Classification of single-crystal “diffuse” images

Zhen Su (zhensu@stanford.edu)

I. Abstract

In the x-ray diffraction experiment, we collected about 8 million diffraction images in total, where less then 0.3% (20 thousand) were “good” images that could be used for further scientific study. Even among those “good” images, about 95% was “non-diffuse” image (1b, 1d, 1f), and only 5% (1 thousand) was “diffuse” image (1a, 1c, 1e) that we decided to extract out using machine learning methods in this report. To achieve this goal, we built two models — the diffusion map method and the convolutional neural network (CNN) based on the famous VGG16 pipeline [5][6]. For training and testing, we only used a small portion of those 20 thousand “good” images due to the difficulty to label all of them within limited time. After several failed tried, we finally got a 91.3% classification accuracy for the diffusion map method and a 93.6% accuracy for the CNN model.

II. Introduction

X-ray diffraction is one of the most important techniques to study the structure of macromolecules. In the real experiment, x-ray beam hits the macromolecule and generates the diffraction image 1, each image is a spherical section cut (from a random direction) of the 3D Fourier transformation of the measured object, as shown in figure 2. To make an analogy, if we cut a bread thousands of times and save all the section-cut images, later on we can use all these section-cut images to reconstruct the whole bread. For the same reason, we can use all the collected diffraction images to reconstruct the structure of a macromolecule. The x-ray pulse repetition rate in Linac Coherent Light Source at SLAC national laboratory is 120HZ, which means that we can collect 120 images per second. Even so, more than 99.7% are useless images. Among those 0.3% “good” images, only 5% have “diffuse” features (“good” images = “diffuse” images + “non-diffuse” images).

Why we want to find “diffuse” images? Some recent papers [1] [2] point out that “diffuse” feature is a window to study the molecular dynamics. Currently, finding “diffuse” images from all the “good” images highly depends on the labor force and some empirical metrics [1]. It will be a significant progress if we can find a practical and efficient method to search those “diffuse” images automatically.

What is the difference between “diffuse” and “non-diffuse” images? As we mentioned before, x-ray diffraction technique measures the Fourier transformation of macromolecules, as shown in figure 2. If our crystal is perfectly periodic, then the Fourier transformation will also be regular spots. However, if our crystal is not perfect with some defects or lattice displacement, then the Fourier transformation will be regular spots plus small “diffuse” features. This is the main difference between “diffuse” and “no-diffuse” images. Three sample images with many “diffuse” features are shown in the left side of figure 1. Even for them, “diffuse” signal is only $\frac{1}{10} \sim \frac{1}{5}$ of the circular background, let alone for other images with tiny “diffuse” features. It’s a challenging work to classify them 100% accurately even with human eyes. In this report, we will introduce two machine learning models that can automatically separate the “diffuse” and “non-diffuse” images, we will implement the diffusion map model and a CNN model based on VGG16 pipeline [5] [6] which finally achieves a high classification accuracy.

