Identification of Monolayer Material

Jun Ho Son (json2@stanford.edu), Yue Yu
(johnyu10@stanford.edu), Sze Cheung Lau (szlau@stanford.edu)

Category: Physical Sciences

I. INTRODUCTION

In condensed matter physics, two-dimensional material has been particularly rich and interesting playgrounds for exciting and exotic optical and electronic properties. A common way to obtain two-dimensional material in our three-dimensional world is to start from layered materials such as graphene, transition metal dichalcogenides (TMD) and hexagonal boron nitride. In such materials, atoms within a same layer form robust chemical bonds with each other, while much weaker inter-layer van der Waals force hold the materials into the layered structure. It turns out that van der Waals force is so weak that one can exfoliate a single two-dimensional layer from aforementioned multi-layered materials with a deceptively simple mechanical procedure, so-called mechanical exfoliation. In this process, adhesive tape loaded with thin crystals of our layered materials is pressed on a substrate and then peeled off; some portion of the crystal is transferred onto the substrate. However, one caveat of this method is that the transferred crystal has a variety of shapes and thickness; especially, while some of the samples will surely be mono-layer, there is no guarantee in general that the transferred crystal is consisted of a single layer. In the current laboratory setting, the only way to select out mono-layer samples is to manually inspect all the transferred samples with optical microscopes. This procedure is time-consuming and labor-intensive. In this project, we would like to explore possibility of utilizing computers do the task. In particular, we aim to write a program that can identify mono-layer samples from photos of the samples taken in optical microscopes using tools from machine learning.

All images are labeled as T/F manually by whether they contain any monolayer material as shown in fig.1. We used convolutional neutral network to train the binary classification model, and make predictions on the testing set. However, to get a first impression of the ability of machine learning techniques, we started with logistic regression as a benchmark test. Due to limited number of images, we applied various data augmentation methods on the training sets.

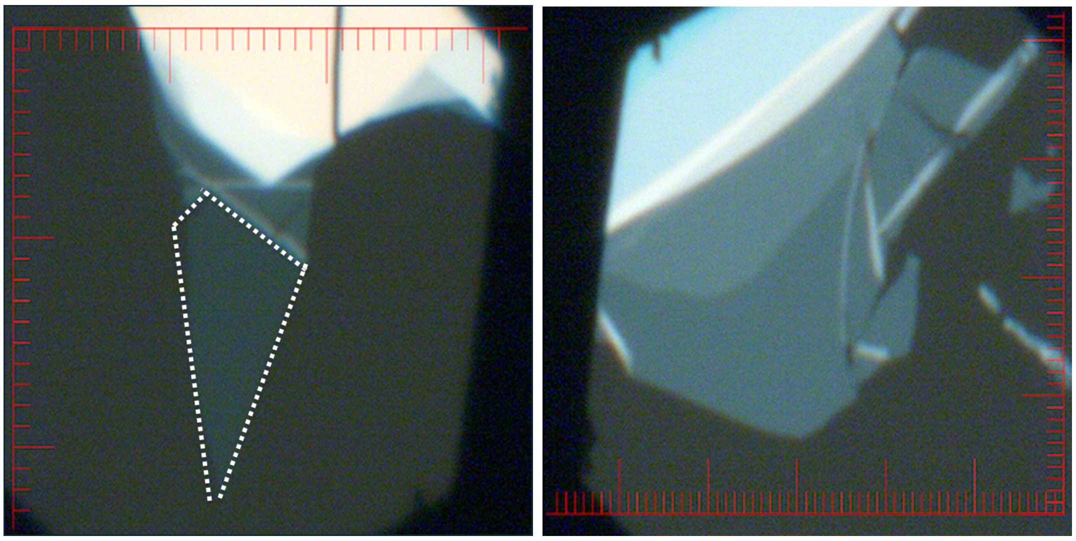

FIG. 1: Examples of data we collected. Left: A true photo containing monolayer material (area enclosed by dotted line); Right: A false photo without monolayer material.

