THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MATERIALS SCIENCE AND ENGINEERING

USING UNSUPERVISED MACHINE LEARNING TO EXAMINE MICROSTRUCTURE OF 17-4 STAINLESS STEEL

THOMAS FEIDA BINA
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Materials Science and Engineering
with honors in Materials Science and Engineering

Reviewed and approved* by the following:

Robert Allen Kimel
Associate Teaching Professor of Materials Science and Engineering
Associate Head for Undergraduate Studies in Materials Science and Engineering
Honors Advisor, Thesis Supervisor

Joan Redwing
Professor of Materials Science and Engineering
Faculty Reader

* Electronic approvals are on file.

ABSTRACT

This work provides an unsupervised machine learning method to examine the microstructure of 17-4 stainless steel. Using a variant of a *k*-means algorithm, features of steel samples imaged via transmission electron microscopy were analyzed and clustered into unique regions. Each of these regions may correspond to an individual phase in the material. This technique did not require a priori description or labeling of the target material system. The described method may be used in an automated manner which has the potential to be effective in rapid identification of phases across a large data set. The work presented here is the first step in developing a larger automated method to identify and characterize the microstructure of 17-4 stainless steel.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The completion of this work would not have been possible without the help and support of numerous individuals. I would like to first thank the NexusLIMS team at NIST – June Lau, Gretchen Greene, Josh Taillon, Ray Plante, and Marcus Newrock – and the entire NIST SURF program for providing me the opportunity to work with them on this project. Their mentorship and guidance have been invaluable for making this project a success. I would like to thank Dr. Mike Katz for providing the images that were used in this work. I am especially grateful to my thesis supervisor Dr. June Lau for providing expertise and guidance through the course of my project and her help in aiding in my professional and personal development.

Thank you to the Penn State Materials Science and Engineering Department to all the people I have met and spent time with during these last four years. Finally, I would like to thank my parents for supporting me for all these years, opening opportunities and helping me to reach my dreams.

Chapter 1 INTRODUCTION

One of the basic principles of materials science is the fact that structure and properties are intimately related. Control of microstructure can be strongly related to the ability to govern the resulting properties of the material. With this consideration in mind, knowledge of material microstructure is understandably paramount for development of materials with desirable properties. Consequently, materials characterization is necessary to understand material microstructure. A key element of materials characterization is the ability to observe and interpret material microstructures. Through the application of various microscopic techniques, images of microstructure can be obtained. Once these images have been obtained, it is then up to scientists and researchers to derive information from these images. However, going forward, computational methods to analyze images should be strongly considered. One method which holds promise for this application is the use of artificial intelligence.

Artificial intelligence-based computational methods are powerful tools which have been applied for great effect across a wide range of disciplines. Within the broader category of artificial intelligence lies a subfield known as machine learning. Machine learning is utilized in many now common applications including speech recognition, image recognition, and the feed of your Facebook account. Many of the predominant large-scale applications of machine learning fall within technologically oriented fields. However, machine learning applications are increasingly finding success in applications related to science and scientific research.

In this work, a machine learning method was developed and applied to analyze the microstructure of 17-4 stainless steel. Specifically, an unsupervised machine learning method was used to find trends in data without the need for a large pre-existing training data set. Utilizing a *k*-means clustering algorithm as a base point, an algorithm to examine the microstructure of 17-4 stainless steel – the most widely used precipitation hardened steel – was developed. Microstructure of the steel was captured via transmission electron microscopy (TEM) imaging. The algorithm was subsequently applied to individual micrographs of the 17-4 stainless steel and was able to identify distinct clusters within the microstructure. These cluster regions may correspond to individual phase microstructures in the metal. This work constitutes a solid first step into the development of a robust methodology for the analysis of the microstructure of 17-4 stainless steel.

1.1 ABET Engineering Considerations

Given the computational nature of this work, the work presented here may not be directly applied to some of the ABET consideration such as environmental issues or social and political issues. However, the application of machine learning approaches has potential for significant impact through its use as an element for automation to increase efficiency. How this idea relates to three of the ABET considerations is briefly addressed in the following sections.

1.1 (a) Manufacturability

As previously stated, one of the key tenets of materials science is the interrelation between structure and properties. Understanding of a material’s microstructure is paramount to controlling the properties and behavior of the product. Control of material properties over industrial-scale production requires significant investment into verification technologies ensuring that correct microstructure is produced. A machine learning approach presents an efficient method to verify material microstructures. Additionally, given enough data to support such an algorithm, a machine learning approach may be more accurate than similar verification processes undertaken by a human which has risks of human error. Such benefits may enable more effective manufacture of materials which require certain elements to be present in their microstructure.

1.1 (b) Economic Issues

From an economic standpoint, the application of machine learning processes is potentially beneficial in industry due to two main reasons. Firstly, automation of a process through machine learning means a lower requirement for human personnel and thereby a lower cost. Secondarily, a machine learning process has potential to decrease costs generally associated with the development and manufacture of a material. As has been emphasized, machine learning has the potential to greatly increase efficiency and accuracy of the manufacturing process. Such an impact should additionally be found beneficial from an economic perspective.

1.1 (c) Sustainability

As mentioned in the previous sections, usage of a machine learning application as is presented here potentially increases efficiency of production by increasing the ability to ensure uniformity in the resulting microstructure. This may manifest over a long term as increased structural stability or wear resistance. These types of material characteristics may be negatively affected by small faults in the microstructure which have a potentially higher probability of being missed given human oversight. This additionally has potential benefits regarding health and safety, for example, if flaws are caught due to the use of machine learning as a verification tool.

Chapter 2 BACKGROUND

2.1 Transmission Electron Microscopy and Microstructural Characterization

An understanding the mechanism behind transmission electron microscopy (TEM) and how it relates to materials characterization, particularly with respect to metals is critical for this work. TEM, generally, functions by transmitting an electron beam through a specimen. Images can then be created by observing how the electron beam interacts with the sample. Electrons are initially generated from a tungsten needle via a thermionic emission process and accelerated by an electron potential towards the sample. Electrostatic and electromagnetic lenses focus the individual electrons into a beam. As the electron beam is transmitted through the specimen, it changes in electron density, phase, and periodicity as the electron beam interacts with the sample. Images are then generated when the electron beam, having passed through the sample, interacts with a screen.[1] A schematic of the overall workings of a TEM microscope is displayed in Figure 1.[2]

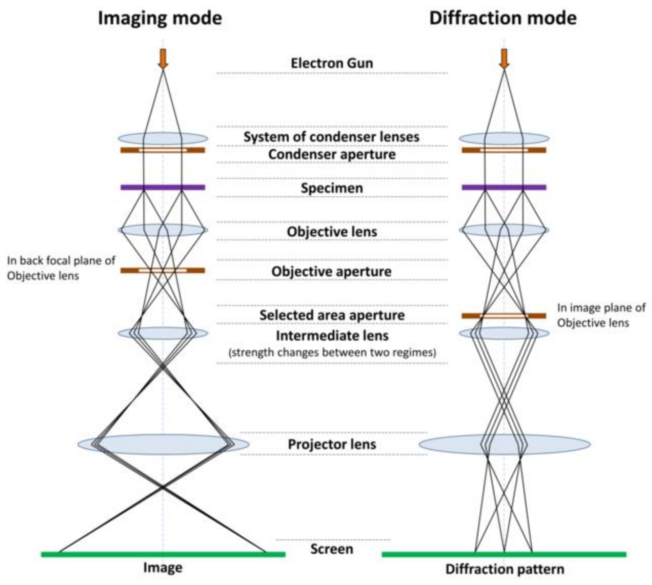

Figure 1: Schematic of a TEM microscope

There are two major modes in which a TEM may be operated: imaging mode and diffraction mode. For this work, only data from imaging TEM was utilized. However, for completeness, diffraction TEM will be briefly discussed. Diffraction imaging is directly related to the unique crystallographic planes which are present in each material. A material's diffraction pattern is directly related to how the electron beam interacts with the interplanar spacing within the material. Analysis of the diffraction pattern can then be used to derive information about the sample. A crystalline material will generate a spread of dots while a polycrystalline or amorphous material will possess a diffraction pattern composed of arcs or rings. An example diffraction pattern for 17-4 stainless steel is shown in Figure 2.

