection, photovoltaic characterization, image processing, convolutional autoencoder**

I. Introduction

As the world moves towards renewable and reduced emissions energy sources, photovoltaics (PV) have become a economically dominant technology, as the price on PV modules continues to fall. The variable cost of fuel (sunlight) for PV is zero, but the large upfront capital expenditure of PV makes it an amortized asset, meaning that in order to be financially viable, a PV system must have a long expected lifetime. Defects in PV modules, introduced in manufacturing, shipping, or in the field, can significantly reduce system lifetime. Therefore, there is a enormous economic interest in detecting and measuring possible defects in a PV module before, during, and after its lifetime in the field.

One measurement that is used for this purpose is electroluminescence (EL), which generates an image of the PV module that quantifies how well it can produce electricity. The physics behind an EL image is relatively straightforward. When operating, a solar panel generates electricity through the photovoltaic effect, where light provides the necessary energy for electrons in the valence band of the module material (boron or phosphorus doped silicon) to excite beyond their valence band and become free electrons, which can move across a p-n junction to induce a current.

If this is run in reverse and a current is *applied* to the PV panel, it will then *emit* light in the near-infrared spectrum, but only in areas where it would have produced light in normal operation. This makes an EL image a valuable measurement of the module, as dark areas in the image correspond to where the module does not produce power. This procedure is commonly used as part of an assembly line quality assurance step, or as a field measurement to evaluate module lifetime. In either case, evaluation of when a EL image looks different from the ordinary is a useful step to automate, and if needed, evaluate manually.

However, manual inspection of many EL images is quite costly, and as the PV industry continues to scale up in near short term, the industry needs methods to perform automated review of EL images. The literature over the last few years has been focused on using Convolutional Neural Networks (CNNs) to classify EL images based on labeled examples, as in [1]. This methodology essentially automates the processing of selecting filters in the spacial domain by learning convolutional kernels from the data. However, CNNs are very dependent on data quantity and quality, and in particular are vulnerable to the imbalanced class problem, where a dataset does not contain an adequate balance of the intended targets. This results in a model learning a non-useful target distribution. For example, a model trained on a dataset with 999 EL images that represent a ”Good” module, and 1 that represents a ”Bad” module, the model will be correct 99.9% of the time by simply always classifying any input as ”Good”. While representative of the training set, this output is not desirable. Unfortunately, this scenario is very common in EL image data, as manufacturing and operations and maintenance procedures are designed for a low failure rate in order to be profitable, and companies are generally reluctant to perform destructive testing on a large scale, which would be needed to construct a complete, class-balanced dataset for supervised learning. Additionally, failures in PV modules can be quite diverse, and difficult to classify into well-defined classes due to different module types, manufacturing processes, and varied failure triggers in the field. Even expert labeling of EL images is often subjective and subject to various errors. These issues make supervised models like CNNs less desirable for some applications.

Consider Deitsch *et. al.* [2], who developed an EL image dataset of 2624 solar cells, with annotated defect regions. This dataset was generated via manual annotation of the ground truth defects in the EL images with the help of an expert. While this led to the creation of a useful open dataset, this procedure is inefficient, non-reproducible, and inaccessible to researchers without an expert. As such, many unannotated EL image datasets of solar cells exist, and our project aims to examine the the representative accuracy of learned defect features from these datasets in an unsupervised setting.

Unsupervised learning, in contrast, does not require labeled training data, and seeks to estimate the natural clusters in the data, which hopefully correspond to real-world classifications. As they are primarily data-driven, unsupervised models can avoid human bias in the labels, and identify failure types that may not be understood *a priori*. Additionally, clustering methods can be more robust to imbalanced classes, and are often more explainable when a misclassification occurs. However, the main strength of unsupervised methods in this case is in exploratory data analysis and human-in-the-loop feedback. Oftentimes, if a model can be used to raise a flag that a certain module has some defect, it is usually inspected manually. The value of automated efforts such as this is to narrow the amount of samples that require human intervention, not to replace the human element entirely, while avoiding injection of bias in a labeling process and the associated increased costs of labeling. Therefore, in this work, we present two types of unsupervised feature extractors, one that uses classical image processing techniques and one with a deep autoencoder model. By comparing these techniques to each other, we can identify which techniques perform the best across various datasets and provide actionable insight and model guidelines/code for use in industry. In summary, this work presents the following notable contributions:

1. A classical feature extraction pipeline that does not ultilize machine learning
2. A Convolutional Autoencoder feature extraction pipeline using state-of-the-art ML techniques
3. Feature vector clustering methods and comparison between various approaches
4. Performance quantification of methods

II. Dataset

There are multiple open EL image datasets available. The first is from CWRU, and hosted on OSF [3]. Another dataset [4] from Lawrence Berkeley National Lab is hosted as part of the Duramat Consortium (which BGP is affiliated with), primarily focusing on crack detection. Deitsch *et. al.*'s open EL dataset [2] predates the former datasets, and is another benchmark in the community.