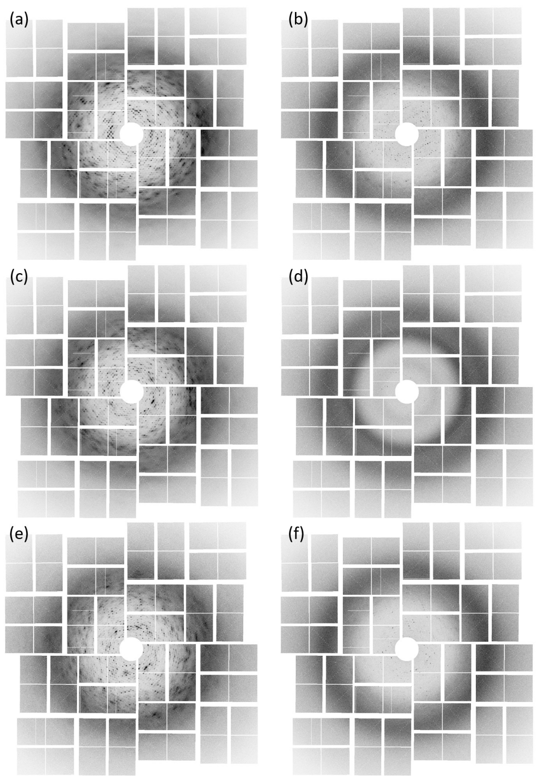

Fig. 1. Six sample images from our experiment. (a),(c),(e) in the left side are “diffuse” images; while (b),(d),(f) in the right side are “non-diffuse” images. Image (a),(c),(e) are specially selected with the most “diffuse” features

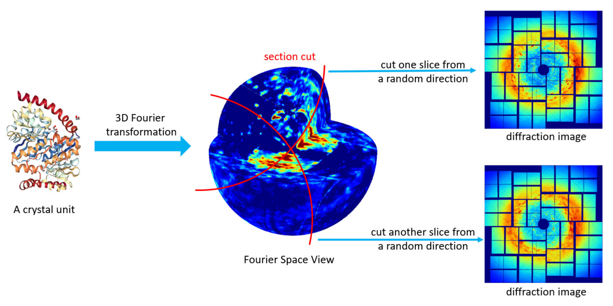
section cut
cut one slice from a random direction
3D Fourier transformation
diffraction image
A crystal unit
cut another slice from a random direction
Fourier Space View
diffraction image

Fig. 2. This figure schematically shows how x-ray diffraction images are generated. Each image is a section cut of the Fourier Space View from a random direction. (Note: this is just a schematic figure, not all the images are from the same experiment)

III. Difficulties

A. Huge background and detached detector

The circular background, which comes from the solvent diffraction, is much larger than the "diffuse" feature. Removing the isotropic background of each image works well to make "diffuse" feature stand out. Even so, problems still exist in calculating the isotropic background, such as the detached detector shape (figure 1). Our detector is not a single panel, but an ensemble of 32 detached pieces (figure 1). Considering those blank parts strongly slows down the speed in calculation.

B. Distribution of "diffuse" features

As we mentioned above, each image is a section cut of a 3D object from a random direction (figure 2). Therefore, although "diffuse" features are regularly distributed in the 3D space, they become randomly distributed in the 2D image due to the random section-cut direction. In this case, we don't know whether $L2$ norm is still a reasonable metrics in calculating the similarity matrix for the diffusion map method. More seriously, all traditional distance metrics like cosine distance, correlation coefficient may not work any more.

C. Various background

As we can see in figure 1, the background in each image is nearly isotropic and identical, but sometimes they can be very different in other experiments. Therefore, it's still uncertain whether the trained model works for other experiments. If it can't detect "diffuse" image from another experiment, we have to label thousands of images every time we do an experiment. It's important to standardize the whole process to build a model with high generality.

IV. Dataset

All the diffraction images were collected in Linac Coherent Light Source at SLAC National Laboratory in a recent experiment "cxiclp9515" for studying the DNA repair reaction. After the first step selection, only 20 thousand "good" images (including "diffuse" and "non-diffuse" images) were left. Each image is a $\mathbb{R}^{1734} \times \mathbb{R}^{1731}$ array. For the diffusion map model, we used 1000 labeled images for classification and testing. For the CNN model, we used 500 labeled images for training and another 500 labeled images for testing. Sample images are shown in figure 1.

V. Related work

A recent paper [1] has realized the importance of classifying "diffuse" and "non-diffuse" images, but they used an empirical method to treat each image as a combination of two parts: the isotropic background and the anisotropic feature. Since the "diffuse" feature is mainly anisotropic, so they expect "diffuse" images to show relatively high anisotropic intensity than "non-diffuse" images. We followed the same procedure and got the result in figure 3, where y axis stands for the overall anisotropic intensity (x axis is the overall peak intensity but it's not related to our project). Among the top 500 images above the red line with the largest y coordinate (highest anisotropic intensity), about 74% are "diffuse" images and 26% are "non-diffuse" images. Even though the accuracy was not high, it helped us a lot in the first-step selection.