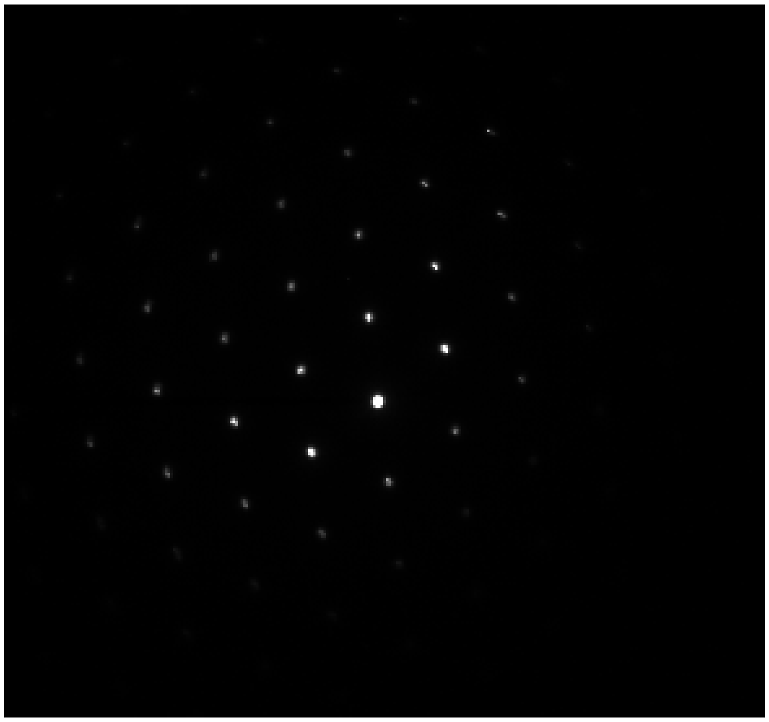

Figure 2: Diffraction pattern of a 17-4 stainless steel sample

TEM's imaging modalities can be further divided into two main categories: bright-field or dark-field imaging. In visual appearance, the images generated by these two modes may be considered opposite. Image contrast found in TEM images is a result of differences in local electron density in the sample. As the electron beam interacts with the sample, the amplitude and phase of the beam changes. These differences are manifested in the image as regions of different color. For a bright-field image, electrons directly transmitted through the sample are captured by the image aperture – thus areas of the sample with high mass or crystallinity appear dark. Conversely, for a dark-field image, the opposite is observed. Electrons scattered from the beam are preferentially captured to form the image, in this case resulting in darkness in areas where the sample is not present.[3] An example of the difference in bright-field versus dark-field imaging modalities is seen in Figure 3.[4]

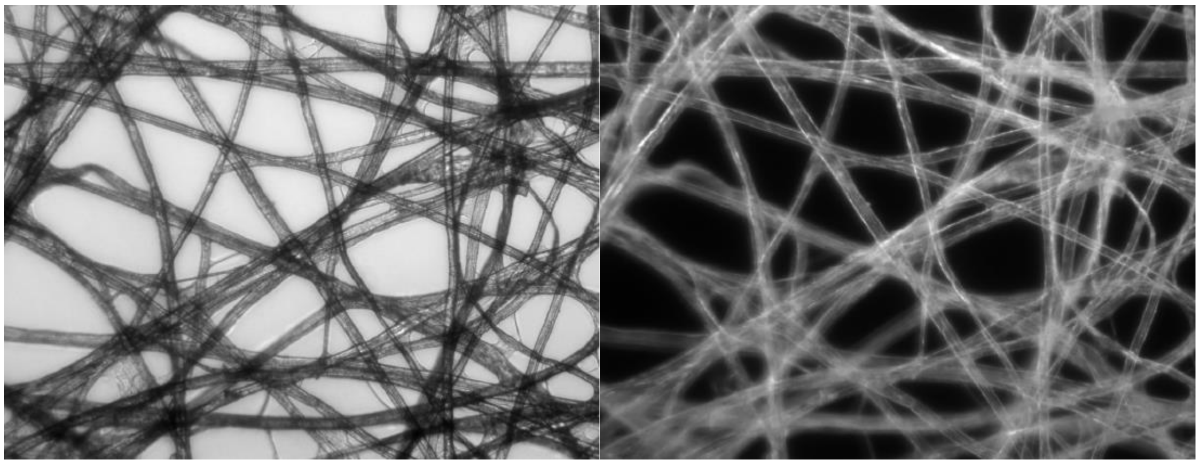

Figure 3: Comparison of bright-field (left) vs. dark-field (right) imaging

2.2 Composition and Microstructure of 17-4 Stainless Steel

Stainless steel is one of the most used structural materials across a wide range of applications. Stainless steel is an iron-based alloy characterized specifically by the presence of chromium. Within the greater stainless-steel category, there are families into which specific alloys fall, the largest of those being austenitic, ferritic, and martensitic stainless steels. 17-4 stainless steel falls into the martensitic family. Additionally, it is a precipitation hardened alloy and is the most used alloy of the precipitation hardened stainless steels.[5] It is heat treated in order to introduce fine particles of impurity phases to impede dislocation movement throughout the metal crystal lattice. This process increases the yield strength of the alloy. The standard alloy composition of 17-4 stainless steel is displayed in Table 1.[6]

Table 1: Standard composition of 17-4 stainless steel

Element	Content (%)
Iron, Fe	73
Chromium, Cr	15.0 - 17.5
Nickel, Ni	3.0 - 5.0
Copper, Cu	3.0 - 5.0
Manganese, Mn	1.0
Silicon, Si	1.0
Tantalum, Ta	0.45
Niobium, Nb (Columbium, Cb)	0.45
Nb + Ta	0.15 - 0.45
Carbon, C	0.070
Phosphorous, P	0.040
Sulfur, S	0.030

Martensitic steels form given a critical cooling rate for which excess carbon in the normal face-centered cubic austenite is unable to diffuse out of the crystal structure. The carbon instead is trapped in interstitial locations in the iron matrix forming a body-centered tetragonal structure. This martensitic form may be considered a supersaturated solid solution of carbon in iron, which strongly contributes to the body-centered tetragonal structure.[7]

The microstructure of 17-4 stainless steel is primarily martensitic in composition with potentially retained austenite and δ ferrite phases. The martensitic phases are generally characterized by either plate or lath martensite, or a combination of both. Microstructure of a martensitic steel is shown in Figure 4.[7] In 17-4 stainless, there may be small precipitates of Cr and Nb carbides in addition to nano-Cu precipitates due to the precipitation hardening process.[8]

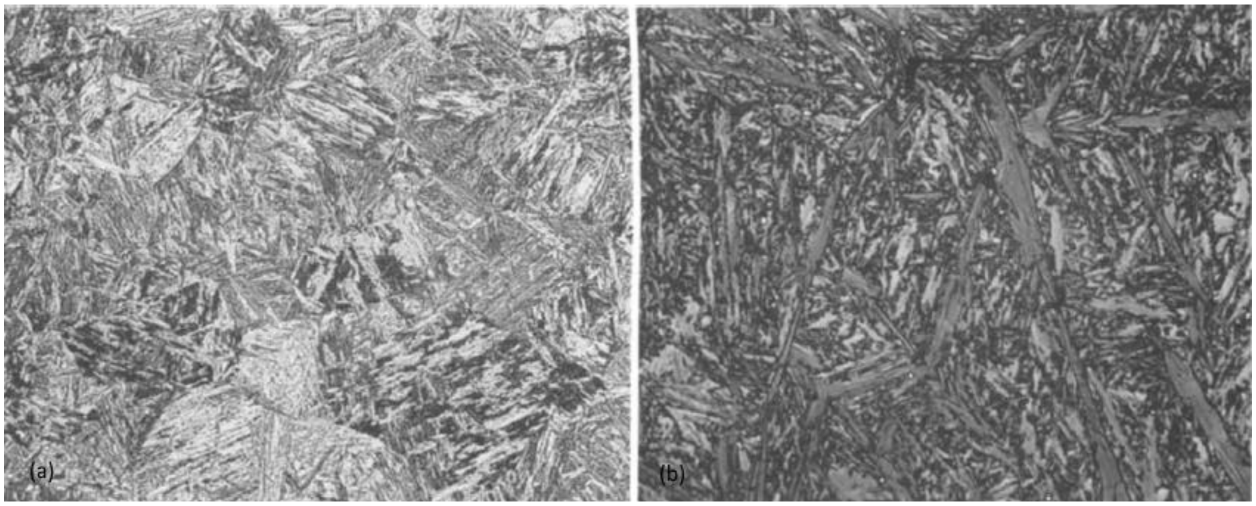

Figure 4: Example microstructures of lath martensite (left) and plate martensite (right)

2.3 Artificial Intelligence and Machine Learning

Artificial intelligence can be considered “intelligence” demonstrated by machines. It can be thought of as using computers to mimic the natural functions associated with the human mind, including capabilities such as learning and responding to stimuli. Within the broader field of artificial intelligence, one large subset field is that of machine learning. As might be gleaned from its name, machine learning is a field of study focused on developing algorithms which are capable of learning. Machine learning uses algorithms and statistical models to perform tasks without the need for explicit instructions given by a user or programmer, instead becoming more accurate over repeated iterations through an exercise. Machine learning algorithms generally rely on observing or establishing patterns in data to improve and accomplish tasks.

What might be the most common and well-known approach to machine learning is known as supervised machine learning. Supervised machine learning is centered around training an algorithm or model through sample data. This training data allows the algorithm to develop a set of rules based on the training data.[9] The training data provides a set of correct input and outputs which the algorithm can then use as a basis to draw conclusions when analyzing real or novel data.

2.3 (a) Unsupervised Machine Learning and Cluster Analysis

For this work, however, a supervised learning approach was not taken. Instead, an unsupervised machine learning algorithm was used. The key point of an unsupervised learning algorithm is that no labelled training data is used. Instead, data is input without the set of correct input and output values that would be used with a supervised learning approach. An unsupervised learning algorithm works by identifying commonalities or trends in a set of data.