II. RELATED WORK

There is a work[1] which applied SVM within images to separate one image into different regions. Presumably, each region corresponds to a particular layer count. A similar work[2] applied mixed gaussian model to distinguish regions within an image. Another work[3] applied k-mean clustering to do the same task.

If we have knowledge of RGB value as a function of layer count, the above methods could provide an one to one correspondence between regions and number of layers. However, this knowledge is based on experimental conditions, and may differ from one microscope to another. Furthermore, the above works essentially treated each pixel independently, and separate pixels into groups. This neglects correlation between pixels, which could be important in certain circumstances. Our work, however, focuses on whether monolayer material exists in a given image. Each image is predicted as true or false, and we keep information about correlations between pixels.

III. DATASET AND FEATURES

All images were taken by ourselves from laboratory. Due to time limit, we only obtained 265 images. 15 and 31 images were taken out for testing set and validation set. Remaining images were used in training set. Each image contains 375 times 375 pixels.

Before the training process, we applied various data augmentation methods on training set, including multiples of 90° rotation and mirror reflection. The size of the training set was expanded to 8 times. During each epoch in the training process of convolutional neutral network, we further applied random operations on each image. Those operations include random small angle ($\theta < 30°$) rotation, shift, zoom, and shear. Due to their physical nature, monolayer materials could appear in any shape and size. The transformed images remain relevant to our problem and inherited the labels from the original images.

IV. METHODS

In this section, we briefly introduce the structure of convolutional neural network. As illustrated in Fig.2, the network consists of 3 layers of convolution operations, 1 fully connected layer and 1 output layer. In convolution layer l, given an filter w and a feature map o^{l-1}, following operations will be performed to produce a new feature map o^l.

$$\begin{aligned} x^l_{i,j} &= \sum_m^{l_m} \sum_n^{l_n} w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l_{i,j} \\ o^l &= f(x^l) \end{aligned} \tag{1}$$

Filter or kernel w^l with size (l_m, l_n) connects neurons from layer $l-1$ to l. b^l is the bias term. Output o^l is calculated from activations x^l by activation function f. In each layer, the number of feature maps multiplies with the number of filters. The size of each feature map will be reduced, from (l_x, l_y) to $(l_x - l_m + 1, l_y - l_n + 1)$. In the first convolution layer, 16 filters with the size $(11, 11)$ were used. Since we were using the filters with large sizes, we also put strides to be (3,3) in the first layer, which denotes that the position in which filter was applied moves by 3 in either direction instead of 1 in standard CNN. The second convolution layer and the third convolution layer has 24 filters of the size $(5, 5)$ and 36 filters of the size $(3, 3)$ respectively. After each convolution, we performed max pooling to decrease the size of the "image" to one half in each dimension. Activation functions were taken to be ReLU functions everywhere for the convolutional layers.

After 3 layers of convolution, we connect all feature maps to the fully connected layer with 256 neurons, with ReLU activation function. The dropout regularization[4] is used on fully connected layers to prevent overtraining. As the final step, we feed the output of 256 neurons into one output neuron with sigmoid activation function for the binary classification.

We used Tensorflow[5] package to implement the above CNN.

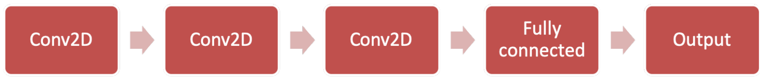

FIG. 2: Illustration of convolutional neural network. The network consists of 3 layers of convolution operations, 1 fully connected layer and 1 output layer.

V. BENCHMARK TEST-LOGISTIC REGRESSION

As a benchmark test, we first tried logistic regression. As expected, logistic regression behaved poorly on the prediction. It converged within a few epochs and predicted all images to be false on both the validation set and the testing set. One of the possible reasons is that our training set is still too small for logistic regression.