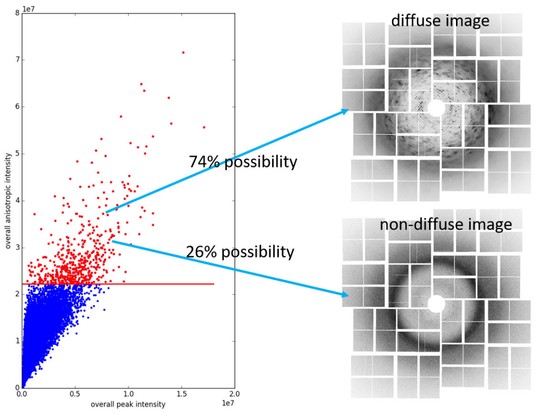

Fig. 3. Classification result by the overall anisotropic intensity. In the left panel, x axis is the overall peak intensity, and y axis is the overall anisotropic intensity. Top 500 images with the highest anisotropic intensity are labeled by red color. Among these 500 red spots, 74% are "diffuse" images and 26% are "non-diffuse". This is a empirical classification method [1] that simply requires some preprocess although the accuracy is very low

VI. Model One: Diffusion Map

A. Method introduction

Diffusion Map is a method to embed high dimensional data into a low dimensional Euclidean space. Assuming we have N images, then we can calculate their pair-wise similarity and save into a $N \times N$ similarity matrix S, followed by several normalization steps. The resulting normalized matrix M has right eigenvalues and eigenvectors (λ_l, ψ_l). Then, the diffusion distance D between two images $(x^{(i)}, x^{(j)})$ is:

$$D(x^{(i)}, x^{(j)}) = \sum_l \lambda_l^2 \left(\psi_l(x^{(i)}) - \psi_l(x^{(j)})\right)^2$$

Inspecting the form of this equation, we know that each image $x^{(i)}$ can be represented in the new set of coordinates as:

$$\Psi(x^{(i)}) = (\lambda_1\psi_1(x^{(i)}), \lambda_2\psi_2(x^{(i)}), \ldots)$$
$$\Rightarrow D(x^{(i)}, x^{(j)}) = ||\Psi(x^{(i)}) - \Psi(x^{(j)})||^2$$

so that the Euclidean distance in the diffusion coordinates approximates the diffusion distance. The only thing is to calculate the similarity matrix S.

If we denote "diffuse" images as class A, and "non-diffuse" images as class B, the presupposition for diffusion map method to work is: two images in class A (or class B) share relatively large similarity; one image in class A and another image in class B share relatively small similarity.

B. Failed tries

From each raw image 1, we removed the isotropic background, then set the mean value to 0 and variance to 1, the resulting image $x^{(i)} \in \mathbb{R}^{1734} \times \mathbb{R}^{1731}$ was the input vector for diffusion map method. In the first try, we represented the similarity metrics using euclidean distance $S_{ij} = e^{-||x^{(i)}-x^{(j)}||^2/\sigma}$. In the second try, the similarity was $S_{ij} = e^{cc^*(x^{(i)},x^{(j)})}$ where cc^* is the correlation coefficient. However, both metrics failed to classify "diffuse" and "non-diffuse" images. Reason of failure: Recall the second challenge in the previous chapter "Challenges", the "diffuse" feature in each image is randomly distributed. Therefore, two images from the same class didn't share a high similarity, which broke the presupposition of diffusion map method.