One avenue by which this is done is through cluster analysis. This method of unsupervised learning divides the input data into individual objects and then groups those objects into clusters. Objects are defined by some parameter which is used to separate them into clusters. Each cluster is unique from the others because all the objects grouped into that cluster are more alike in some way than the objects placed into other clusters.[10]

One popular methodology of clustering uses centroids to establish clusters. In this methodology, clusters are defined by a central data point which may or may not be a part of the dataset itself. Clusters are assigned around this central data point, referred to as a centroid, by considering some parameter which differentiates the centroid from the input data. For example, a common metric is to define clusters based on Euclidean distance from the centroid. An example of this method of clustering is shown in Figure 5 – data points in the same cluster are visualized to have the same color.

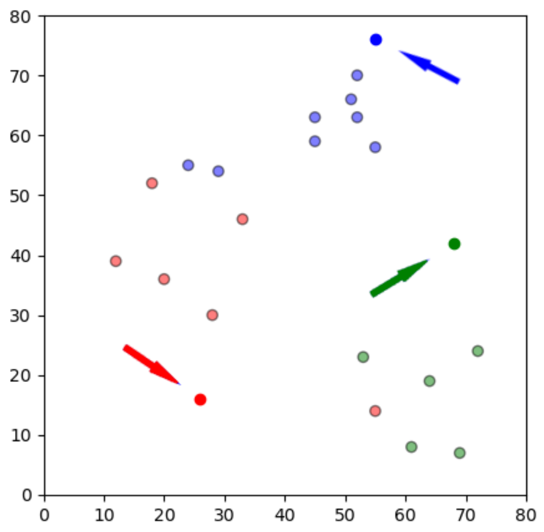

Figure 5: Assignment of data points to centroids (highlighted by arrows)

When the number of centroids – and therefore the number of clusters – is fixed, it is referred to as a *k*-means clustering algorithm, where *k* is the number of allowed clusters. The general goal of a *k*-means approach is to assign all data values into clusters and minimize the variance of the values sorted into each cluster. This goal is typically accomplished by traveling through multiple iterations to refine which data points are included in each specific cluster. Generally, the process is termed "the *k*-means algorithm," though it is also referred to as Lloyd's algorithm.[11] Traditionally, this algorithm entails alternating between an assignment step and an update step. During the assignment step, each object is assigned to a cluster – traditionally, to the cluster with the smallest Euclidean distance. This results in cluster partitions assigned based upon a Voronoi diagram corresponding to the values.[12] An example of a generated Voronoi diagram generated after an iteration is seen in Figure 6.

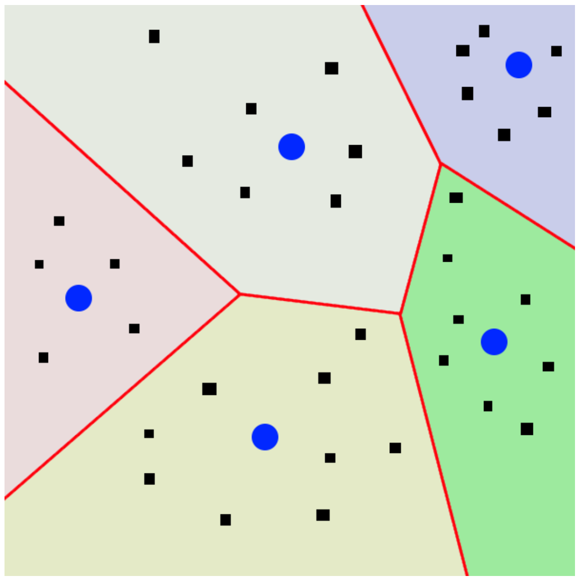

Figure 6: Voronoi diagram possibly generated via a *k*-means clustering iteration

After all objects in the data are assigned to clusters, the algorithm then moves on to the update step. At this point, the algorithm recalculates the value of the centroids. Figure 7 shows how centroids may be updated given certain points of data.[13]

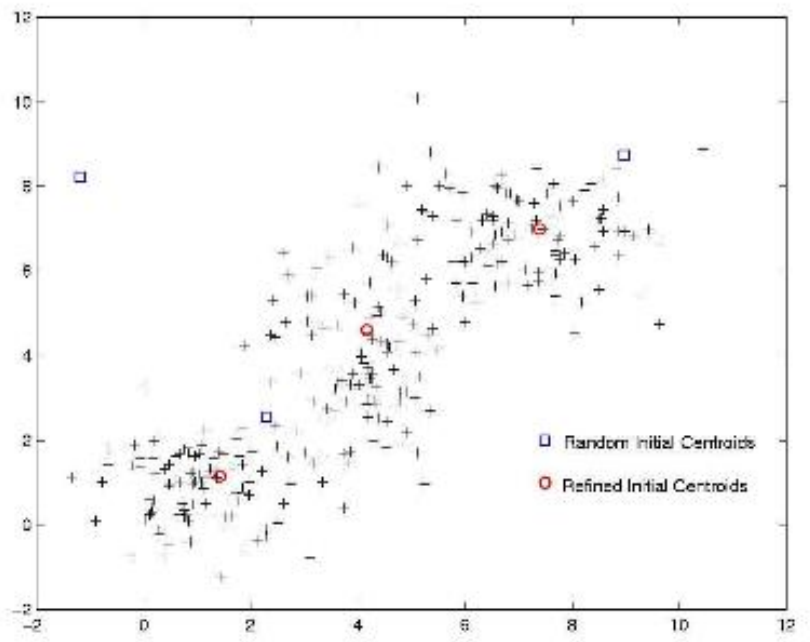

Figure 7: Potential result of a clustering analysis where each cluster is shown with a different color

This update is based upon the new clusters generated in the previous assignment step. Following the update step, the algorithm then re-enters the assignment step, now using the updated centroid to generate clusters. This two-step process continues to repeat until there is no longer a change in centroid values during the update step. In this work, a modified version of a *k*-means algorithm was used, with assignment and update both based upon color difference instead of Euclidean distance.

2.4 Literature Review

Machine learning has been applied in conjunction with microscopy with significant success in the biological field. In this field, machine learning has found perhaps its greatest success in automated identification of cells. Specific identification of several cell types has been accomplished via machine learning. This includes identification of induced-pluripotent stem progenitor cells, white blood cells, and cancerous cells.[14–16] These studies have incorporated both supervised and unsupervised learning approaches in combination with imaging modes beyond electron microscopy.

With regards to microstructural identification in the materials science world, machine learning has also been studied. For individual material systems, unique machine learning approaches must be developed. Research focused on applying machine learning to material microstructure is predominantly focused on classification of a specific morphology in a material system. Pattan, Mytri, and Hiremath developed a neural network-based approach that was able to classify cast iron morphology via differentiation of the graphite grain morphologies that were present. This work utilized a labeled data set comprised of reference images classified by experts.[17] In another work, Chowdhury et. al. developed a classification algorithm to identify dendritic morphology using computer vision and a neural network machine learning approach.[18] In a recent study, Chan et. al. used an unsupervised machine learning technique combining topological classification, image processing, and clustering to both identify and characterize material microstructure in three dimensions. This work was able to examine a broad spectrum of microstructure types by integrating information from a variety of characterization sources.[19]

Chapter 3 METHODOLOGY

3.1 Microscopy of 17-4 Stainless Steel and Data Acquisition

TEM micrographs of steel microstructure were obtained using a 300 keV Titan scanning transmission electron microscope (FEI Company, Hillsboro, OR). Steel samples were examined by a staff scientist at the National Institute of Standards and Technology and images were saved to a central file server specifically designated to host the image data generated by the Nexus facility. The data associated with each image file were saved in proprietary file formats corresponding to the manufacturer of the device used to capture the image. In order to access the data saved in these formats, the HyperSpy Python library was used.[20] HyperSpy allowed for access into the data generated by the detectors. This included access to the images captured by the detectors and the metadata associated with those images. Example images captured and used for analysis are shown in Figure 8.