III. Methodology

The overarching method has the following steps:

1. Compute a feature vector $v \in \mathbb{R}^k$ from an image $I \in \mathbb{R}^{m \times n}$, where k is a hyper-parameter that is empirically determined.
2. Assemble feature vectors for all N images, forming a matrix $X \in \mathbb{R}^{N \times k}$.
3. Cluster on the samples, and group each image into a cluster $1 \ldots g$, where g is the desired number of groups.
4. Map each g_i to a corresponding real-world, physical classification.

Step 1 will be performed with A) Classical Image Processing, and B) Deep Learning, which are explained below.

A. Classical Image Processing

The classical image pipeline will follow that of our previous work in [5] where an image processing & feature extraction pipeline is constructed to perform clustering on representative vectors of EL images. These vectors are extracted with a customized feature detector, which will be improved upon in this paper. A block diagram of the pipeline is shown in Figure 2.

The first step of the classical image processing pipeline is feature extraction, consisting of three different methods. One method is that of local feature extraction, where "patches" of the image are identified and vectorized by intensity. Note that this process is very local, and only operates on patches in order to retain scale and

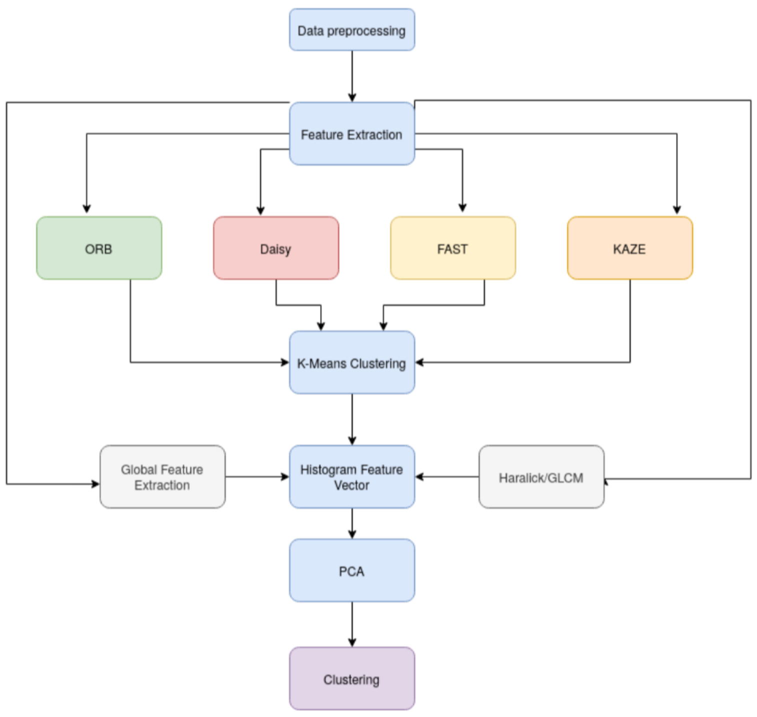


Fig. 1: The unsupervised classical image processing framework shown in [5].

rotation invariance. Methods like ORB [6], Daisy [7], FAST [8], and KAZE [9] can all be used interchangeably. These methods present generalized approaches; two further methods are used as more application-specific additions to these patch feature extractors.

Another set of features are known as Haralick or Grey Level Co-occurance Matrix (GLCM) texture features [10]. These features are more specific to this problem space, as they are restricted to greyscale images. Haralick features work by computing dissimilarity and correlation between images patches. This process is more direct and a better judge of image texture then the generalized patch feature extractors.

The final set of features utilized is very specific to this problem. Note that the previous two methods operated only on image patches; this can be an issue for this domain, since many features of interest (cracks, corrosion) in EL images stretch over much of the image. The final method of "global feature extraction" is quite simple: take a column-wise sum over the entire image, and normalize this vector between zero and one. This concept was first presented in [11] as "normalized busbar width", where it was noted to correlate with a drop in power due to corrosion. This work adapts the process to form a feature vector that can easily delineate the corrosion failure mode.

