Using Convolutional Neural Networks For Fast X-Ray Bragg Peak Position Determination

Aditya Sriram, Andy Wan, Boyan Stoychev, Donald Dean, Sean Chen

Introduction

Synchrotron X-ray experiments allow for advanced characterization of materials and devices, enabling data that contains rich information about nanoscale structural and electronic properties. In one such recent experiment performed at Argonne National Laboratory's Advanced Photon Source, an intense X-ray beam was focused down to several microns, and used to study the evolution of the crystal structure of individual cathode primary particles in a Li-ion coin cell battery during charging/discharging cycles. This was done by locating spots in the sample where several Bragg diffraction peaks appeared on an X-ray detector, and repeatedly taking snapshots of these peaks over the course of one charge/discharge cycle.

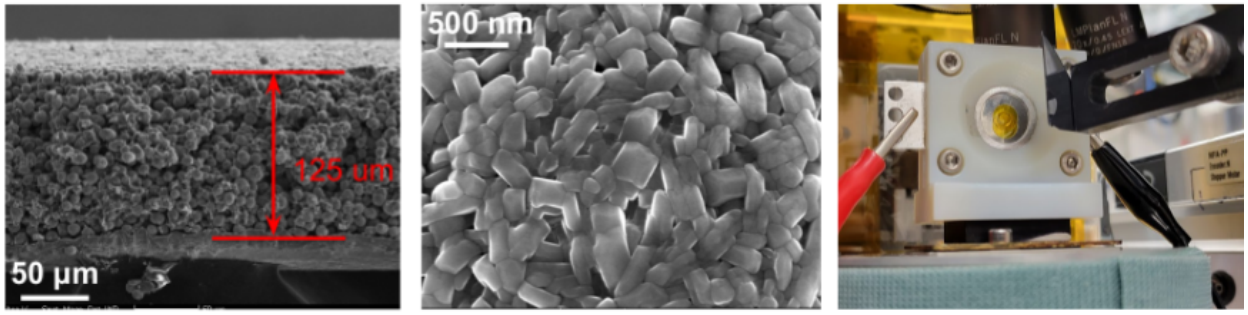

Figure 1: Scanning Electron Microscopy images of battery cathode material (left) and individual nanocrystals (middle). Coin-cell battery with hole for X-rays to pass through, mounted at the synchrotron beamline (right).

Dataset

The data from this experiment consists of several time-series of 2-D detector images containing multiple peaks. Figure 2 shows three example detector images.

In this data, the position of each peak contains information about the crystal lattice size while the shape of it contains information about the particle size as well as strain in the crystal structure.

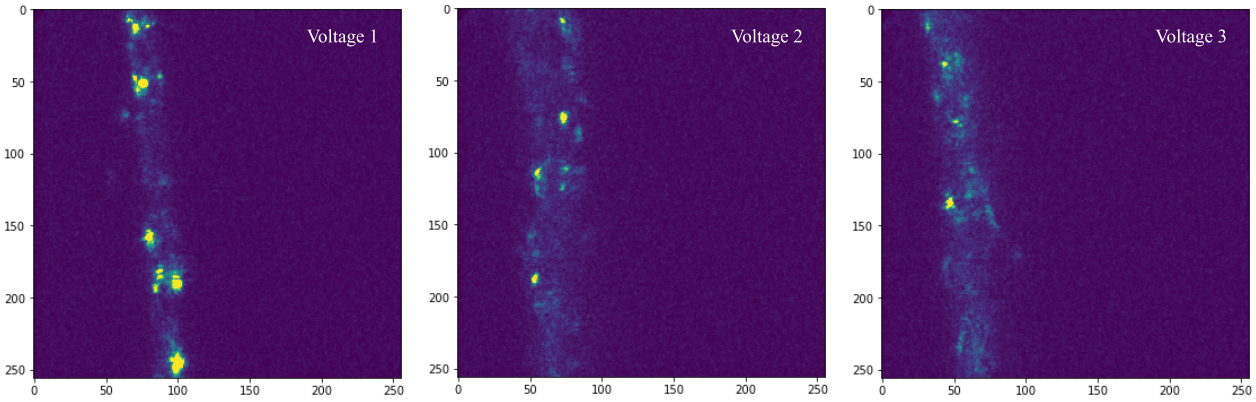

Figure 2: Example dataset showing 2D detector images with Bragg peaks, showing how they shift in position as well as evolve in shape.