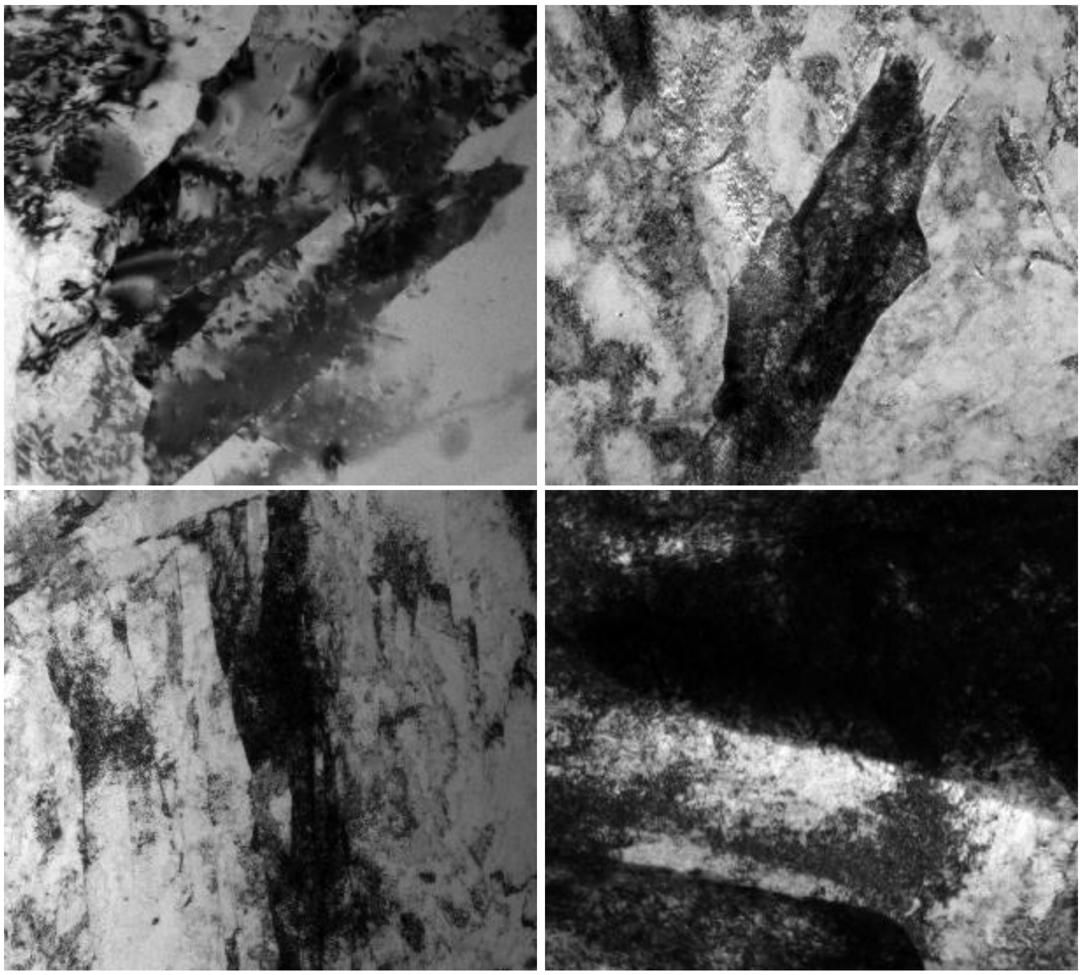

Figure 8: Examples of 17-4 stainless steel TEM micrographs used

For preliminary setup of the machine learning model, only images and data associated with 17-4 stainless steel samples should be utilized. Ideally, this process would be undertaken automatically. However, the image metadata output from the microscope system did not contain enough data labels to automatically parse the 17-4 stainless steel files. Therefore, the 17-4 stainless steel files were instead filtered by utilizing the data labelling system employed by the scientist who captured the images. Each time the scientist used the microscope to analyze a 17-4 stainless steel sample, the files were all saved together, including both microstructural images and diffraction patterns. Thus, the 17-4 stainless steel images were further differentiated by image type such that only the microstructural images were used for the machine learning algorithm. Differentiation of these image types was simply accomplished by filtering using a metadata label which described whether an image was a diffraction pattern or microstructural image. After parsing of the image types, the microstructural images of 17-4 stainless steel then can be utilized for the machine learning algorithm.

3.2 Steel Micrograph Image Interpretation

The first step undertaken was to translate the image file into a dataset which can be used by the computer. This was accomplished by using the Python Imaging Library. The Python Imaging Library was used to load each micrograph image and translate it into a two-dimensional array of values. Within the array, each cell corresponded to an individual pixel in the image, and the data value of each cell corresponded to the RGB color value of the associated pixel. This set of data was then further organized into a DataFrame structure using the Python Data Analysis Library (pandas). Under this data structure, the position of the pixel along the x-axis (i.e. width) and y-axis (i.e. height) was saved, in addition to saving the red, green, and blue color values of the pixel.

3.3 Clustering of Steel Microstructural Features

Centroids were initialized using individual pixels in the image. A variable number of centroids were initialized for each run of the algorithm. This value ranged from three to ten centroids. To initialize each centroid, a random pixel in the image is selected. The centroid then takes on the color value of that pixel. It is that color value which was used to define the initial cluster around that centroid image based upon the color difference of the other pixels from the color of the centroid. An example distribution of randomly initialized centroids is shown in Figure 9, where there is a single pixel selected at each of the highlighted points.

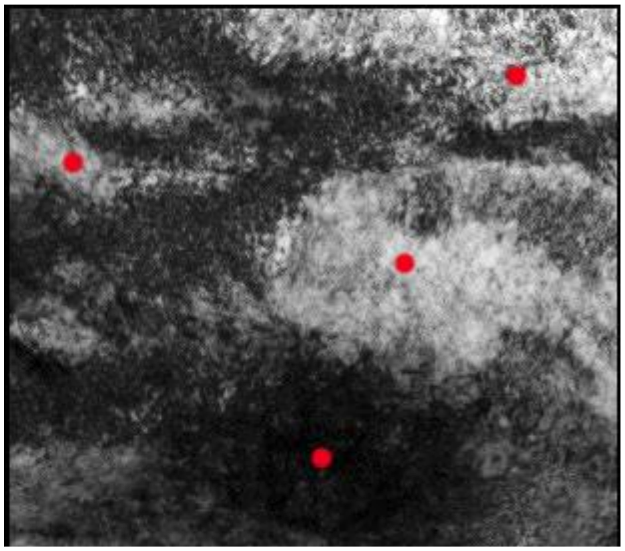

Figure 9: Example distribution of 4 randomly initialized centroids

Color difference between the centroid and the surrounding pixels was evaluated by a difference formula as depicted in Figure 10 where R, G, and B correspond to the red, green, and blue color values of the centroid pixel and surrounding pixel, respectively.

$$\text{color distance} = \sqrt{(R_{centroid} - R_{pixel})^2 + (G_{centroid} - G_{pixel})^2 + (B_{centroid} - B_{pixel})^2}$$

Figure 10: Equation to calculate color distance between a centroid and another pixel

For each centroid in an individual image, the color distance between that pixel and all other pixels in the image was calculated. Once the color difference for all pairs of centroid pixel and surrounding pixels was calculated, every pixel in the image was assigned to a specific cluster. Clusters were assigned by based upon the smallest color difference between the pixel and the centroid; for each pixel, the centroid to which it had the closest color was the cluster to which the pixel was assigned. An example initial cluster pattern is shown in Figure 11.

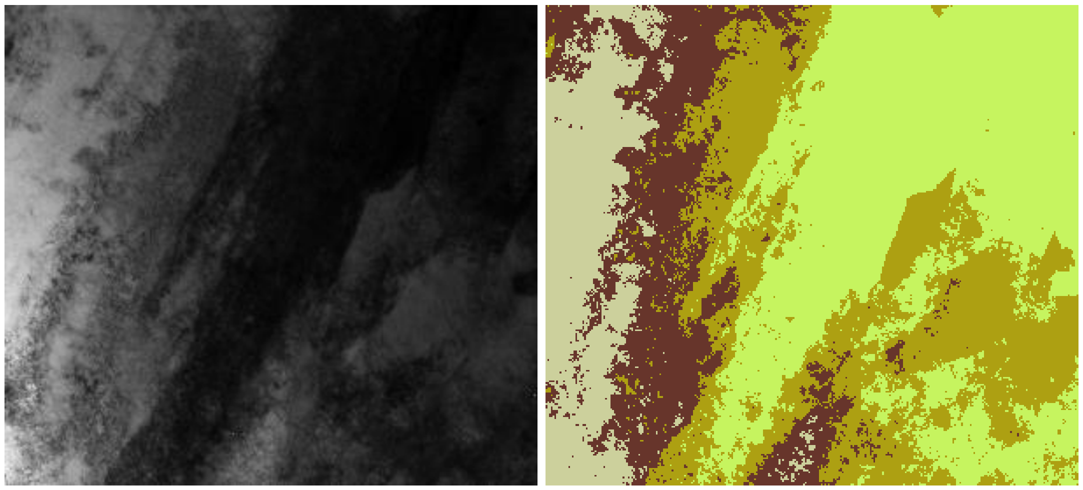

Figure 11: Clusters established (right) for the left image using the initial centroid values

After clusters were assigned, centroid values were updated by taking the average red, green, and blue colors of all pixels within that cluster. The resulting average values were used as the updated centroid colors; thereafter each centroid no longer corresponded to a specific pixel but instead had its own value. After the value assigned to each centroid was updated, the color difference of each pixel in the image was again calculated against the updated centroid value. Following this new set of calculations, each pixel in the image was once again assigned to a new cluster based on the smallest color difference between the pixel and the centroid. The color value of each centroid was then updated again by taking the average color of all the pixels assigned to the corresponding cluster. This process of updating centroids and reassigning pixels to clusters continued until there was minimal change in the new centroid values – specifically that none of the updated centroids differed from their previous values by greater than two units in the RGB color system. A progression in the clusters over four iterations is shown in Figure 12.

