Identifying Two-Dimensional Materials with Machine Learning

Shannon Gray
Department of Physics and Astronomy, Colby College
University of Michigan Physics REU and
PI: Liuyan Zhao
(Dated: August 7, 2020)

The goal of my project this summer was to create a machine learning program that could classify images with or without 2-Dimensional flakes. In the future, this program could be expanded to identifying the material of the flake and the number of layers of a sample (such as mono- or bi-layer). Eventually, the program could be used directly in the process of collecting images of flakes.

I. INTRODUCTION

Professor Liuyan Zhao's research group studies condensed matter with a focus on discovering and understanding electronic phases in quantum materials. One example of such a material is graphene, which is the two-dimensional counterpart to graphite and a quasi-metal, meaning it has properties of a semi-conducting metal. As a result, it has promising uses in the development of quantum technologies. To identify these materials, which are on the scale of one-atomic thickness, optical spectroscopy is used. However, the process of finding these two-dimensional flakes is extremely time intensive and often relies on the experience of the user, with various opportunities for human error. Because of this, several groups have begun to utilize machine learning in this process [1–3]. My goal this summer was to develop a machine learning program that can identify images with flakes as a first step in automating this process.

In contrast to coding all of the rules for how to identify a flake by hand, machine learning can create its own rules for classification and automatically learn them efficiently. Generally, machine learning can be divided into two groups: supervised learning, which develops a predictive model based on both input and output data, and unsupervised learning, which groups and interprets data based only on input data. For this project, I used supervised machine learning for classification to create a program that predicts inputs to be within given classes. Supervised machine learning utilizes a train-test split function, which takes 75% of the input data to use to create the rules for classification, and then uses the other 25% of the data to see how well those rules generalize to unseen data.

The classifiers utilized here are decision trees, random forests, and gradient boosted decision trees. Decision trees classify inputs, or data instances, by creating decision nodes based on given features, which in turn create branches where this process can be repeated until the tree reaches a given depth. Decision tress are easy to visualize and often easier to understand, however they also often overfit, meaning that the rules created do not generalize well to new data. Random forests and gradient boosted decision trees are both ensembles of decision trees, which make them better at generalizing to new data than single decision trees. Random forests combine all of their decision trees' predictions at the end of the process using the majority, and boosted decision trees average the results throughout the process.

II. METHODS

For our input data, each instance was an image taken in the Zhao lab. I was provided 50x and 100x magnification images of the samples from members of the lab: Hongchao Xie, Eunice Paik, Jason Horng, Jacob Waelder, Kuan-Wen Chen, and Long Zhang. I was not able to obtain images without flakes directly, and because I needed non-flake data in order to have a non-trivial classification program, I used the images I was given to get images without flakes. Multiple methods were used in getting non-flake images to be able to determine if one or the other could be causing data leakage within the program. The first method was taking screenshots of the parts of the 50x magnification images without the flake. The other method was indexing the array of the 50x magnification images to select a region of the array without the flake. These two methods could result in varrying pixel sizes or image quality, so having both is helpful for comparison. To make the data more realistic to the future application, there were more images without flakes than images with flakes. Specifically, there were 145 images without flakes and 90 images with flakes.

In order to input the images into the program, I first needed to extract features from them. For an image, features can be information such as the RGB (Red, Green, Blue) values, number of pixels of a certain color, optical contrast, and more. To extract features such as the RGB color of the flake, the difference in RGB colors between the flake color and the background, and the approximate number of pixels of the flake, I used a function called color quantization, shown in Fig. 1. Color quantization utilizes k-means, a method of vector quantization that partitions each color in the image into the nearest mean color of a given number of colors. With the information provided from

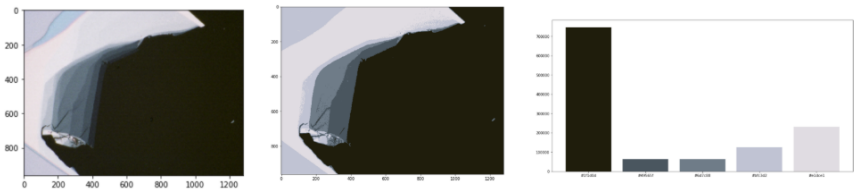

FIG. 1. This shows a progression of the original image with a sample flake, then the image after color quantization has been applied to simplify the image to five colors. It also shows a bar graph, which shows the five colors in the image, and the number of pixels of each color is represented by the height of each bar. The color quantization function returns the colors from darkest to lightest, and because the flake is the thinnest and most transparent, it will be the color that is closest to the background color.

Classifier	Training	Test
Decision Tree	80%	68%
Boosted Decision Tree	84%	71%
Random Forest	87%	79%

TABLE I. Accuracy scores per classifier found after running the program for flake classification on the data with the screenshotted no flake images. These results are from using the features of the flake color and color difference between the flake and the background.

Classifier	Training	Test
Decision Tree	90%	81%
Boosted Decision Tree	94%	87%
Random Forest	85%	85%

TABLE II. Accuracy scores per classifier found after running the program for flake classification on the data with the images with no flakes that came from indexing the array of the original image. These results are from using the features of the flake color and color difference between the flake and the background.

color quantization, the features of the color of the flake, color difference between the flake and the background, and number of pixels of the flake could be calculated or accessed directly.

