Improving machine-learning classification accuracy of x-ray scattering images

Ronald Lashley [e], Nicole Meister [a, b], Ziqiao Guan [b], Bo Sun [f], and Dantong Yu [c, d]

[a] Stony Brook University, [b] Centennial High School,

[c] Brookhaven National Laboratory, [d] New Jersey Institute of Technology,

[e] Lincoln University (PA), [f] Rowan University

Abstract

In technology, image recognition software uses various methods to extract information from an image. X-ray scattering images contain visual features such as rings, spots, and halos. Using machine learning, we are proposing to increase the accuracy of image classification by increasing the amount of data. Due to the real dataset being hard to get and/ or not being ready, we use synthetic x-ray scattering images that are generated using physics for training and testing purposes. The synthetic images are as close to possible to the real dataset making it efficient enough to use. Previously, synthetic images that have been generated have a size of 256 x 256 pixels. Now, we have increased the resolution of the synthetic images by generating synthetic images with a size of 1,000 x 1,000 pixels. Increasing the resolution of the image provides us with more information of the attributes, increasing our ability to recognize and classify the images by tagging them. Here at the Computer Science Initiative, we have changed a few parameters in the previous python code that was used for the original dataset size to increase the size of the generated images to 1,000 x 1,000 pixels. We used the generated synthetic images to train and test our machine learning system that classifies and tags the x-ray scattering images. The real dataset that needs to be classified and tagged are from the National Synchrotron Light Source II (NSLS-II) at Brookhaven National Laboratory (BNL). Once the training and testing is verified to be efficient enough we will apply the machine learning with the real dataset.

Introduction

X-ray scattering is a powerful technique for probing the physical structure of materials at the molecular and nanoscale, where strong X-ray beams are shined through a material to learn about its structure at the molecular level. This can be used in a wide variety of applications, from determining protein structure to observing structural changes in materials. Modern x-ray detectors can generate 50,000 to 1,000,000 images/ day, thus it's crucial to automate the workflow as much as possible. Our goal is to build and automatic machine learning pipelines for data analysis tasks including a hierarchical physics aware learning, streaming and real-time learning capabilities, experiment and decision making assistance.

Machine Learning, figure 1, itself is undergoing a shift with a re-thinking from traditional, naive neural networks, towards deep learning models where the neural hierarchy is more rational, optimized, and informative. This has already led to clear advances in several fields including computer vision and speech recognition, and we aim to demonstrate similarly transformative gains with respect to scientific image streams. The core idea in deep learning is to design multiple levels of representations corresponding to a hierarchy of features, wherein the high-level concepts and knowledge are derived from the lower layers.

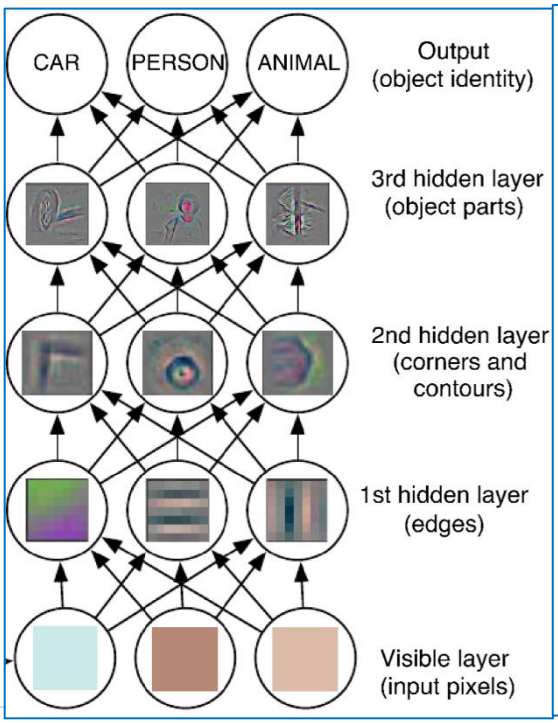

FIGURE 1

These methods will be used in real-time to extract hierarchical and physically-meaningful insights from scientific datasets collected at NSLS-II beamlines:

1) Low-level: identifying characteristic features in a diffraction image;
2) Intermediate-level: detecting the occurrence of a physical process from a sequence of images; and
3) High-level: learning and predicting scientifically-meaningful trends.