C. *New similarity metrics*

Failed tries led us to come up with a new similarity metrics. Since each image was a section cut of a 3D object, then the common line of two section cuts should share the same distribution which means a high correlation. Therefore, we converted each image back to the 3D object and find their common lines as shown in figure 4. This is the reverse process of figure 2. (How to merge a 2D image back to 3D is shown here [4] [3], it was implemented by my own code). After we embed the 2D image to 3D Fourier Space, the input data is no longer a 2D array but a 3D vector $x^{(i)} \in \mathbb{R}^{200} \times \mathbb{R}^{200} \times \mathbb{R}^{200}$. For every two images, their similarity metrics was the correlation coefficient in the common line: $S_{ij} = e^{cc^*(x^{(i)}_{common}, x^{(j)}_{common})}$.

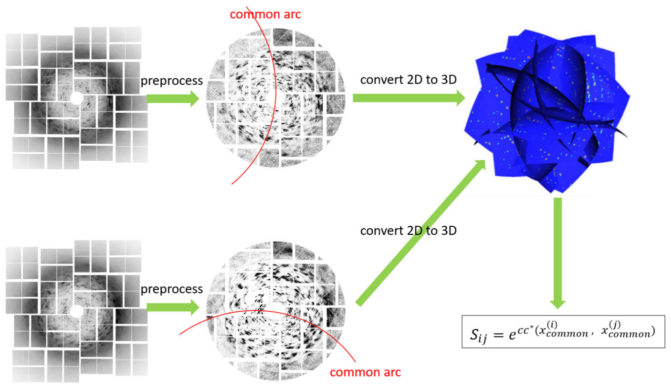

Fig. 4. The schematic illustration of the new similarity metrics, this is a reverse process of figure 2. In figure 2, we cut slices from the 3D Fourier Space, while now we embed the section cuts back to 3D Fourier Space. The new similarity metrics is the correlation coefficient of the common line.

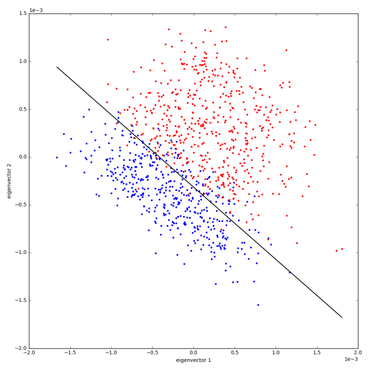

Fig. 5. The clustering result with diffusion map method, where the red spots stand for "diffuse" images and blue spots stand for "non-diffuse" images

D. *Result and discussion*

Using the similarity matrix S calculated with the new metrics, we implemented the diffusion map algorithm with 1000 labeled images. A nice clustering result was found using the 4th and 5th eigenvectors, which is shown in figure 6. The classification accuracy is about 91.3% using logistic regression to the clustering data. The biggest drawback for this model is the calculation speed, especially when we convert a 2D image to the 3D Fourier space. To calculate the similarity matrix of 1000 images, it took me about 2 hours with 84 parallel cores.

VII. Model Two: Convolutioal Neural Network

A. *Method introduction*

Building a CNN model was motivated by its high efficiency compared to the diffusion map method. What we want is a real-time classifier that can synchronize with the experiment pipeline.

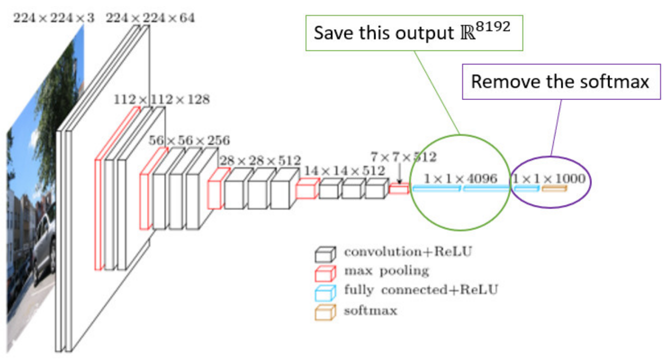

Fig. 6. The architecture of VGG16 [7] and our modifications

Even though millions of images were collected in one experiment, only about one thousand have "diffuse" features. As a consequence, we actually didn't get enough data to train a large neural network model, so instead we used a well-trained neural network model VGG16, whose 16-layer structure is shown in figure 6. In short, what we did was to feed VGG16 with our images, and then took the VGG16 output as the input features for our own logistic regression classifier. Implementation of VGG16 pipeline is shown in [6]