Once the three feature vectors (one from the patch feature extractor, one from Haralick/GLCM features, and one from the busbar corrosion global detector), they are directly appended together to form a very long vector for each image. This vector is generally quite unwieldy and high-dimensional; Principal Component Analysis (PCA) is then employed to shorten it. Since the three methods all attempt to quantify the same thing, there is signifigant co-linearity within the merged feature vector. This allows PCA to reduce the dimensionality by roughly an order

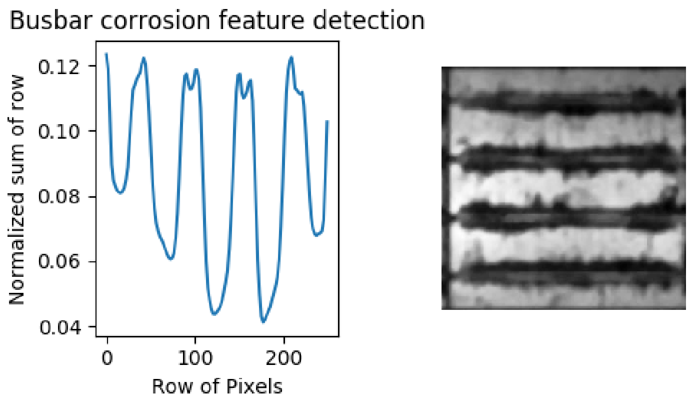


Fig. 2: A vector captured by a column-wise sum over the image and then normalized. This roughly corresponds to the width of the busbars.

of magnitude, although the exact length of the feature vector is dependent on the data and empirical judgement. The reduced feature vector is then used as input the the clustering metric.

B. Deep Learning

In this work, we will use a Convolutional Autoencoder (CAE) as the deep learning technique. A CAE learns many convolution kernels to automate the process of feature extraction by down-sampling a image to a vector, thereby creating a representational "bottleneck", and then up-sampling the image back up with transposed convolutions. The loss between the original image and reconstructed image is computed, and then backpropagated through the network to learn convolutional kernels. These convolutional kernel parameters can be similar to kernels constructed by a classical image processing methodology, but often become difficult to interpret in deeper networks.

Once the model has been trained to the point where it can successfully reproduce an image, the vector "bottleneck," referred to as the latent space, is used as a vector representation of the image. This process can be viewed as a non-linear and high-dimensional version of Principal Component Analysis (PCA), which is commonly applied on vector data for dimensionality reduction by exploiting co-linearites in the data [12]. Note that the size of the latent space is a significant hyper-parameter for CAE/VAE models, and is a somewhat arbitrary and empirical choice to be optimized with a hyper-parameter tuning framework.

The architecture of our model is loosely based on the U-Net design from [13], and also draws concepts from the Variational Autoencoder from [14], both of which feature many convolutional layers with a latent space representation of the model. Typically, this latent space is a relatively small vector; by forcing information to flow through the latent space to reconstruct the original image, the model learns a low-dimensional encoding of the image data, which is what we were looking for. Note that reconstructive step is less important for this application, and is only used as a model training tactic.

We choose to employ the variational trick such that the decoding half of the network also learns a denoising process, making our model into a "Variational Autoencoder" (VAE). This enables to model to function in a *generative* fashion; if we provide the model with noise at the latent space, it will decode that noise into a plausible image. This work does not make use of this feature, but the variational method is still employed because it has been empirically found to be more stable in an EL image context. A possible explanation for this is that the optimization of a VAE relies on the evidence lower bound (ELBO) optimization strategy for the log-likelihood of the data. The mechanics of this strategy result in the model slightly overfitting the data, which may be desirable behaviour in this case [15]. The output of the model also features the characteristic blurring of the VAE as seen in Figure 5; this property does not make a difference in this application as we do not utilize the reconstruction process for generation. For realistic generation, adversarial methods [16] may be employed at the cost of training stability; we declined to use them in this work due to the small dataset size. Our full architecture is shown in detail in Figure 3; although following a somewhat standard structure, the model is original to this work and therefore does not utilize pre-trained weights or other forms of transfer learning.

In any case, the resultant output of the model is a feature vector extracted from the encoder half of the autoencoder of the desired length. Therefore, both the classical image processing and machine learning pipelines output a matrix of feature vectors, where each vector corresponds to a single image. This matrix is then passed to a clustering method.

C. Clustering

In this work, we deploy various clustering algorithms. The first the Gaussian Mixture model. This algorithm assumes that the data is drawn for a mixture of Gaussian distributions, and seeks to estimate the probability that each point belongs to a certain cluster via expectation maximization [17]. This approach provides a more flexible approach then simpler methods like k-means[1], but does assume that the data comprises of a mixture of Gaussians. This may not always be the case, especially in a high dimensional space.

