Date: April 24, 2023

To: Dr. Alan Barhorst, PE, Professor and Head

Department of Mechanical Engineering

From: Gabriel Dimonde

Cc: Dr. Domec, Instructor

Department of Mechanical Engineering

Subject: ENGR 400 - Final Project - An Exploration of Autoencoders and one of their uses in Convolutional, and Fully Connected Networks.

1 Overview

The aim of this project in general, from the class syllabus, was to build, train, validate and tune at least 2 models on an engineering dataset. The goal of this effort was to do the above on a Convolutional network and a Fully Connected network as models. The dataset chosen was the Ultra High Carbon Steel (UHCS) dataset, available from:

http://hdl.handle.net/11256/940

The idea was to train Convolutional and Fully Connected networks to be able to classify the micrograph image according to the type of carbon steel microstructure present in the image. Then compare the respective models on their performances.

Since the ENGR400 class was not going to be able to learn about the usefulness of Autoencoders (due to too many breaks), and because there was an opportunity within the project to do so, Autoencoders were also used. They were used for pretraining layer weights for the aforementioned models on unlabeled micrograph images as roughly half of the dataset contained unlabeled micrographs.

2 Importing the dataset

The data used for this project was retrieved from kaggle.com which was organized by Safiuddin et al. The data contained 962 image files (.PNG) and an excel sheet containing the metadata for the images. For example, it contained the filename, percentage of carbon, the anneal temperature, anneal time, cool method, magnification, and primary microconstituent. The image files were images of size 645 by 481 pixels. It is these images that are being used as the X vector and the the "primary microconstituent" column in the excel sheet is the Y vector for model training. The features that are not being used are the percentage of carbon, the anneal temperature, anneal time, cool method, and magnification. The reason that these are not being used is due to the thought that it would add a confound to the question: what gives the model the best accuracy? If the classifier models are only trained using images then it is known that the layers above are adequately set-up if the classifier performs well.

Of the labeled images, the class frequency was very stratified. See Table 1.

Table 1: Below is the class frequency in count and in percentage of the entire labeled set.

primary microconstituent	count	percentage
spheroidite	372	62.2%
network	101	16.9%
spheroidite+widmanstatten	77	12.9%
pearlite+spheroidite	28	4.68%
pearlite	15	2.50%
pearlite+widmanstatten	5	0.836%

After using ChatGPT to write code to distribute the images according to their respective class into a directory and sub-directories, Table 2 shows the class frequency of the labeled test set.

Table 2: Below is the class frequency in count and in percentage of the test set.

primary microconstituent	count	percentage
spheroidite	75	61.5%
network	21	17.2%
spheroidite+widmanstatten	16	13.1%
pearlite+spheroidite	6	4.92%
pearlite	3	2.45%
pearlite+widmanstatten	1	0.819%

3 Preprocessing and Data Augmentation

The images were first blanket renamed from Croppedmicrograph#.png to micrograph#.png for the purposes of making the file names synonymous with the names listed in the "path" column of the excel metadata file. Next the images were separated with respect to whether their names existed in the "path" column of the excel file.

The labeled images were placed into separate directories according to their respective class. First they were all placed into the Labeled_Dir directory as seen in Figure 1.

After this, the labeled images were then separated again with a 20% split into the Labeled_Test_Dir that exactly resembled the Labeled_Dir. The code for the above preprocessing steps was done by ChatGPT.

The dataset, as mentioned above, included 961 images but only 598 images are labeled. The unlabeled images were used to train Fully Connected and Convolutional Autoencoders for pretraining layers to be transferred into a classifier model.