For machine-learning, there are two types of datasets that are used. The first is real dataset, which is collected by shining powerful x-rays through a material and the attribute is labeled by material experts. The second type dataset is synthetic scattering dataset, where the data is generated by simulation software. Real data for be hard to get in the quantity needed or it can just not be ready of yet. Because of this, we generate synthetic x-ray scattering images for training and testing of the image recognition and tagging software. The simulation software can generate synthetic scattered images based on physics laws making it as close to the real dataset as possible.

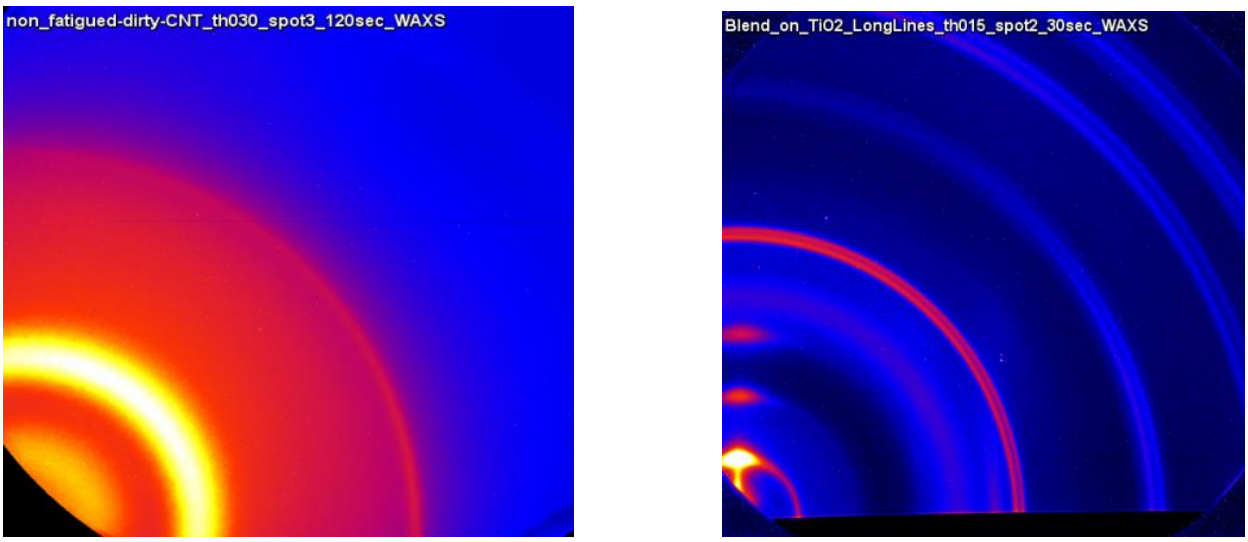

FIGURE 2

Examples of false color images with the "Ring" tag. Tags can include a diverse selection of images, which makes classification of x-ray scattering images difficult

Previously, the synthetic images, figure 2, that were generated had a size of 256 x 256 pixels as seen in Figure 2. Now, we believe if we increase the size of the synthetic images that we generate, we will be able to obtain more information/ data to improve the classification and tagging of the images. If this is seen to be true and efficient then the scientist at the National Synchrotron Light Source II (NSLS-II) can build an x-ray machine or calibrate it to get larger images. The size of synthetic data that we have increased to are 1,000 x 1,000 pixels.