Another clustering method of interest is hierarchical clustering, which is also known as "agglomerative" or "divisive" clustering, depending the implemented approach [18]. The agglomerative approach begins by assigning samples to their own cluster, and then merges clusters based on a distance metric. In contrast, the divisive approach places all samples in one cluster, and then splits that cluster based on a distance metric. Both cases utilize some form of distance metric, which can be any valid metric over a defined metric space. This is an advantage of hierarchical clustering; in higher dimensional spaces, Euclidean distance is quite poor and it is usually better to use something like L_1 (Manhattan/taxicab) distance [19], which we utilize in this work.

D. Comparative Analysis

In this step, the different models are compared with each other. It is expected that each model will have multiple variations that have small, but significant differences, such as different deep learning hyperparameters, different clustering methods, and different filter coefficients.

Note that comparing unsupervised learning methods is quite difficult, in that human labeled samples may not exactly map to clusters a unsupervised algorithm determines. However, this does not strictly imply that the algorithm itself is incorrect; it

[1] k-means is in fact a special case of GMMs where variance is assumed to be spherical

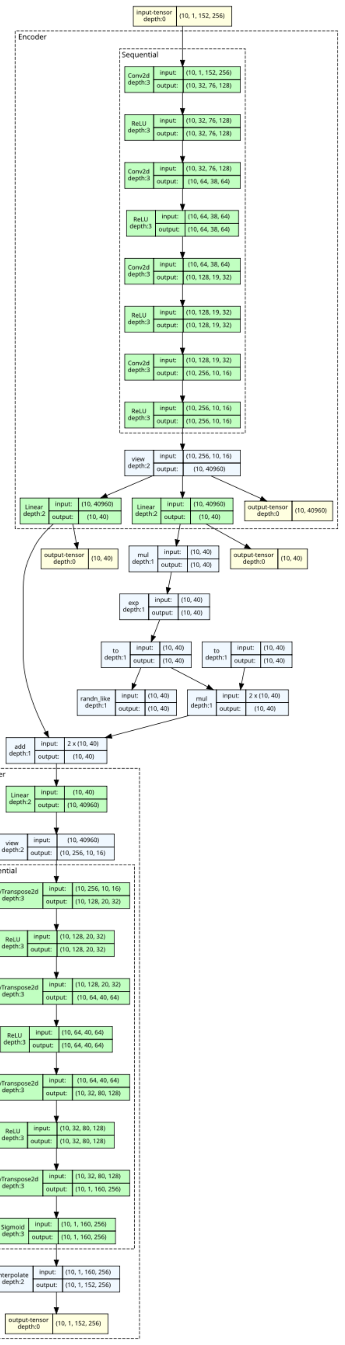

input-tensor depth:0 (10, 1, 152, 256)
Encoder
Sequential
Conv2d depth:3 input: (10, 1, 152, 256) output: (10, 32, 76, 128)
ReLU depth:3 input: (10, 32, 76, 128) output: (10, 32, 76, 128)
Conv2d depth:3 input: (10, 32, 76, 128) output: (10, 64, 38, 64)
ReLU depth:3 input: (10, 64, 38, 64) output: (10, 64, 38, 64)
Conv2d depth:3 input: (10, 64, 38, 64) output: (10, 128, 19, 32)
ReLU depth:3 input: (10, 128, 19, 32) output: (10, 128, 19, 32)
Conv2d depth:3 input: (10, 128, 19, 32) output: (10, 256, 10, 16)
ReLU depth:3 input: (10, 256, 10, 16) output: (10, 256, 10, 16)
view depth:2 input: (10, 256, 10, 16) output: (10, 40960)
Linear depth:2 input: (10, 40960) output: (10, 40)
Linear depth:2 input: (10, 40960) output: (10, 40)
output-tensor depth:0 (10, 40960)
output-tensor depth:0 (10, 40)
mul depth:1 input: (10, 40) output: (10, 40)
output-tensor depth:0 (10, 40)
exp depth:1 input: (10, 40) output: (10, 40)
to depth:1 input: (10, 40) output: (10, 40)
to depth:1 input: (10, 40) output: (10, 40)
randn_like depth:1 input: (10, 40) output: (10, 40)
mul depth:1 input: 2 x (10, 40) output: (10, 40)
add depth:1 input: 2 x (10, 40) output: (10, 40)
Decoder
Linear depth:2 input: (10, 40) output: (10, 40960)
view depth:2 input: (10, 40960) output: (10, 256, 10, 16)
Sequential
ConvTranspose2d depth:3 input: (10, 256, 10, 16) output: (10, 128, 20, 32)
ReLU depth:3 input: (10, 128, 20, 32) output: (10, 128, 20, 32)
ConvTranspose2d depth:3 input: (10, 128, 20, 32) output: (10, 64, 40, 64)
ReLU depth:3 input: (10, 64, 40, 64) output: (10, 64, 40, 64)
ConvTranspose2d depth:3 input: (10, 64, 40, 64) output: (10, 32, 80, 128)
ReLU depth:3 input: (10, 32, 80, 128) output: (10, 32, 80, 128)
ConvTranspose2d depth:3 input: (10, 32, 80, 128) output: (10, 1, 160, 256)
Sigmoid depth:3 input: (10, 1, 160, 256) output: (10, 1, 160, 256)
interpolate depth:2 input: (10, 1, 160, 256) output: (10, 1, 152, 256)
output-tensor depth:0 (10, 1, 152, 256)