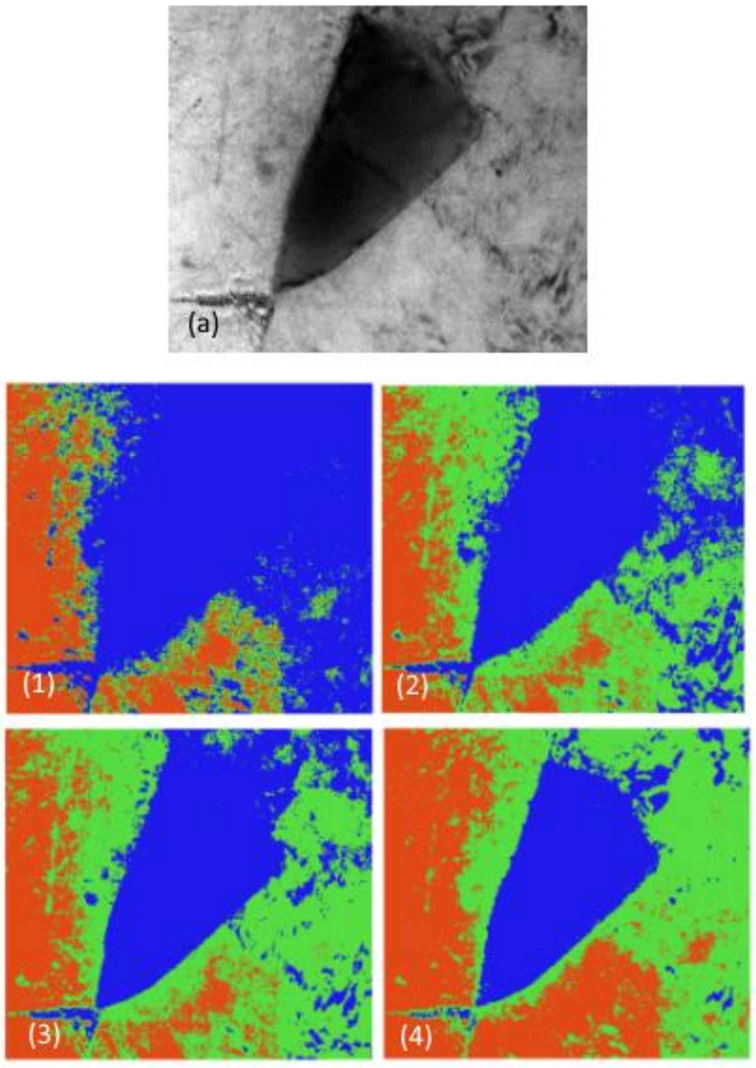

Figure 12: Iterations of the assignment and update process to reach stability in the centroid values

Chapter 4 RESULTS AND DISCUSSION

The *k*-means algorithm was able to accurately create clusters using the electron micrographs as a basis. Clusters determined by the algorithm for some micrographs are shown in Figure 13. The clusters accurately match the microstructural features seen in the TEM image. However, there are points to consider with regards to the effectiveness of the identification algorithm.

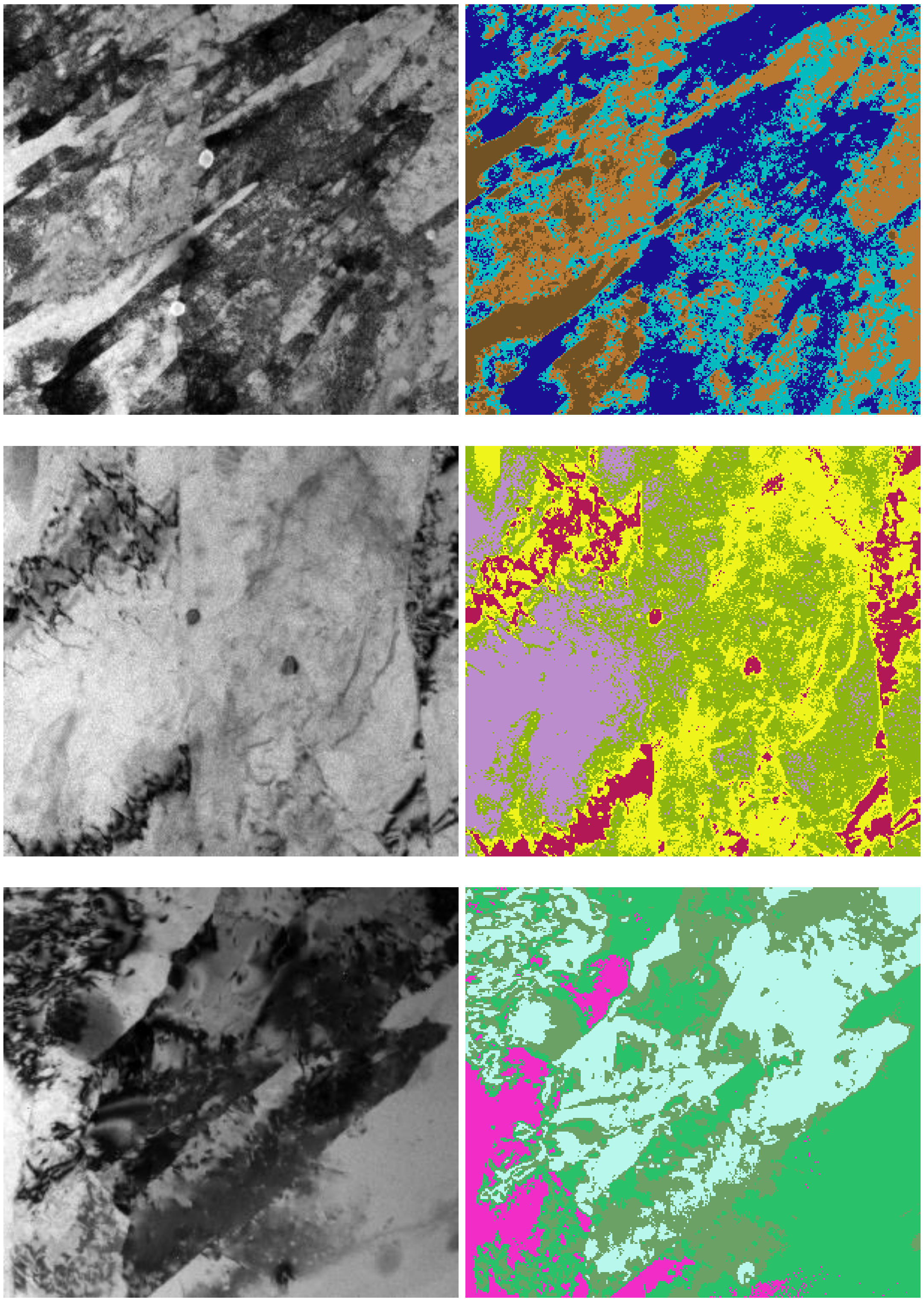

Figure 13: Clusters (right) defined by the algorithm for each of the adjacent images.

***k* = 4 for all images presented**

One concern of the method described here is the randomized initialization of the centroids. Because the centroids are randomly initialized, there is a potential for inconsistencies in the resulting clusters identified. While the iterative update process should compensate for the randomized initialization, in the most extreme case there still exists the potential for entire phases in the material to be ignored, particularly if they are a minor phase. All images seen in Figure 14 are derived from the same microstructure and have the same number of clusters (*k* value). The position of the initial pixel used as the initial centroid is highlighted for each of the micrographs. Because the initial centroid values are taken at random, there is some degree of variation in the results of the algorithm, particularly in regions with more complex microstructure. However, when observing the differences between the entire cluster pattern for each respective image, there is small apparent difference. The algorithm was particularly effective at accurately clustering large areas of a single phase together, as can also be seen in Figure 14.