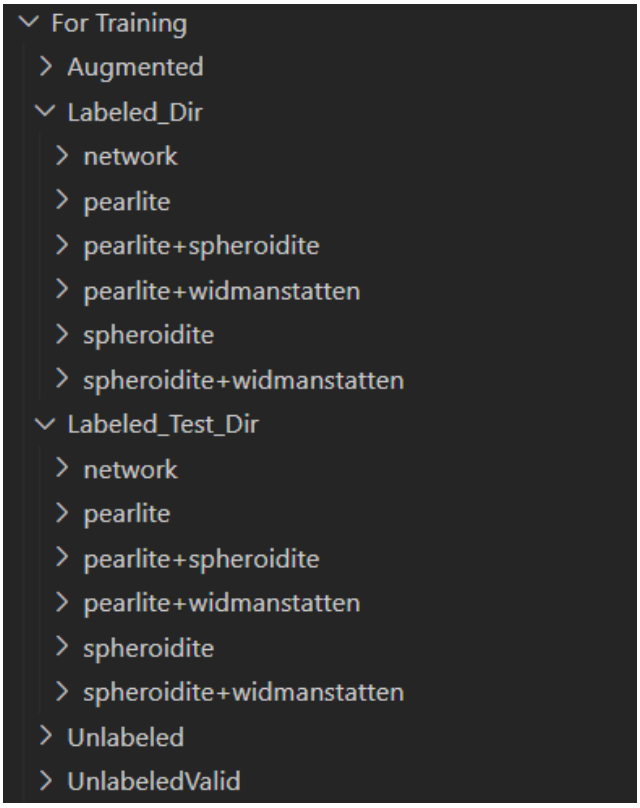

Figure 1: The file structure containing the micrograph images for the .flow_from_directory() method.

The unlabeled images were not augmented images, because of two reasons: Reason one includes the inability to feed a datagen.flow() object to a .fit() method as both an x and a y vector. This was thought to apply to all .flow() method types. Reason two, once preprocessing layers are implemented into the Autoencoders it is unknown whether the error will be calculated between correspondingly augmented images or not. It was for these above reasons that the unlabeled images were preloaded into memory as arrays with size (?,256,256,1), where the ? denotes batch size. There were 295 unlabeled images for training, and 68 for validation.

The parameters specified for data augmentation are found in Table 3. Notable parameters are the fill mode and the validation split, where the fill mode is set to "reflect" in order to maintain some microstructure shape just in case the edge of the image gets overlapped. The models trained here are not capable of recognising a mirror line. The validation split for the labeled training images was set here to be 20%.

Table 3: Data augmentation parameters as passed to the ImageDataGenerator() iterator.

rotation range	5
width shift range	0.1
height shift range	0.1
height shift range	0.1
brightness range	[0.1,2]
channel shift range	0.5
shear range	0.2
zoom range	0.2
horizontal flip	True
vertical flip	True
rescale	1./255
fill mode	reflect
validation split	0.2

Example augmented images are shown in Appendix 6.1.

4 Models

The structure of the Fully Connected Autoencoder followed the typical ideas behind undercompleteness, where information is funneled through more and more restrictive pathways until only the patterns that the information contains remain, see Table 4 for the specific structure of the Fully Connected network.

Table 4: Architecture of the Fully Connected Autoencoder.

Layer	**Output Shape**	**Param #**
Flatten	(None, 65536)	0
Dense	(None, 2000)	131074000
Dense_1	(None, 1000)	2001000
Dense_2	(None, 750)	750750
Dense_3	(None, 1000)	751000
Dense_4	(None, 2000)	2002000
Dense_5	(None, 65536)	1311337536
Reshape	(None, 256, 256, 1)	0
Total Params:	267,716,286	
Trainable Params:	267,716,286	
Non-trainable Params:	0	

The structure of the Convolutional network is outlined in the table below. Features to note are that a kernel size of 11 was used for the first Convolutional layer, 'same' padding was used throughout the model, there is a Global Average Pool in the middle, and there is stacked layers when the filter numbers are equal to 256 and 512.

Table 5: Architecture of the Convolutional Autoencoder.

Layer	**Output Shape**	**Param #**
Reshape	(None, 256, 256, 1)	0
Conv2D	(None, 256, 256, 64)	7808
MaxPool2D	(None, 128, 128, 64)	0
Conv2D_1	(None, 128, 128, 128)	73856
MaxPool2D_1	(None, 64, 64, 128)	0
Conv2D_2	(None, 64, 64, 256)	295168
Conv2D_3	(None, 64, 64, 256)	590080
MaxPool2D_2	(None, 32, 32, 256)	0
Conv2D_4	(None, 32, 32, 512)	1180160
Conv2D_5	(None, 32, 32, 512)	2395808
MaxPool2D_3	(None, 16, 16, 512)	0
GlobalAvgPooling	(None, 512)	0
Flatten	(None, 512)	0
Dense	(None, 4096)	2101248
Conv2DTranspose	(None, 32, 32, 256)	37120
Conv2DTranspose_1	(None, 64, 64, 128)	295040
Conv2DTranspose_2	(None, 128, 128, 64)	73792
Conv2DTranspose_3	(None, 256, 256, 1)	577
Reshape	(None, 256, 256, 1)	0
Total Params:	7,014,657	
Trainable Params:	7,014,657	
Non-trainable Params:	0	

The performance of the Fully Connected Autoencoder was a minimum of validation mean square error = 0.0648, while the Convolutional Autoencoder attained a minimum validation mean square error = 0.0639. This can be visualized by looking at the image reconstructions, which are shown below in Figures 2 and 3.