Input for VGG16: After several simple preprocesses, each raw image was downsampled to 224×224 dimension and then converted to the RGB format, as shown in figure 7, which ends up with the input vector for VGG16: $x^{(i)} \in \mathbb{R}^{224} \times \mathbb{R}^{224} \times \mathbb{R}^{3}$.

Output of VGG16: VGG16 converts each RBG image to two $\mathbb{R}^{4096}$ vectors, as shown in figure 6, followed by a 1000-class softmax classifier. We remove the softmax classifier and save the output features in last two layers: $y^{(i)}_{VGG} \in \mathbb{R}^{8192}$ as the input vector for the logistic regression classifier, as shown in figure 7.

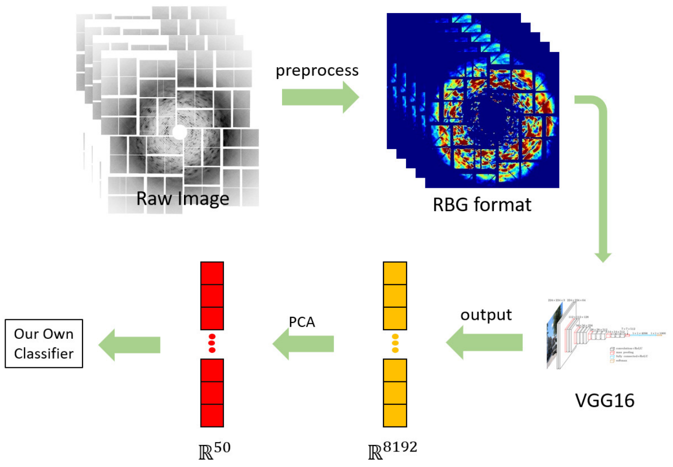

Fig. 7. The flow chart of our CNN pipeline, we save the output $\mathbb{R}^{8192}$ of VGG16 as the input for our own classifier, the principle components analysis (PCA) is also applied in between

B. Principle Component Analysis

VGG16 output is a 8192-dimension vector but we only have thousands of images, which is comparable to or smaller then the dimension of features (8192). In this case, over-fitting is very possible to appear. Therefore, a principle component analysis (PCA) step is applied between VGG16 and the Logistic Regression classifier. As shown in figure 7, we use the PCA algorithm to extract the dominate 50 features from the 8192-dimension output. The resulting $x^{(i)} \in \mathbb{R}^{50}$ is the input vector for the Logistic Regression.

C. Logistic regression

We used 500 dataset $(x^{(i)} \in \mathbb{R}^{50}, y^{(i)} \in \mathbb{R})$ to train a simple logistic regression model. After the training process converges as shown in figure 8, we tested it on another 500 dataset and the result is shown in table I.

$$LL\theta = \sum_i y^{(i)}\log[\sigma(\theta^T x^{(i)})] + 1 - y^{(i)}\log[1 - \sigma(\theta^T x^{(i)})] + \lambda||\theta||^2$$

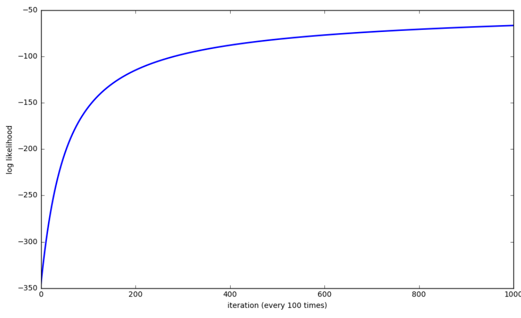

Fig. 8. The log likelihood curve during the training process, x axis is the iteration (every 100 times is one iteration)

D. Result and discussion

The testing results in table I were separated by the precision rate and recall rate, because we have bias on the "diffuse" image rather than the "non-diffuse" image that is not so important to us. What we want is a high precision rate for "diffuse" image (95.3%), which means that among all the images predicted as "diffuse" image, most of them should be exactly "diffuse" images.