III. RESULTS

After running the classifiers with the cropped versions of the images with no flakes, and the features of color difference between the flake and the background in RGB as well as the flake color in RGB, the accuracy scores ranged from 81% to 87%, shown in Table I. The most accurate classifier was the random forest classifier with a 79% accuracy score on the test data. However an 87% accuracy score on the training data implies that this classifier may be overfitting. Furthermore, the classifiers all had a relatively high number of false negatives, with the random forest classifying 11 images with a flake as not having a flake. To try to improve upon our accuracy and lower the number of falsely classified images, I tried running our algorithm with the alternative method of obtaining images without flakes.

After running the classifiers on the data with the indexed images, the accuracy scores improved across all the classifiers. As shown in Table II, the training and test scores on the random forest classifier are the same, which implies that this classifier is not overfitting. Furthermore, it only predicted 5 false negatives instead of the 11 with the random forest on the other data. Although this classifier appears to be performing better than before, it still did not have as high of accuracy scores or low of false classifications as desired. To address this, I started to explore post-processing methods to aid in the analysis.

This post-processing method is a mask, shown in Fig. 2, that can be applied to any of the images. The function I made for this generalized mask utilizes color quantization, which was also used in extracting features of the image. Using color quantization to obtain the color of the flake, the mask makes all of the pixels that are the color of the

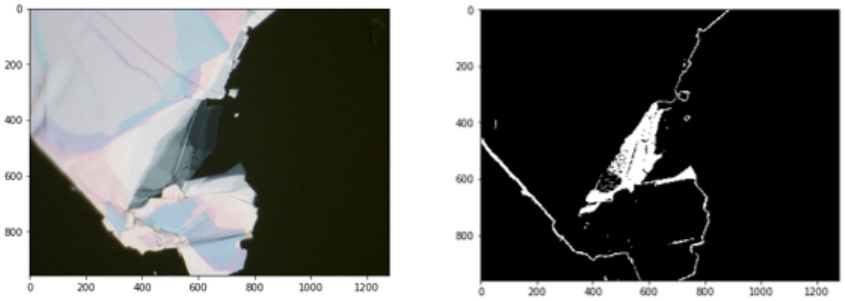

FIG. 2. A comparison of an original image that contains a flake and its corresponding mask.

flake white while making all the other pixels in the image black. This mask can be helpdul in understanding why the program is classifying images falsely by directly seeing why it is classifying images with a flake as not having a flake, and vice versa. However, as shown in Fig. 2, the mask is still including the outline of other materials of the image that are not the flake, so improving this function will likely be necessary before implementing it into post-processing.

IV. DISCUSSION

In comparision to groups who have done similar projects with machine learning classification of 2-Dimensional materials, this project has achieved similar accuracy scores. Other groups used deep learning and neural networks in their programs, which are more complex algorithms that often require more data or computing power, and reported accuracy scores in the range of 70-90% [2, 3]. The scores achieved by this project are within this range. The highest accuracies reported were done with classifying graphene layers, with an accuracy of 96.78%[4]. With more images of flakes, and original images without flakes, this program has potential for reaching higher accuracies. The accuracy scores could also be improved through new features used for classification being extracted from the image and added.

Other than improving accuacy scores and minimizing the number of false classifications, this project can be expanded upon in a number of ways. First, the program can be applied to classifying the type of material. This would expand it to multiclass classification, but can be implemented by simply adding the features and targets desired. I started to implement this material classification this summber, but there was not enough data of each material to see significant results. Similarly, it could be expanded to predicting whether the flake is a monolayer or bilayer. These will both be necessary steps in reaching the broader goal of automating the process of collecting samples of flakes.

[1] Li, Yuhao, Kong, Yangyang, Peng, Jinlin, Yu, Chuanbin, Li, Zhi, Li, Penghui, Liu, Yunya, Gao, Cun-Fa, Wu, Rong. (2019). Rapid identification of two-dimensional materials via machine learning assisted optic microscopy. 10.1016/j.jmat.2019.03.003.

[2] Saito, Y., Shin, K., Terayama, K. et al. Deep-learning-based quality filtering of mechanically exfoliated 2D crystals. npj Comput Mater 5, 124 (2019).

[3] Han, Bingnan et al. "Deep-Learning-Enabled Fast Optical Identification and Characterization of 2D Materials." Advanced Materials 32.29 (2020): 2000953. Crossref. Web.

[4] Lin, Xiaoyang & Si, Zhizhong & Fu, Wenzhi & Yang, Jianlei & Guo, Side & Yuan, Cao & Zhang, Jin & Wang, Xinhe & Liu, Peng & Jiang, Kaili & ZHAO, Weisheng. (2018). Intelligent identification of two-dimensional nanostructures by machine-learning optical microscopy. Nano Research. 10.1007/s12274-018-2155-0.

V. ACKNOWLEDGMENTS

I would like to thank Professor Liuyan Zhao, as well as head graduate student in the Zhao group Elizabeth Drueke for mentoring me throughout this project and providing their insights and guidance. Furthermore, I would like to thank Laura Zichi for her contributions to this machine learning program and collaboration. This project was supported by the NSF as a part of the University of Michigan Physics REU program. I would also like to thank Professor Myron Campbell and Professor James Lui for organizing this program under the unique circumstances this summer, and for all the effort they put into making it an exceptional exprience.