Notice how the reconstructed images in Figure 2 look as if there is a ghostly image of spheroidite in the background of an otherwise fuzzy image. Clearly the Fully Connected network reconstructions are not dynamic other than the global shade of the image.

The Convolutional network reconstructions do however seem to be more dynamic in the sense that the shading of the reconstructed images are in the correct places. This means that once the model is adjusted further, the Convolutional Autoencoder could reconstruct the images with a lesser degree of loss.

The Fully Connected classifier took the first four layers of its Autoencoder and added 4 dense layers with 200 neurons each, and then an output layer. The Convolutional Classifier was composed of the first 12 layers from the Convolutional Autoencoder, a Flatten Layer, 4 dense layers with 200 neurons each, and the output layer with 6 neurons. One will notice that these upper layer additions are identical. It was thought that a consistent dense network after the transferred layers would lend to a more viable comparison between the two Classifier's lower layers. In hindsight, the dense network for the Convolutional classifier bottlenecks the image information even further. So, for the future, the additional layers for

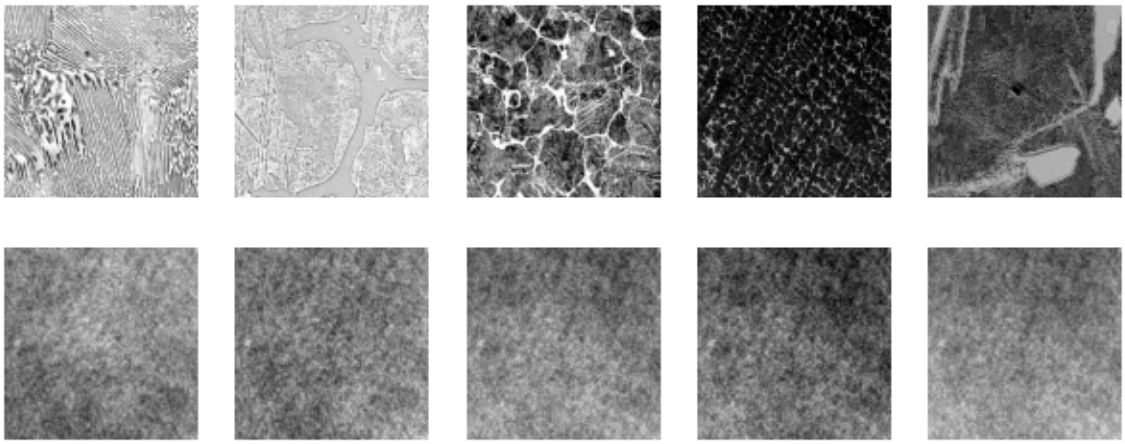

Figure 2: Comparison between the input images (top row) and the reconstructed images (bottom row) for the Fully Connected network. Note that the output images are all very similar to each other.