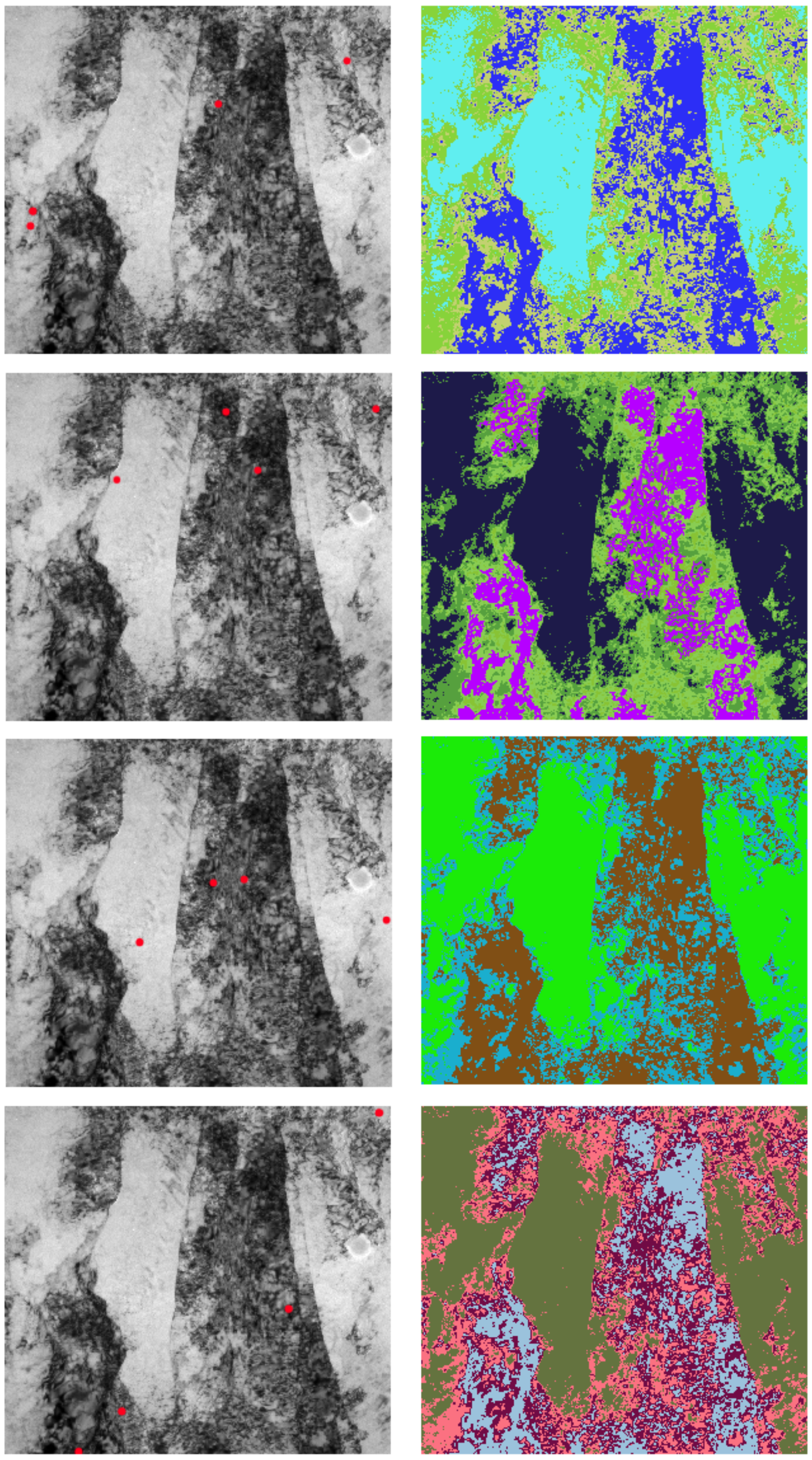

Figure 14: Clusters defined, all with *k* =4, but with differing initial centroids. Initial centroid positions are highlighted on the original micrographs

A key point of consideration is *k*, the number of clusters that was assigned for each image. Because *k* must be pre-defined for the algorithm to function, there exists the possibility that phases can be either over-accounted or under-accounted for. Figure 15 shows images of the same micrograph with differing *k* values. Proceeding from image (a) to image (d), *k* is three, four, six, and ten, respectively. It can be seen in this example that an increased *k* value results in additional clusters being defined at cluster edges. At regions where clusters border each other given lower *k* values, the clustering using an increased *k* value creates a new cluster along the border which previously existed. This may be indicative of over-counting of the number of phases in the material. Particularly in the image with $k = 10$, the features seen in the micrograph are over-divided into clusters, no longer properly correlating to the actual division of phases.

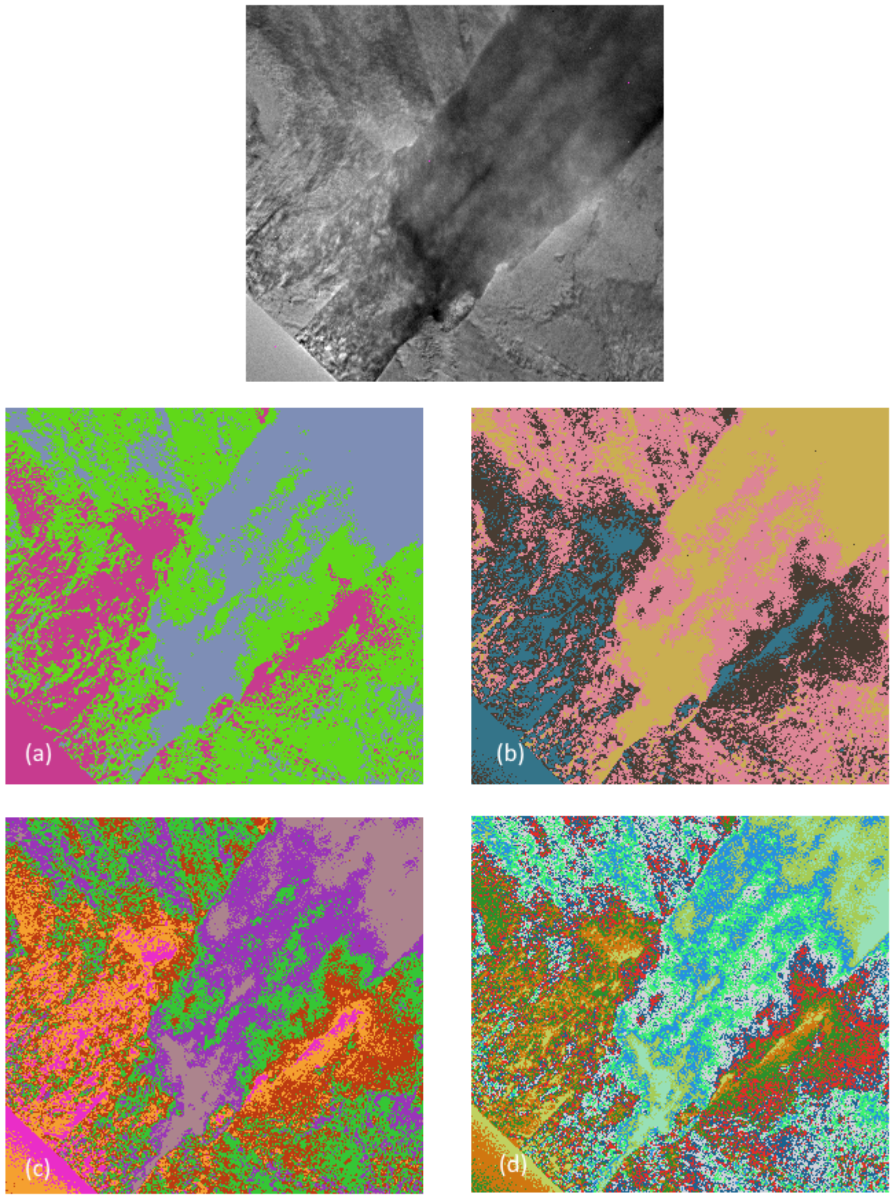

Figure 15: Each cluster pattern originates from the same micrograph but has a different *k* value of 3, 4, 6, and 10, respectively

There is some degree of "fuzziness" at the edges of clusters. Particularly for micrographs with a lower resolution, there is not a clean, high contrast edge which differentiates phases in the material. This factor can be particularly prominent when there are finer phases interspersed within a larger major phase, as seen in Figure 16. In the micrographs at lower resolution or with more finely interspersed phase features, a color gradient is found at boundaries. It is particularly at these areas of gradient that finer microstructural features are lost during the assignment of clusters. This gradient is then interpreted as a single phase in the microstructure, instead of a mixed-phase region.

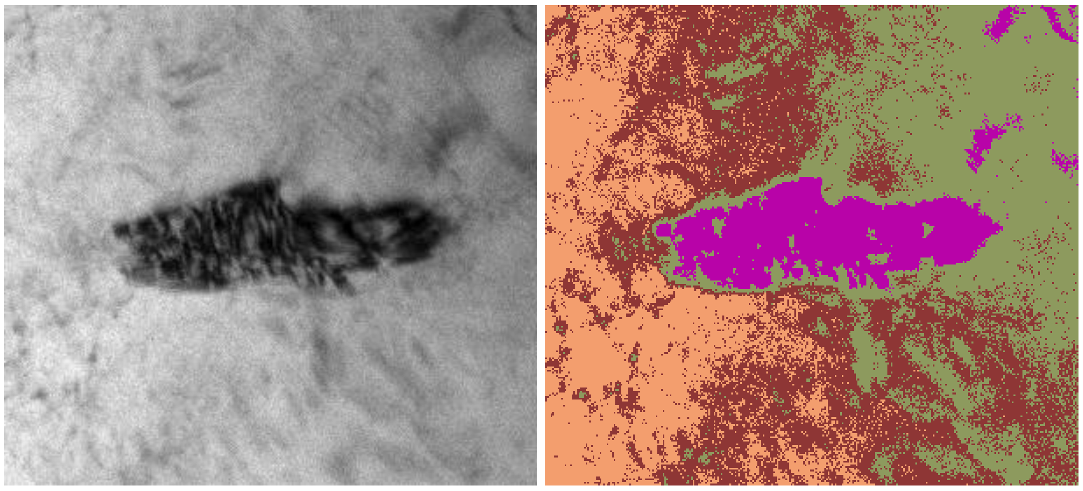

Figure 16: Loss of cluster resolution due to interspersion of features in the microstructure

4.1 Limitations

The largest limitation of the present study is an intrinsic characteristic of an unsupervised machine learning methodology. It is difficult to verify that the results of the algorithm are accurate in a vacuum. This is difficult due to the nature of an unsupervised method which does not utilize a dataset of known inputs and outputs to obtain results, as would be used for a supervised learning method for example. Because the algorithm functions insularly on the input data by finding patterns, the meaning of the result may not necessarily be the intended output –

in the case of this work, microstructural features of the steel. As such, verification procedures are necessary. Such procedures may include cross examining the algorithm's results against known data, for example.