Methods

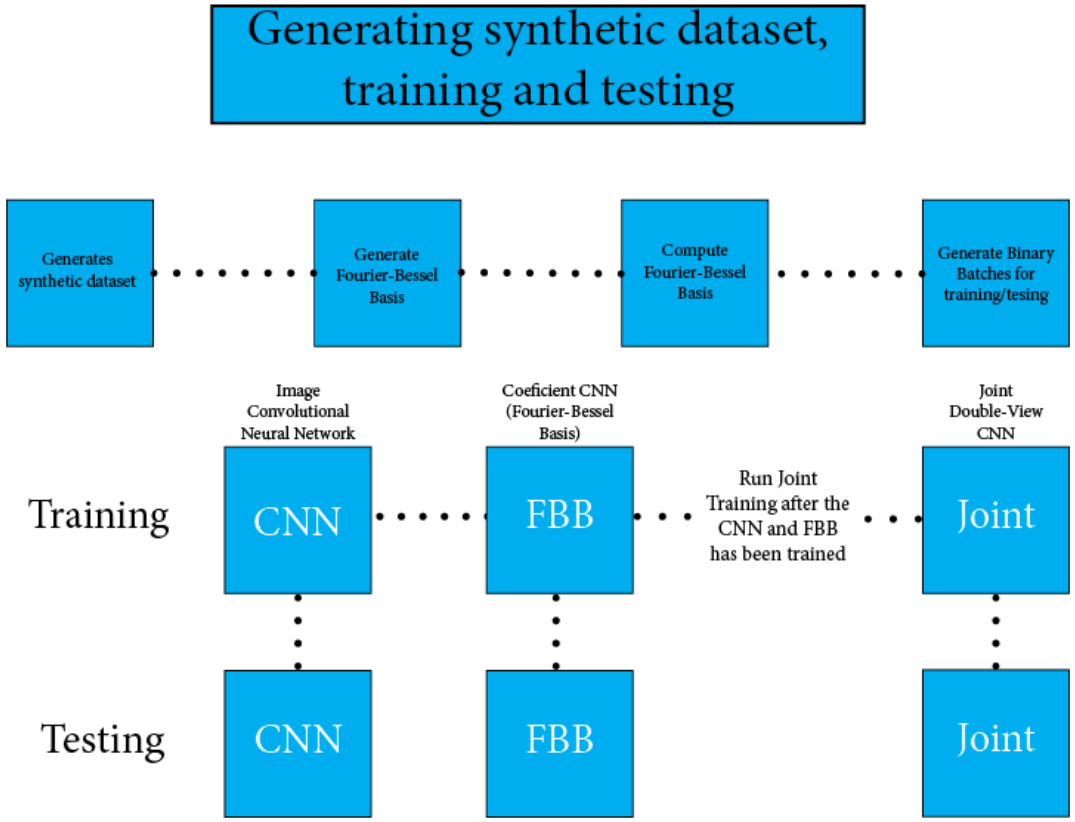

FIGURE 3

Process for generating, training, and testing of synthetic x-ray scattering images.

Simulation

First, we will generate synthetic images to use for testing and training. The synthetic images will be generated using various experiments that have certain variables placed to develop x-ray scattering images. Some of these variables are constant and will be the same across an experiment(s). These experiments use physics properties to generate the data and images. It also uses randomization to avoid creating duplicate images which would be no help when it comes to training. The randomization also helps when it comes to running multiple jobs to generate the data. Without the randomization, there would be a chance that each job would produce identical datasets. Previously, the image size that was generated were 256 x 256 pixels. Now we have changed a few parameters so that the image that is generated is 1,000 x 1,000 pixels. The benefit of

generating a larger image with a higher resolution is that it provides us with more information of the attributes, increasing our ability to classify the images.