VI. RESULTS AND CONCLUSION

We tried various different parameters to get the best accuracy on validation set. The codes and parameters can be found in (https://github.com/szlcodes/cs229/blob/master/letslearnbetter2.py). We tried using 1 to 5 convolution layers. For 1 to 3 convolution layers, the results did not differ too much. However, for 4 and 5 convolution layer, the networks had a similar behavior to simple logistic regression. Eventually we chose to optimize parameters for 3 convolution layers. Those parameters includes number of filters, size of filters, learning rates, decay rates, and momentum, which can be found in the above link.

The loss and accuracy as a function of number of epochs are plotted in Fig.3, for training set and validation set. The loss function of training set had a decreasing trend with fluctuation, since random operations (small angle rotation for example) were performed on images in every epoch. All images in training set were used in each epoch. We ran the codes for 60 epochs, and it is likely that the loss and accuracy on the validation set had converged.

The fluctuation in the figures is relatively big, since our dataset was very small. We tried different validation set (same parameters in CNN), and there was no qualitative difference in the results.

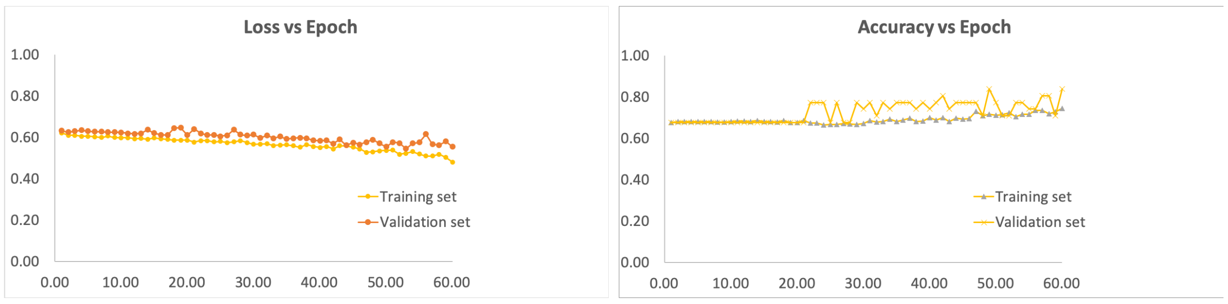

FIG. 3: Loss and accuracy as a function of number of epochs, for training set and validation set.

80% accuracy was achieved on the testing set (67% true negative, 13% true positive, 20% false negative, and 0% false positive). It should be noted that the most serious problem in our result is that our dataset is too small. It is crucial for us to prepare a large dataset in the future.

VII. CONCLUSION

In this work, we applied convolutional neural network on identification of monolayer materials. Convolutional neural network performed better than the benchmark test of logistic regression, with an accuracy of 80% on testing set. We applied various data augmentation methods on the training set to expand dataset. However, the dataset was still too small, and imposed a limit on our final result.

All the images were taken by ourselves, and we plan to gather more images in the future. At the same time, we could vary structures and parameters in the neural network, and further improve the performance of the program.

VIII. CONTRIBUTION

Lau and Yu collected data and did data augmentation. Son drafted the training code. We all tested and modified the code, and wrote up the final report.

[1] X. Lin, Z. Si, W. Fu, J. Yang, S. Guo, C. Yuan, J. Zhang, X. Wang, P. Liu, K. Jiang, et al. (2018).

[2] S. Masubuchi and T. Machida, npj 2D Materials and Applications **3**, 4 (2019).

[3] Y. Li, Y. Kong, J. Peng, C. Yu, Z. Li, P. Li, Y. Liu, C.-F. Gao, and R. Wu, Journal of Materiomics **5**, 413 (2019), ISSN 2352-8478, URL `http://www.sciencedirect.com/science/article/pii/S2352847819300048`.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Journal of Machine Learning Research **15**, 1929 (2014), URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015), software available from tensorflow.org, URL `https://www.tensorflow.org/`.