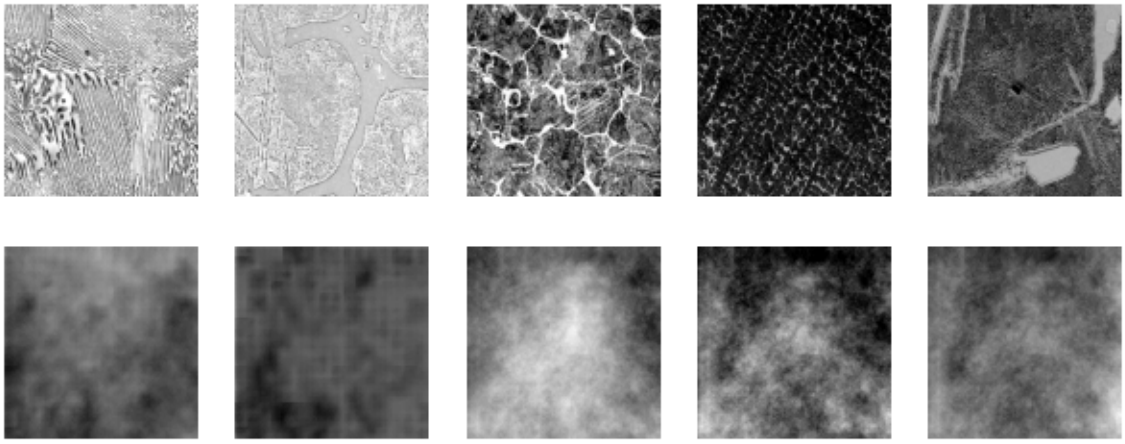

Figure 3: Comparison between the input images (top row) and the reconstructed images (bottom row) for the Convolutional network. The images in the top row are the inputs, while the images on the bottom row are the reconstructions. Something interesting to look further into here is the tiled patterns as seen heavily on the second image from the left on the bottom row. Intuitively, this seems to be related to the output of the Global Average Pooling layer.

this Convolutional network will need to have more neurons than the GlobalAvgPool output.

To tune the Fully Connected model initially the first four layers (From the Fully Connected Autoencoder) were frozen, the model was trained with Stochastic Gradient Descent at a constant learning rate of 0.001 for 4 epochs. Next, the only layers that were frozen were the first three and then training resumed for 6 more epochs. Finally, all layers were unfrozen and the model was trained again for 10 epochs. The tuning for the Convolutional classifier did not involve unfreezing all of the lower layers. The first 12 layers from the Convolutional Autoencoder were frozen and then the model, like the Fully Connected Classifier, was trained with Stochastic Gradient Descent at a learning rate of 0.001 for 4 epochs. Next only the first 7 layers were frozen then it was trained at a learning rate of 0.001, the number of epochs trained is unknown (the cell output was extremely long and the important information was not written down). Both models were compiled with categorical crossentropy loss, and both models were measured with respect to accuracy.

The Fully Connected classifier happens to be little better than a Dummy Classifier that chooses the most common class available to it. It achieves an evaluation accuracy of 62.3%, too close to the class probability of spheroidite. To see how the model performed for all classes, the confusion matrix for this model is shown in Figure 4. The Convolutional classifier also happens to be just as useful as a Dummy Classifier, see Figure 5.

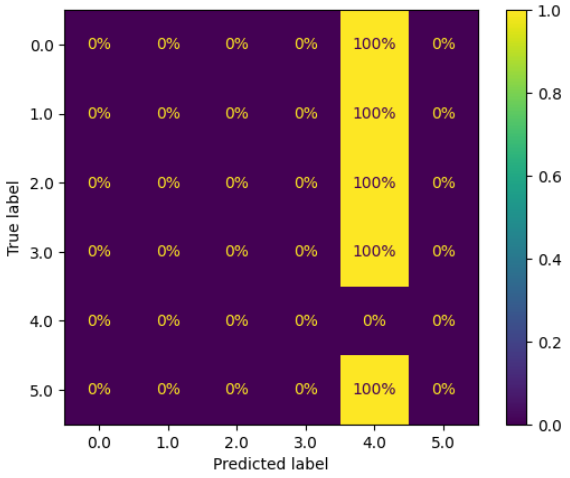

Figure 4: Above is the confusion matrix for the Fully Connected classifier. This model chose class 4 (spheroidite) across the board.

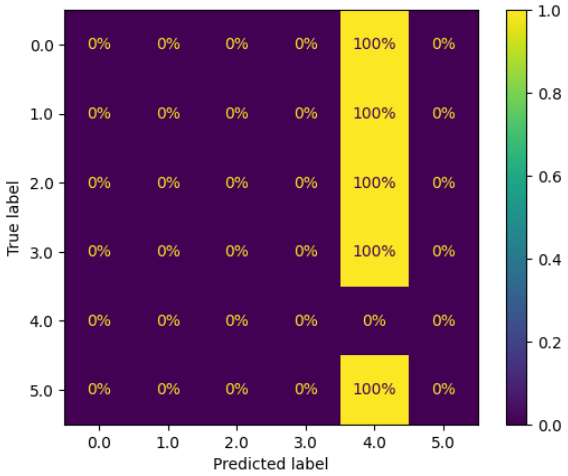

Figure 5: This model chose class 4 (spheroidite) across the board.