Fig. 3: The full model architecture of our Convolutional Autoencoder, with image shapes shown for a arbitrary input.

is quite common for unsupervised methods to pick up on intra-dataset variance that was not identified in *a priori* labeling. This represents both an advantage and disadvantage of unsupervised ML. For example, in an EL image context, one issue that has come up has been of different cell types. PV cells can be monocrystalline, and appear very smooth in an image, or polycrystalline, and have many small dark spots. A model may recognize these differences and sort poly and mono crystalline samples into different clusters, even if both are undamaged. This may be undesirable behaviour if we do not care about cell type, but may be useful if we do.

IV. Tools and Technologies

This work will utilizes the Python programming language. For the deep learning portion, the PyTorch framework is used. The classical image processing pipeline is implemented using the scikit-learn package. The individual feature extraction/segmentation algorithms are implemented using the OpenCV framework. Model training and hyperparameter tuning is carried out using the CWRU HPC system.

V. Results

A. Classical Image Processing

A total of 272 configurations were tested, incorporating different methods and parameters:

- Feature Extraction Methods: ORB, Daisy, FAST, and KAZE. These methods were chosen for their robustness and varying characteristics in capturing key points and descriptors.
- Global Feature Inclusion: Both scenarios, with and without the inclusion of global features, were tested to determine their impact on the model's performance.
- Haralick Texture Features: We considered the extraction of Haralick features to evaluate their effectiveness in improving clustering outcomes.
- Dimensionality Reduction with PCA: Models were evaluated with and without PCA, and various dimensionalities (2, 5, 8, 12) to determine the optimal number of principal components.
- Clustering Methods: Gaussian Mixture Models (GMM), Agglomerative Clustering (AGG), and Spectral Clustering were employed to segment the features into two groups—defective and non-defective.

For pre-processing, each image underwent a Gaussian blur with a 5x5 kernel to reduce noise and enhance feature detection. In the bag of words construction, we utilized a histogram length of 100. If global features were incorporated, they had a vector length of 125. The clustering was performed into two categories, aiming to classify the images as either defective or non-defective.

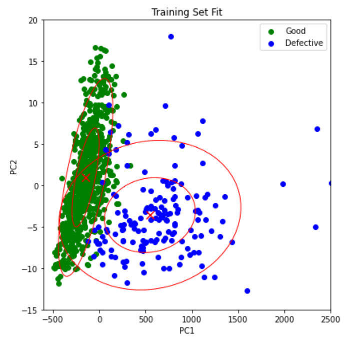
The clustering results were evaluated based on accuracy and f1-score, with 22 models achieving an accuracy rate and f1 of 98%. These models utilized either spectral or agglomerative clustering. The spectral and agglomerative clustering methods were applied directly to the transformed testing set, without utilizing any information or parameters derived from the training data. The best-performing model utilizing GMM achieved a 97% accuracy. This model was particularly significant as it demonstrated the ability to discern meaningful patterns in the data, utilizing just two dimensions after PCA reduction.

Figure 3 below illustrates the PCA-reduced feature space for the model employing FAST feature extraction, supplemented with global and Haralick features, and further processed through a GMM. This visualization not only confirms the model's efficacy in retaining relevant features in a reduced dimensionality but also highlights the clear segmentation of the data clusters, substantiating the model's 97% accuracy in and 96% f1-score in classification.

This framework has established a robust baseline for identifying effective combinations of feature extraction and clustering techniques with classical methods.