The usage of a *k*-means-based algorithm presents some difficulty due to the requirement for a predefined *k* value. In the context of this work, the *k* value corresponds loosely to the number of unique phases present in the microstructure. As such, knowledge of the number of phases present in the material is necessary for the *k*-means algorithm to function most effectively. This factor significantly limits the application of the algorithm as it is presented in this work for uses as a discovery tool. In a case in which the number of phases is unknown, or a specific phase is sought after, it would be difficult to use a *k*-means algorithm effectively. That said, in cases where the system is well-known, the algorithm presented here has potential to be useful in automated processing of micrographs or in verification procedures ensuring the material is presented as expected.

Chapter 5 CONCLUSIONS AND FUTURE DIRECTION

This work successfully developed an unsupervised machine learning method which identified patterns in the microstructure of 17-4 stainless steel. Through the machine learning algorithm, the microstructure of 17-4 stainless steel was successfully divided into regions which potentially correspond to individual phases in the metal. The work presented here can is a first step into applying machine learning to analyze the microstructure of 17-4 stainless steel. Such a method presents possible significant benefits to increase both precision and efficiency in the characterization of the microstructure of 17-4 stainless steel. Even beyond the stainless steel material system, a fully developed machine learning approach has the potential to greatly improve microstructural characterization in general.

5.1 Future Directions of Work

It would be beneficial to continue analysis of the clusters generated by the algorithm. In its current form, the algorithm operates on a single micrograph to create clusters and thereby examine microstructure. Because it uses only a single image to generate clusters, the algorithm is especially sensitive to defects and other flaws which may be present in the material. This can be compensated for in future study by performing analysis on the clusters that are generated by the algorithm. A potential avenue would be to gain an understanding of the trends generated clusters across a large dataset. This includes an analysis of the color associated with each of the clusters identified as well as an analysis of the morphology of the clusters defined. Indeed, one of the more powerful potential applications which could result from this work is the ability to use the clusters defined by the algorithm presented here as data for further analysis of feature morphology, for example.

Another avenue for future work, is creation and application of a supervised learning-based algorithm to pursue analysis of the microstructure of 17-4 stainless steel. Using such an approach may present more accurate outcomes as compared to an unsupervised method. Certainly, it would be interesting to compare the results of each methodology. However, application of this method would require access to a sizable set of data which would need to be labelled. Such a data set may be difficult to acquire and process for such a purpose.

REFERENCES

1. Nellist, P. D. The Principles of STEM Imaging. in *Scanning Transmission Electron Microscopy: Imaging and Analysis* (eds. Pennycook, S. J. & Nellist, P. D.) 91–115 (Springer, 2011). doi:10.1007/978-1-4419-7200-2_2.
2. Kvaalen, E. *Schematic view of imaging and diffraction modes in TEM*.
3. Gauvin, R. Review of transmission electron microscopy for the characterization of materials. in *Materials Characterization and Optical Probe Techniques: A Critical Review* vol. 10291 102910C (International Society for Optics and Photonics, 1997).
4. TEM: Bright field versus dark field. *Chemistry LibreTexts* https://chem.libretexts.org/Courses/Franklin_and_Marshall_College/Introduction_to_Materials_Characterization_-_CHM_412_Collaborative_Text/Electron_and_Probe_Microscopy/TEM%3A_Bright_field_versus_dark_field (2019).
5. Smith, W. F. Structure and properties of engineering alloys. *CERN Document Server* https://cds.cern.ch/record/441615 (1993).
6. Stainless Steel - Grade 17-4 (UNS S17400). *AZoM.com* https://www.azom.com/article.aspx?ArticleID=6778 (2012).
7. Structure/Property Relationships in Irons and Steels. in *Metals Handbook Desk Edition* (ed. Davis, J. R.) 153–173 (ASM International, 1998). doi:10.31399/asm.hb.mhde2.a0003090.
8. *Metals Handbook Desk Edition*. (ASM International, 1998). doi:10.31399/asm.hb.mhde2.9781627081993.
9. Learned-Miller, E. G. Introduction to Supervised Learning. 5.

10. MacKay, D. J. C. Information Theory, Inference, and Learning Algorithms. 640.

11. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**, 129–137 (1982).

12. Bakolas, E. & Tsiotras, P. The Zermelo–Voronoi diagram: A dynamic partition problem. *Automatica* **46**, 2059–2067 (2010).

13. Bradley, P. S. & Fayyad, U. M. Refining Initial Points for K-Means Clustering. 10.

14. Zhang, H. *et al.* A novel machine learning based approach for iPS progenitor cell identification. *PLOS Comput. Biol.* **15**, e1007351 (2019).

15. Nassar, M. *et al.* Label-Free Identification of White Blood Cells Using Machine Learning. *Cytometry A* **95**, 836–842 (2019).

16. Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V. & Fotiadis, D. I. Machine learning applications in cancer prognosis and prediction. *Comput. Struct. Biotechnol. J.* **13**, 8–17 (2015).

17. Pattan, P. C., Mytri, V. D. & Hiremath, P. S. Classification of cast iron based on graphite grain morphology using neural network approach. in *Second International Conference on Digital Image Processing* vol. 7546 75462S (International Society for Optics and Photonics, 2010).

18. Chowdhury, A., Kautz, E., Yener, B. & Lewis, D. Image driven machine learning methods for microstructure recognition. *Comput. Mater. Sci.* **123**, 176–187 (2016).

19. Chan, H., Cherukara, M., Loeffler, T. D., Narayanan, B. & Sankaranarayanan, S. K. R. S. Machine learning enabled autonomous microstructural characterization in 3D samples. *Npj Comput. Mater.* **6**, 1–9 (2020).

20. Francisco de la Peña *et al. hyperspy/hyperspy: HyperSpy v1.5.2*. (Zenodo, 2019). doi:10.5281/zenodo.3396791.

APPENDIX

Code used:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import os
import numpy as np
import pandas as pd
import math
import copy
import random
import time
from PIL import Image


# In[2]:


file_location = 'C:/Users/tfbin/Desktop/steel images/'
files = []
for file in os.listdir(file_location):
    files.append(os.path.join(file_location, file))

# crop all HyperSpy generated plots to only microscope image
crop_left = 25
crop_right = 329
crop_top = 46
crop_bottom = 320

for image in files:
    temp = Image.open(image)
    cropped = temp.crop((crop_left, crop_top, crop_right, crop_bottom))
    cropped.save('C:/Users/tfbin/Desktop/Cropped Images/' + image.split('/')[-1])


# In[3]:


cropped_location = 'C:/Users/tfbin/Desktop/Cropped Images/'
# create list of file paths to all cropped images
cropped_image_locations = []
for file in os.listdir(cropped_location):
    cropped_image_locations.append(os.path.join(cropped_location, file))

# create list of all cropped images
cropped_images = []
for location in cropped_image_locations:
    cropped_images.append(Image.open(location))


# In[4]:


# work with only one image initially
file_num = 0
working_image = cropped_images[file_num]
pixels = working_image.load()
copied_image = working_image.copy()
pixel_copy = copied_image.load()

# k = number of centroids
k = 5
centroids = {
    i+1: [np.random.randint(0, working_image.width), np.random.randint(0, working_image.height)]
    for i in range(k)
}

# set of color values each cluster will be shown as when viewing
cluster_colors = []
for i in range(k):
    cluster_colors.append([random.randint(1,255), random.randint(1,255), random.randint(1,255)])

initial_centroid_colors = []
for _ in centroids.keys():
    centroid_coord = centroids[_]
    centroid_rgb = pixels[centroid_coord[0], centroid_coord[1]]
    initial_centroid_colors.append(list(centroid_rgb)[0:-1])

# change the centroid pixels to pink for visualization
for i in centroids.keys():
    temp_coord = centroids[i]
    pixel_copy[temp_coord[0], temp_coord[1]] = (255, 51, 236)

copied_image.save('C:/Users/tfbin/Desktop/Figures/{}'.format(cropped_image_locations[file_num].split('/')[-1]+' initial.png'))


# In[5]:


# create dataframe with coordinates and color for each pixel of an image
image_column_names = ['x', 'y', 'R', 'G', 'B']
image_data = pd.DataFrame(columns = image_column_names)
iteration = 0
for i in range(working_image.width):
    for j in range(working_image.height):
        rgb = pixels[i,j]
        image_data.loc[iteration] = [i, j, rgb[0], rgb[1], rgb[2]]
        iteration = iteration + 1


        # In[6]:


        # create dataframe which will have the spacial and color distances from the centroids for each pixel
        diff_data = pd.DataFrame(columns=None)

        for i in centroids.keys():
```

```
        centroid_coord = centroids[i]
        centroid_color = pixels[centroid_coord[0], centroid_coord[1]]

        temp_columns = ['distance_from_{}'.format(i), 'color_diff_from_{}'.format(i)]

        temp_df = pd.DataFrame(columns = temp_columns)

        diff_data = pd.merge(temp_df, diff_data, how='outer', left_index=True, right_index=True)

    diff_data = diff_data.iloc[:, ::-1] # reverse order so centroid 1 is first, then 2 - 3 - etc.