5 Conclusion

In light of the failures of the Fully Connected and Convolutional Autoencoders to be able to reconstruct the images to the point where a human could identify microstructures within the reconstructed images, it was figured that the Classifiers had no chance in obtaining a degree of accuracy higher than the highest class occurrence probability. The most frequently occurring microstructure in the images was spheroidite, and that was the class that both models predicted across the board for the entire test set.

Some steps to take in the future to improve the performance are to allow for more information to pass through the Autoencoders. In other words, future Autoencoders should have a larger latent space vector to allow more information to pass through. Another consideration is that the latent space vectors for both Autoencoders were of different sizes. The Fully Connected Autoencoder had a latent space vector of length 750, while that for the Convolutional was 512. Given the the Convolutional Autoencoder was capable of reproducing the correct local shading in its image reconstructions is remarkable compared to the Fully Connected Autoencoder. This means that future steps should focus more on training the Convolutional Autoencoder for the purposes of transfer learning.

References

[1] DeCost, Hecht, Francis, Webler, Picard, and Holm. UHCSDB (Ultrahigh Carbon Steel micrograph DataBase): tools for exploring large heterogeneous microstructure datasets. Accepted for publication in IMMI 2017 DOI: 10.1007/s40192-017-0097-0

`http://hdl.handle.net/11256/940` [Accessed March 25th, 2023].

[2] Safiuddin, Reddy, Vasantada, Harsha, and Dr. Gangolu. Establishing process-structure linkages using Generative Adversarial Networks. `https://arxiv.org/abs/2107.09402`

`https://www.kaggle.com/datasets/safi842/highcarbon-micrographs`[Accessed March 25th, 2023]

6 Appendix

6.1 Appendix A

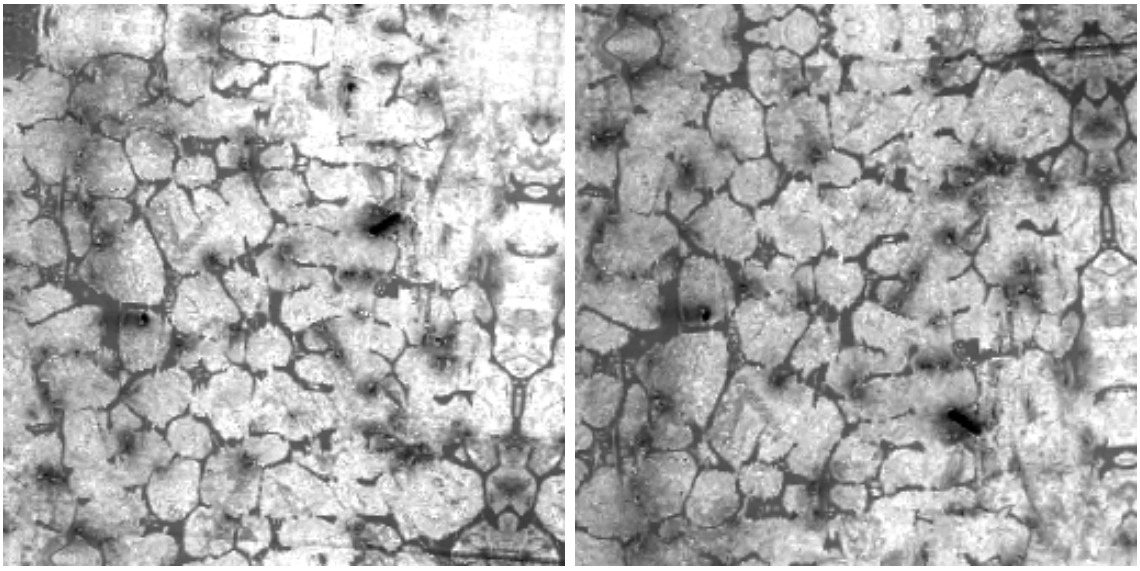

Figure 6: Randomly flipped and slightly channel shifted. One can see mirroring in both images.