B. Autoencoder

An autoencoder is trained in a self-supervised fashion; the output is expected to be the same as the input. We then calculate the



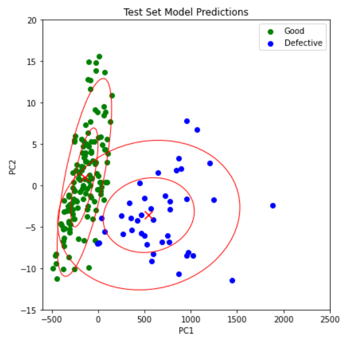


Fig. 4: The left plot shows the transformed training set with the GMM fit to it. The class names were not given, as it was unsupervised but the true labels were displayed for visualization purposes. The right plot shows the transformed test set and the prediction made by the clustering model.

error in this reconstruction, and backpropagate this error through the network to update individual weights. An example reconstruction is shown in Figure 5.

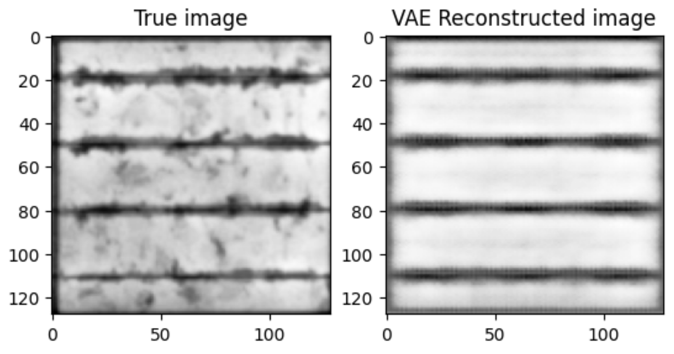


Fig. 5: A reconstructed cell that is experiencing significant busbar corrosion

Note that this reconstruction does *not* consider cell type, as the true image is polycrystalline Si and the reconstruction appears to be monocrystalline Si. This is a notable finding that also appears in the classical image processing methodology. In this use case, the behaviour is to be encouraged as we are trying to cluster failures, not cell type. Further note that the edges of the reconstructed image are not sharp; this is a consequence of choosing a Variational Autoencoder, which seeks to minimize expected failures by learning fuzzy borders. This flaw could be reduced by integrating an adversarial component at the cost of training instability. The training curve for this model is plotted in Figure 6. Note that early stopping was applied on this model (and so the relative magnitude of the loss is still large) to avoid overfitting on very small datasets.

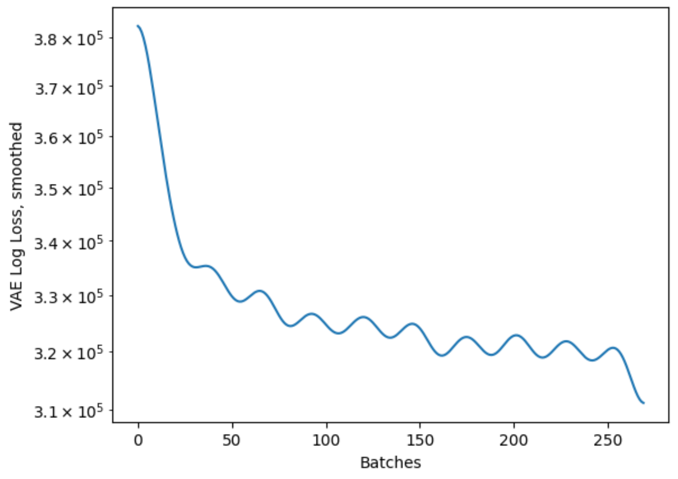


Fig. 6: Model loss curve, where the loss term is a sum of a reconstructive binary crossentropy and a KL divergence between the true and learned noise as a part of the variational method. Convergence to large loss values indicate a lack of data in the training set, a typical issue in small-scale datasets like this. Note that an exact reconstruction is not needed for this work, so blurred or idealized representations of the image are acceptable, provided that the encoded feature vector remain useful.

A major hyper-parameter for this type of model is the length of the encoded feature vector. Generally, the selection process for this parameter is usually similar to that of methods like principal component analysis, as performance at decreasing parameter values is recording in hope of finding a good value. In this case, we want to select a value that preserves most of the relevant information, while the feature vector is small enough to analyze. Alternatively, a feature vector of length 2 or 3 can be used for visualization, as in Figure 7. Figure 7 shows good seperation between “Good” (yellow) and “Corroded” (purple) cells; the “Crack” (teal) cell is mixed in with the ”Good” cluster because it shows a very small difference from them.