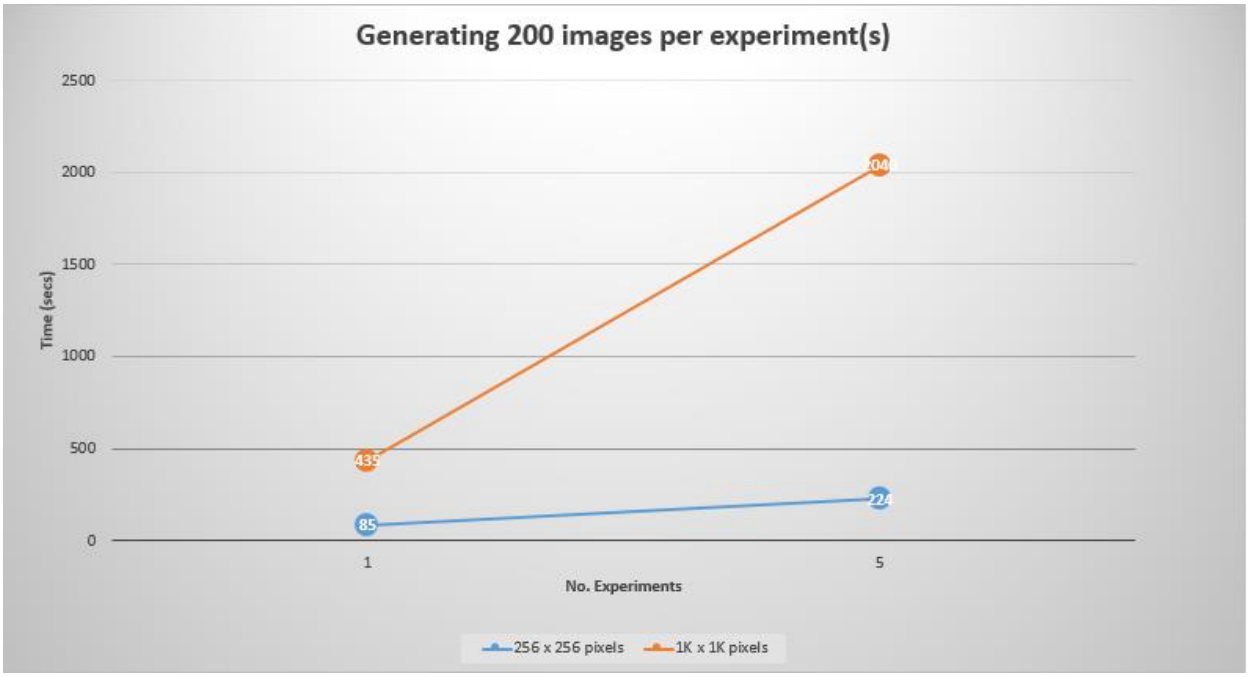

FIGURE 4

Sample display of time to generate images of 256x256-pixel size and 1k x 1k-pixel size. This has been test for 1 experiment of 200 images and 5 experiments with 200 images each.

Big Data Issues

When generating larger synthetic images, we believed that it will take longer which was true. The cluster/ server we are working on only allows us to run jobs for 24 hours and then they will be killed. Because of this, we had to generate the images by running multiple jobs on the server. We produced 50K images for both the 256 x 256-pixels size images and the 1k x 1k images. To generate 1 experiment with 200 images, it took 85secs (1 min 25 secs) for the 256 x 256-pixel and 435secs (7 mins 15secs) for 1 experiment with 200 1k x 1k- pixel images. To generate 5 experiments of 200 images 224 (3 mins 44secs) for 256 x 256-pixel images and 2040 secs (34 mins) for 5 experiments of 200 1k x 1k-pixel images. To generate an experiment with 200 images, it takes the 1k-pixel size image approximately 5 times as long as the 256-pixel size images. To generate 5 experiments with 200 images each, it takes the 1k-pixel size images approximately 9 times as long as the 256-pixel size images. From this rate shown in figure 4, we can see

that to produce 50k images of the 1k x 1k-pixel size will take a lot longer than producing 50k images with 256 x 256-pixel size.

Fourier-Bessel Basis

After running the simulation code, which generates the synthetic images we generate and then compute the Fourier-Bessel Basis. This is slow and could take a day or approximately a few days with the original dataset of 256 x 256-pixel images, therefore we assume it will take much longer for the larger dataset. When it came time to compute the Fourier-Bessel Basis, we had to make some changes in the code. Instead of using the Multiprocessing pooling python module like we did with the 256 x 256 dataset, we had to use a loop for parsing. This change had to be made because of the multiprocessing pooling module not being able to handle the larger size dataset.