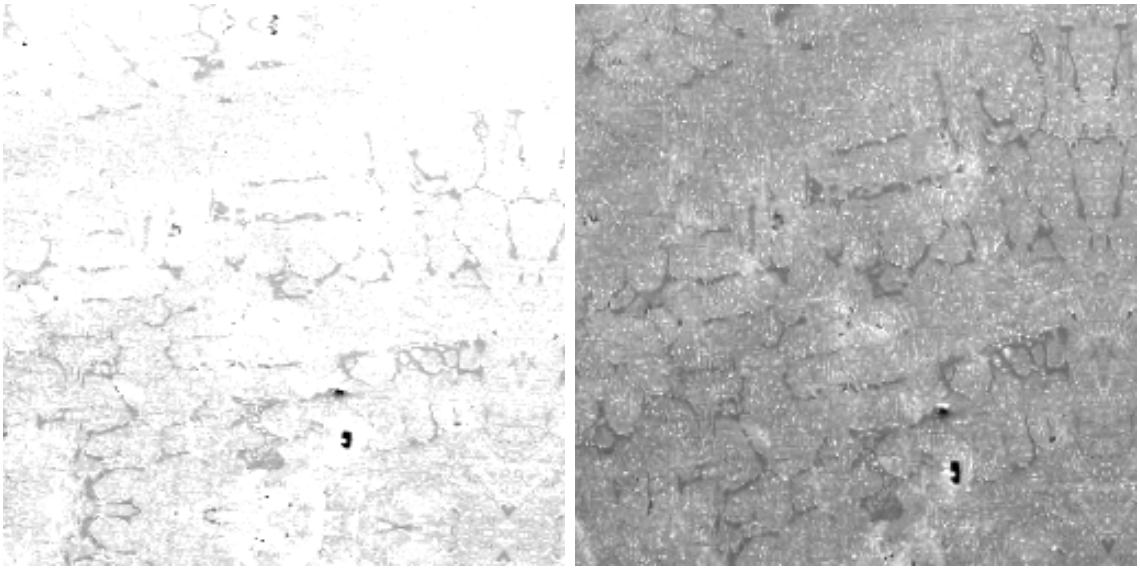

Figure 7: Randomly flipped, rotated and majorly channel shifted.

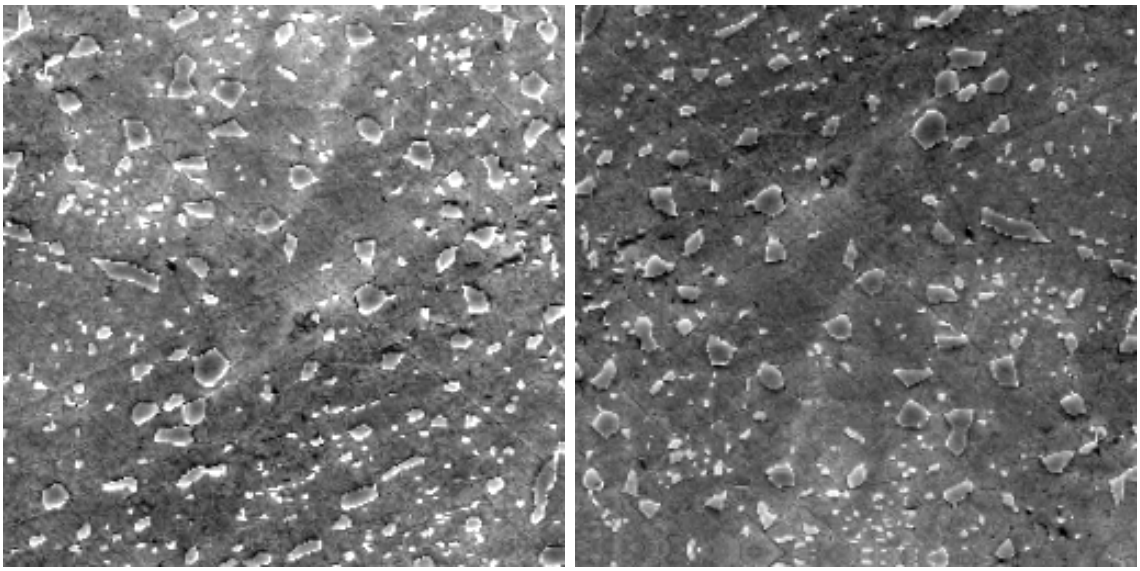

Figure 8: Horizontal and Vertical flip.