TABLE I
TESTING ACCURACY

Classes	Precision rate	Recall rate
"diffuse" images	95.3%	91.4%
"non-diffuse" images	92.1%	95.7%
overall	93.6%	93.6 %

VIII. CONCLUSION

In summary, we have implemented the diffusion map method and CNN model for our image classification project. The testing accuracy for each model is high enough for further investigation. However, the diffusion map method isn't fast enough as a real-time predictor due to the tricky dimension conversion step from a 2D array to the 3D Fourier Space. The CNN model has a relatively high precision rate and high efficiency, which is a good candidate for further tunning and modification. Due to the small number of "diffuse" images, we can only train a small personalized CNN model if we plan to, so relying on VGG16 is still a reasonable choice by now.

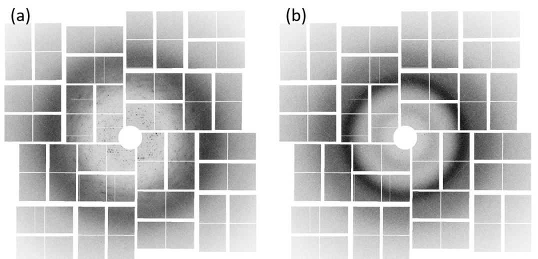

Fig. 9. Two examples of wrong prediction using CNN model. (a) is "diffuse" image but predicted as "non-diffuse" image, probably due to the weak "diffuse" feature. On the contrary, (b) is "non-diffuse" image but predicted as "diffuse" feature, probably due to the strange circular background

IX. FUTURE WORK

First of all, we will train and test the CNN model with larger dataset, probably using all the 20 thousand images. Currently, we haven't found another suitable experiment dataset that can test our model because the "diffuse" image is very rare. Validation from another experiment in the future is indispensable to test its generality. Secondly, as shown in figure 9, some images are predicted incorrectly with CNN model due to the weak signal or strong background. We will definitely try more methods to reduce the background and make "diffuse" feature stand out. What's more, we will start to build a real-time CNN training-prediction-labeling pipeline: we plan to first label very few images and feed the CNN model, then the CNN model will make predictions to the rest of our dataset quickly and pop up some highly recommended possible "diffuse" images for us to label. This strategy greatly enhances the efficiency and reduces the number of images to label. High efficiency and fewer labeling is our final goal.

X. CONTRIBUTIONS

Dr. Chun Hong Yoon and Haoyuan Li provide some suggestions on running VGG16. Diffraction images were collected at SLAC National Laboratory by LP95 team in a recent experiment "cxiclp9515" for studying the DNA repair reaction led by Thomas J Lane (tjlane@slac.stanford.edu).

REFERENCES

[1] Ayyer, Kartik, et al. "Macromolecular diffractive imaging using imperfect crystals." Nature 530.7589 (2016): 202-206.
[2] Wall, Michael E., et al. "Diffuse X-ray scattering to model protein motions." Structure 22.2 (2014): 182-184.
[3] Yefanov, Oleksandr, et al. "Mapping the continuous reciprocal space intensity distribution of X-ray serial crystallography." Phil. Trans. R. Soc. B 369.1647 (2014): 20130333.
[4] Ayyer, Kartik, et al. "Dragonfly: an implementation of the expandmaximizecompress algorithm for single-particle imaging." Journal of applied crystallography 49.4 (2016): 1320-1335.
[5] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[6] https://github.com/machrisaa/tensorflow-vgg
[7] http://www.cs.toronto.edu/ frossard/post/vgg16