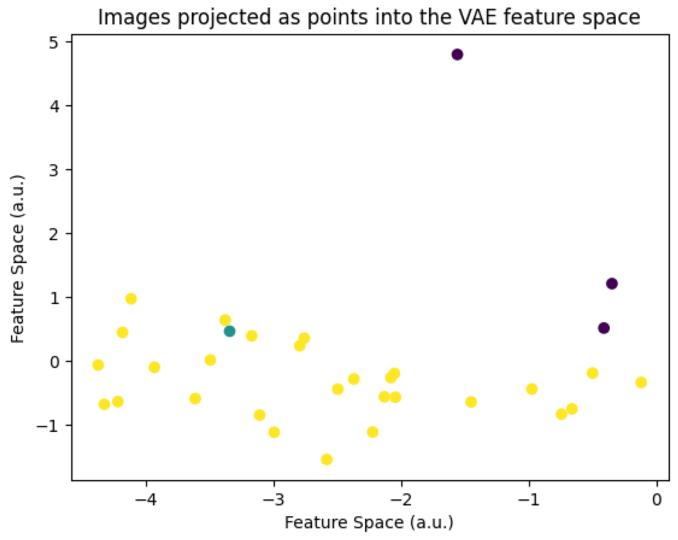


Fig. 7: Images projected into a learned feature space, where different colors correspond to different failure modes

C. Clustering

We choose to evaluate the model in a two-class scenario, representing "Good" or "Defective", as shown in Figure 8. This is the most common usage of these models in the field. Note that our data has been labeled into three classes for supervised learning, but we are only specifying that the clustering method give two clusters. The hope is that the feature embedding are robust and descriptive enough to provide the correct separation between the classes. On this task, we find that the **classical image processing pipeline has an accuracy of 97%** and the **CAE/Deep Learning pipeline has an accuracy of 98%**. These values are quite comparable, which shows how both of these models provide robust pipelines for unsupervised ML.

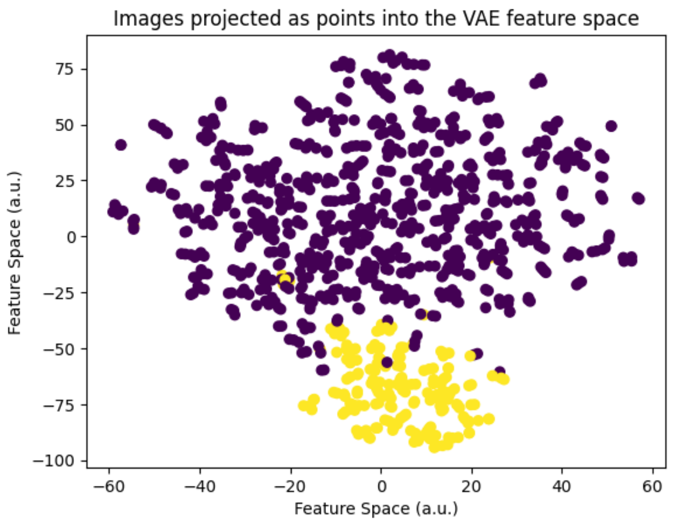


Fig. 8: Clustering result over an entire datset with two clusters

VI. Future Work

In the future, models can be evaluated on larger datasets, which were not included in this study due to copyright concerns[2] or lack of labels. The current dataset is quite trivial for these methods, so a more useful comparison may be accomplished with a larger and more difficult dataset. Additionally, we would like to better evaluate various clustering methods with various metrics, instead of just classification accuracy.

Another topic for future work is direct comparison to supervised methods, such as CNNs. As noted above, these comparisons are often not one-to-one, as supervised and unsupervised learning are very different paradigms. However, simple comparisons like those conducted in this study would be appropriate, and a good topic for full comparative analysis.

VII. Conclusion

In this work, we present two pipelines for solar cell electroluminescence (EL) image processing, both of which perform feature extraction, dimensionality reduction, and automated, unsupervised clustering. The first pipeline makes use of classical image processing techniques and feature detectors, whereas the second presents a deep learning approach using convolutional/variational autoencoders. Model performance is evaluated over an open source dataset, and it is found that both approaches are comparable in accuracy from 97% to 98%, which demonstrates the power of unsupervised learning techniques as a class of algorithum. Further work includes expansion on more difficult datasets and comparison to supervised convolutional neural network models.

[2]Companies generally don't like to release datasets where their product is shown to be defective to the public, so much of the data we have is behind an NDA.