Convert Fourier-Bessel Basis

Following that we generated Binary Batches for training and testing. There are 10 batches that are generated which each contain data from the images. We condense all the generated data in batches of raw data records for fast I/O in TensorFlow. Batch files are divided into fixed length byte intervals that save a (tags, image pixels, FBT) tuple each. Each batch consist of 5000 images each, which is a set parameter in the convert FBB (Fourier-Bessel Basis) program.

Training and Testing

Once the batches were completed we entered training. The 3 types of training are Image Convolutional Neural Network, Coefficient Convolutional Neural Network, and Joint Double View Convolutional Neural Network (combination of the two). For the 256 x 256 dataset, Image CNN can achieve mean average precision of approximately 0.5-0.6 with training. The Coefficient CNN can outperform Image CNN by approximately 0.1. The Joint double-view CNN should be slightly better than both. From this we predict that the results will be proportionally the same with the Joint Double-view CNN being better than both the Image CNN and Coefficient CNN individually. For each type of training we used 9 of the 10 batches and saved the 10th batch for testing. Image CNN and Coefficient CNN must be trained before we trained the Joint Double-view CNN but we were able to train the Image CNN and Coefficient CNN simultaneously. We monitor the training

(learning) progress by the means of observing learning error (console output) and tensor-board. Training error will stop descending and stay (roughly) constant at one point. Once we notice this, we no longer need to wait for the preset training steps to finish so we can kill the process. Training creates checkpoints so that if we add new data for training, we will be able to pick up from where we left off. This keeps us from having to restart training from the beginning if it is not necessary.

Once Training is complete we can test the machine learning. With the synthetic data, we will use the tenth batch (batch-9) for testing. This batch has been saved or testing purposes and wasn't included in the training to avoid the system from being bias.

Conclusion

Due to the length of time it takes to generate and prepare the data for training and testing, we haven't yet trained or tested the system with the 1k x 1k images. There are modules that cannot be used for the 1k x 1k-pixel images like they were used for the 256 x 256-pixel images. The goal is to find python modules that are compatible with large data sizes. This will allow the code to be used on all systems and avoid changes when using a larger dataset. We believe this will be an ongoing problem due to us wanting to max out infinite amounts of data.

After training and testing was complete and the accuracy error rate was satisfactory, we can now use the real dataset from NSLS-II to test the system. Once that is done and is satisfactory, we will look to increase the data size again from 1,000 x 1,000-pixel size to 2,000 x 2,000-pixel size. We are looking to gain even more data to improve machine learning classification of x-ray scattering images. With the dataset now being larger, we need to convert from using Alexnet to using VGGnet. VGGnet is suitable for larger data sizes while Alexnet is suitable for smaller data sizes. The process of using Alexnet was successful for the 256 x 256-pixel images. By switching to VGGnet, we expect the training and testing to be more efficient. Also, we intend to use Google TensorFlow as a pre-processing tool and GPU computing to speed up pre-processing.

Acknowledgements

This project was supported in part by the National Science Foundation, Louis Stokes Alliances for Minority Participation (LSAMP) at Lincoln University of Pennsylvania under the LSAMP Internship Program at Brookhaven National Laboratory. I wish to thank my host, Dr. Dantong Yu, for his professionalism and generosity during the NSF program. I would also like the thank my Visiting Faculty Professor and Mentor, Dr. Bo Sun, for her assistance, guidance, and for selecting me as the student to join her team. Further, I would express my gratitude to the Office of Science Education Program, and all who continue to so willingly assist interns in that branch. I very much appreciate the efforts of the National Science Foundation, LSAMP at Lincoln University of Pennsylvania for their support. Finally, I wish to acknowledge the hospitality and kindness of Brookhaven National Laboratory and the Department of Energy.

References

Computer Science Initiative, Brookhaven National Laboratory

Department of Computer Science, Lincoln University of Pennsylvania

Department of Computer Science, Stony Brook University

National Science Foundation (NSF), Louis Stokes Alliances for Minority Participation (LSAMP)

National Synchrotron Light Source II (NSLS-II) at Brookhaven National Laboratory