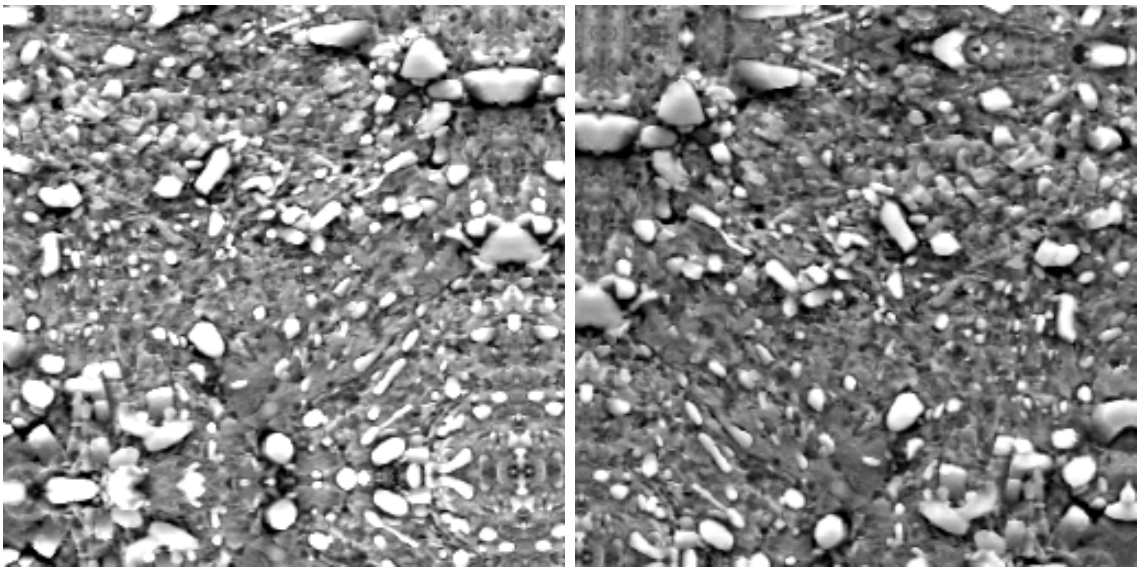

Figure 9: Here one can see the results from the fill mode being set to "reflect". See the bottom right corner of the left image here.

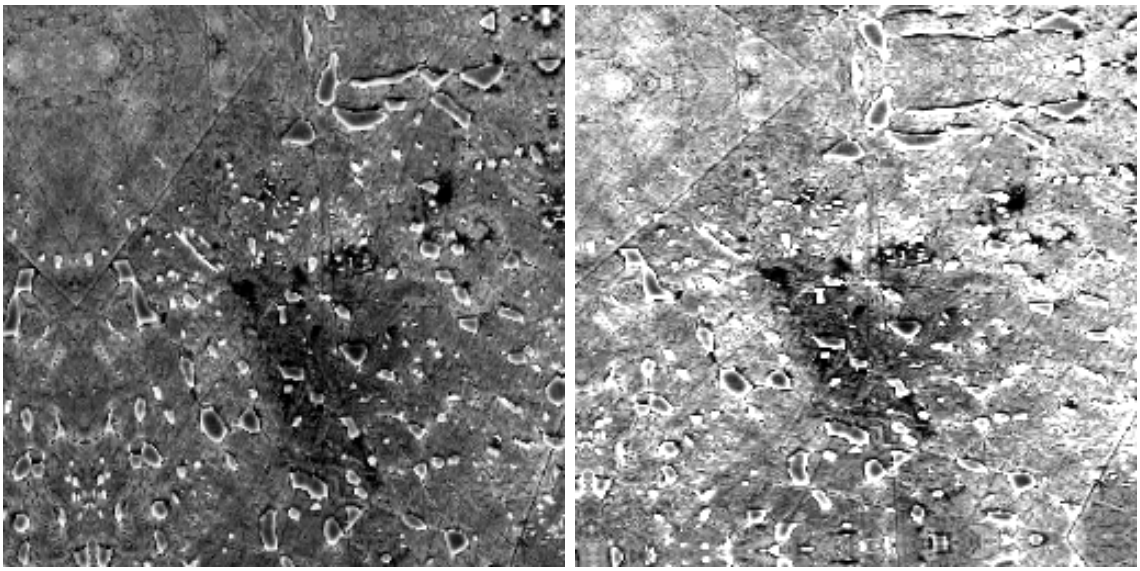

Figure 10: This is another image that contains features that are reflected about the edge of the image. It also seems that the image centers were randomly selected as well.