    # In[7]:


    #Iterate through image_data, calculate differences, and assign to diff_data dataframe
    for index, row in image_data.iterrows():
        pixel_rgb = [row['R'], row['G'], row['B']]
        pixel_coord = [row['x'], row['y']]

        temp_values = []  # temporary arry containing difference values from each centroid e.g. [color_diff_from_1,
                          # distance_from_1, color_diff_from_2, distance_from_2, etc.]
        for i in centroids.keys():
            centroid_coord = centroids[i]
            centroid_color = pixels[centroid_coord[0], centroid_coord[1]]

            # append color difference from centroid
            color_diff = math.sqrt((centroid_color[0] - pixel_rgb[0])**2 + (centroid_color[1] - pixel_rgb[1])**2                + (centroid_color[2] - pixel_rgb[2])**2)
            temp_values.append(color_diff)

            # append linear pixel distance from centroid
            distance = math.sqrt((centroid_coord[0] - pixel_coord[0])**2 + (centroid_coord[1] - pixel_coord[1])**2)
            temp_values.append(distance)

        diff_data.loc[index] = temp_values

    df = pd.DataFrame.join(image_data, diff_data) # combine image_data and diff_data into one dataframe


    # In[8]:


    # create and initialize dictionary containing empty lists, one for each centroid & associated cluster
    clusters = {}
    for i in range(k):
        clusters['{}'.format(i+1)] = []

    # assign pixels to clusters based upon smallest color difference
    # - > update to create centroids of a specific color, not a specific pixel
    for index, row in df.iterrows():
        color_differences = [] # array to hold color_diff values from each centroid for the pixel selected in the loop
        for i in range(k):
            color_differences.append(row['color_diff_from_{}'.format(i+1)])

        assigned_cluster = color_differences.index(min(color_differences)) + 1
        clusters[str(assigned_cluster)].append(index)


    # In[9]:


    # update centroid color values based on newly defined clusters
    updated_centroid_colors = []
    for key in clusters:
        temp_df = image_data.loc[clusters[key], ['R', 'G', 'B']]
        avg_rgb_vals = temp_df.mean(axis=0)
        updated_centroid_colors.append(list(avg_rgb_vals.iteritems()))


    # In[10]:

```

```

# find color differences between the updated centroids and initial centroids, for each centroid
color_diffs_after_update = []
for i in range(k): # iterate through each centroid
    temp_updated = updated_centroid_colors[i]
    temp_updated = [x[1] for x in temp_updated] # select out second value from each of the RGB tuples
    temp_initial = initial_centroid_colors[i]
    # calculate color difference between previous and updated centroid RGB values

    color_diff = math.sqrt((temp_initial[0] - temp_updated[0])**2 + (temp_initial[1] - temp_updated[1])**2                                + (temp_initial[2] - temp_updated[2])**2)
    color_diffs_after_update.append(color_diff)


# In[11]:


# initialize DataFrame with columns: INDEX COLOR_DIFF_FROM_1..2..3.. to track color differences against new centroids
color_diff_from = []
for i in range(k):
    color_diff_from.append('color_diff_from_{}'.format(i+1))
diff_df = diff_data.loc[:, color_diff_from]

num_updates = 1
# looping update process
while max(color_diffs_after_update) > 2:
    # colorize & save figure showing clusters for each loop
    iteration = 0
    for key in clusters:
        temp_df = image_data.loc[clusters[key], :]
        for index, row in temp_df.iterrows():
            pixel_copy[row['x'], row['y']] = cluster_colors[iteration][0], cluster_colors[iteration][1], cluster_colors[iteration][2])
        iteration = iteration + 1

    copied_image.save('C:/Users/tfbin/Desktop/Figures/{}'.format(cropped_image_locations[file_num].split('/')[-1]+' '+str(num_updates)+'.png'))
```

```

        # keep track of number of loops
        num_updates = num_updates + 1

        # recalculate color difference between pixels and new centroid colors
        for index, row in image_data.iterrows():
            pixel_rgb = [row['R'], row['G'], row['B']]

            temp_values = []  # temporary arry containing difference values from each centroid e.g. [color_diff_from_1,
                                     # color_diff_from_2, color_diff_from_3, etc.]

            for i in updated_centroid_colors:
                centroid_rgb = [x[1] for x in i]
                # append color difference from centroid
                color_diff = math.sqrt((centroid_rgb[0] - pixel_rgb[0])**2 + (centroid_rgb[1] - pixel_rgb[1])**2                                + (centroid_rgb[2] - pixel_rgb[2])**2)
                temp_values.append(color_diff)

            #update diff_df DataFrame with new row of color difference values
            for i in range(k):
                diff_df.at[index, 'color_diff_from_{}'.format(i+1)] = temp_values[i]

        # create and initialize dictionary containing empty lists, one for each centroid & associated cluster
        clusters = {}
        for i in range(k):
            clusters['{}'.format(i+1)] = []

        # assign pixels to clusters based upon smallest color difference
        for index, row in diff_df.iterrows():
            # array to hold color_diff values from each centroid for the pixel currently selected in the loop
            color_differences = []
            for i in range(k):
                color_differences.append(row['color_diff_from_{}'.format(i+1)])

            assigned_cluster = color_differences.index(min(color_differences)) + 1
            clusters[str(assigned_cluster)].append(index)

```

```
    # update centroid colors based on average values of previous clusters
    previous_centroid_colors = copy.deepcopy(updated_centroid_colors)
    updated_centroid_colors = []
    for key in clusters:
        temp_df = image_data.loc[clusters[key], ['R', 'G', 'B']]
        avg_rgb_vals = temp_df.mean(axis=0)
        updated_centroid_colors.append(list(avg_rgb_vals.iteritems()))

    # find color differences between the updated centroids and updated centroids, for each centroid
    color_diffs_after_update = []
    for i in range(k): # iterate through each centroid
        temp_updated = updated_centroid_colors[i]
        temp_updated = [x[1] for x in temp_updated]
        temp_previous = previous_centroid_colors[i]
        temp_previous = [x[1] for x in temp_previous]
        # calculate color difference between previous and updated centroid RGB values
        color_diff = math.sqrt((temp_previous[0] - temp_updated[0])**2 + (temp_previous[1] - temp_updated[1])**2                              + (temp_previous[2] - temp_updated[2])**2)
        color_diffs_after_update.append(color_diff)
```

ACADEMIC VITA

THOMAS BINA

EDUCATION

The Pennsylvania State University, Schreyer Honors College

B.S.: Materials Science and Engineering, Minor: Microbiology

RESEARCH EXPERIENCE

National Institute of Standards & Technology (NIST), Materials Measurement Laboratory
Guest Research Fellow Jun 2019 – May 2020
Supervisor: Dr. June Lau

Penn State University, Department of Bioengineering, Materials Research Institute
Undergraduate Research Assistant Sept 2018 – May 2019
Supervisor: Dr. Siyang Zheng

University of Pittsburgh, Department of Mechanical Engineering and Materials Science
Undergraduate Research Assistant May 2018 – Aug 2018
Supervisor: Dr. Anne Robertson

Penn State University, Department of Materials Science and Engineering
Undergraduate Research Assistant Jan 2018 – May 2018
Supervisor: Dr. James Adair

University of Pittsburgh, Department of Chemistry
Undergraduate Research Assistant May 2017 – Aug 2017
Supervisor: Dr. Geoff Hutchison

PUBLICATIONS

1. Petroff, C.A., **Bina, T.F.** & Hutchison, G.R. Highly Tunable Molecularly Doped Flexible Poly(dimethylsiloxane) Foam Piezoelectric Energy Harvesters. ACS Applied Energy Materials (2019), 2, 9, 6484-6489
2. Kunkle, D.E., **Bina, T.F.**, Bina, X.R. & Bina, J.E. *Vibrio cholerae* OmpR Represses the ToxR Regulon in Response to Membrane Intercalating Agents that are Prevalent in the Human Gastrointestinal Tract. Infection and Immunity (2019) DOI: 10.1128/IAI.00912-19

3. Bina, X.R., Wong, E.A., **Bina, T.F.** & Bina, J.E. Construction of a Tetracycline Inducible Expression Vector and Characterization of its Use in *Vibrio cholerae*. Plasmid (2014) 76, 87–94

PRESENTATIONS

- "PEEK: An Innovative Material for Musculoskeletal Implants"
    - MATSE 492: Materials Engineering Methodology and Design
- "Development of the Microscopy Laboratory Information Management System"
    - SURF Colloquium at NIST, Gaithersburg, MD

TEACHING EXPERIENCE

Penn State University Undergraduate Teaching Assistant

- MATSE403 / BME443: Biomedical Materials — Aug 2019 – Dec 2019

HONORS & AWARDS

Dean's List	2016-2020
Penn State Academic Excellence Award	2016-2020
Schreyer Honors Scholar	2016-2020
William and Estelle Turney Scholarship	2018-2020
Lola G. Duff and William H. Duff Merit Scholarship	2016-2020
Richard M. Wardrop, Jr. Honor Scholars Scholarship	2018-2019
John G. Miller Scholarship	2017-2018
Helen and Van Leichliter Scholarship	2017
R&M Peiffer Scholarship in Materials Science & Engineering	2017
Matthew J. Wilson Honors Scholarship	2016-2017

ACTIVITIES

Penn State Materials Advantage	2016-2020
Penn State Club Ultimate Frisbee	2016-2018
PSU Magic: The Gathering Club *(Webmaster)*	2016-2020
Super Smash Brothers Club at PSU	2018-2020