VIII. Group Contributions

1. Benjamin Pierce: Deep Learning/CAE pipeline, project conceptualization, writing, slides
2. Ellis Wright: Classical Image Processing pipeline, live demo, slides
3. Rounak Chawla: Paper edits

IX. Acknowledgements

BGP would like to acknowledge Dr. Jennifer Braid, Dr. Norman Jost, and Emma Cooper at Sandia National Laboratories for continued discussions and code review on electroluminescence image processing via funding from the DuraMAT consortium, as well as Dr. Ahmad Karimi at Oak Ridge National Laboratory for original dataset curation and publication to OSF.

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1937968.

X. Appendix

Code written and data utilized for this project can be found on our GitHub page: https://github.com/bgpierc/cwru-csds490-project

The primary dataset utilized in this project can be found via the OSF at: https://osf.io/4qrtv/

References

[1] M. W. Akram, G. Li, Y. Jin, X. Chen, C. Zhu, X. Zhao, A. Khaliq, M. Faheem, and A. Ahmad, "CNN based automatic detection of photovoltaic cell defects in electroluminescence images," vol. 189, p. 116319. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544219320146

[2] S. Deitsch, V. Christlein, S. Berger, C. Buerhop-Lutz, A. Maier, F. Gallwitz, and C. Riess, "Automatic classification of defective photovoltaic module cells in electroluminescence images," *Solar Energy*, vol. 185, pp. 455–468, Jun. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X19302014

[3] "OSF | Electroluminescent (EL) Image Dataset of PV Module Under Step-wise Damp Heat Exposures." [Online]. Available: https://osf.io/4qrtv/

[4] X. Chen, T. Karin, and A. Jain, "Automated defect identification in electroluminescence images of solar modules," *Solar Energy*, vol. 242, pp. 20–29, Aug. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X22004571

[5] B. G. Pierce, A. M. Karimi, J. Liu, R. H. French, and J. L. Braid, "Identifying Degradation Modes of Photovoltaic Modules Using Unsupervised Machine Learning on Electroluminesense Images," in *2020 47th IEEE Photovoltaic Specialists Conference (PVSC)*. IEEE, 2020, pp. 1850–1855. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9301021/

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2564–2571, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/document/6126544

[7] E. Tola, V. Lepetit, and P. Fua, "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, May 2010. [Online]. Available: http://ieeexplore.ieee.org/document/4815264/

[8] E. Rosten, R. Porter, and T. Drummond, "Faster and better: a machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–119, Jan. 2010, arXiv:0810.2434 [cs]. [Online]. Available: http://arxiv.org/abs/0810.2434

[9] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “KAZE Features,” in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 214–227.

[10] R. M. Haralick, K. Shanmugam, and I. Dinstein, “Textural Features for Image Classification,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973, conference Name: IEEE Transactions on Systems, Man, and Cybernetics. [Online]. Available: https://ieeexplore.ieee.org/document/4309314

[11] A. M. Karimi, J. S. Fada, N. A. Parrilla, B. G. Pierce, M. Koyutürk, R. H. French, and J. L. Braid, “Generalized and mechanistic PV module performance prediction from computer vision and machine learning on electroluminescence images,” *IEEE Journal of Photovoltaics*, vol. 10, no. 3, pp. 878–887, 2020, publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9050914/

[12] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” *AIChE Journal*, vol. 37, pp. 233–243, Feb. 1991, aDS Bibcode: 1991AIChE..37..233K. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1991AIChE..37..233K

[13] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[14] A. Subramanian, “Pytorch-vae,” https://github.com/AntixK/PyTorch-VAE, 2020.

[15] A. Shekhovtsov, D. Schlesinger, and B. Flach, “Vae approximation error: Elbo and exponential families,” 2022.

[16] A. Plumerault, H. L. Borgne, and C. Hudelot, “AVAE: Adversarial Variational Auto Encoder,” Dec. 2020, arXiv:2012.11551 [cs]. [Online]. Available: http://arxiv.org/abs/2012.11551

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977, publisher: [Royal Statistical Society, Wiley]. [Online]. Available: https://www.jstor.org/stable/2984875

[18] G. J. Szekely and M. L. Rizzo, “Hierarchical Clustering via Joint Between-Within Distances: Extending Ward’s Minimum Variance Method,” *Journal of Classification*, vol. 22, no. 2, pp. 151–183, Sep. 2005. [Online]. Available: https://doi.org/10.1007/s00357-005-0012-9

[19] P. Domingos, “A few useful things to know about machine learning,” *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2347736.2347755