Method

The main goal during data processing of many X-ray scattering experiments, including our own, is to determine properties of Bragg peaks on detector images. Traditionally this is done using 2-D Gaussian fitting, but with ever-increasing sizes of datasets, deep-learning methods were implemented to significantly speed up fitting.

BraggNN is a Deep Learning (DL), neural network based, supervised model developed at Argonne National Laboratory. It is capable of determining the center of mass of a bragg peak with sub-pixel precision. The general idea of such a neural network is to extract higher level features from the input, often in the level of pixels, through a hierarchy of multi-layer frameworks. Previous studies have shown that Convolutional Neural Networks (CNN) are parameter efficient due to the translational-invariant property of its representations [1]. Additionally, convolutional weight-sharing, where the same weights are shared across the entire image, is also proven to be beneficial for good model performance [1].

BraggNN is able to precisely determine the center-of-mass of the bragg peak much faster than conventional 2-D Gaussian fitting. When applied to a test dataset, BraggNN gives errors of less than 0.29 and 0.57 pixels, relative to the conventional method, for 75% and 95% of the peaks, respectively. When applied to a real experimental dataset, a 3D reconstruction that used peak

positions computed by BraggNN yields 15% better results on average as compared to a reconstruction obtained using peak positions determined using conventional 2D pseudo-Voigt fitting. With consumer-level computers, BraggNN is able to perform nearly 200 times faster than conventional fitting methods [1].

The architecture of BraggNN consists of a series of CNN layers and Fully Connected (FCN) layers (Figure 3). Each CNN kernel is acting as an artificial neuron that extracts a particular feature (edges, colors, etc) and each neuron has 3 × 3 × c learn-able weights plus one learn-able bias to convolve a feature map (a 3D volume shaped as height × width × depth/channel) with c channels [1]. In between the layers, a ReLU activation function is used to yield the features. The 3D feature map produced by the last CNN layer is reshaped into a 1D vector before feeding it into the first FC layer [1].

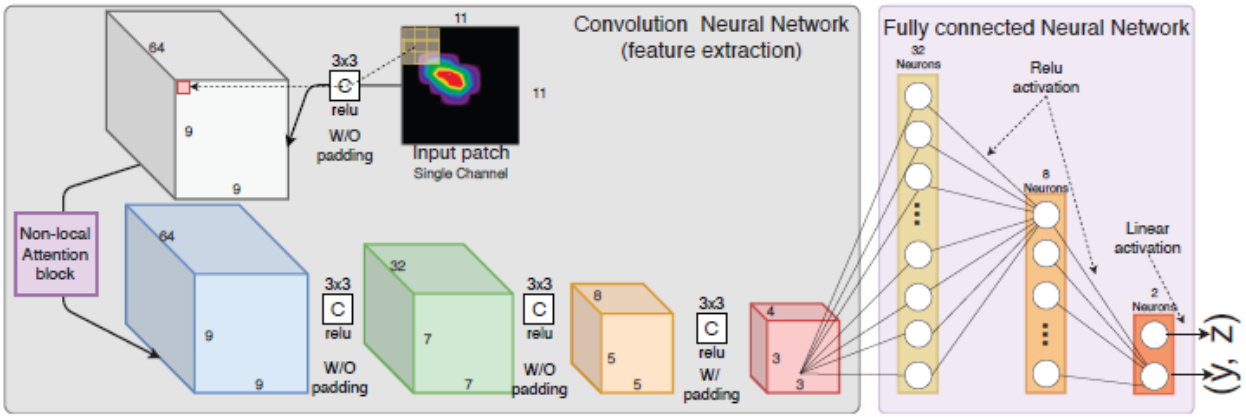

Figure 3: BraggNN architecture

Similar to the CNN layers, each FC layer contains multiple neurons with the same number of learnable weights plus one learnable bias. N neurons in a layer will generate an output vector of dimension N, which will be fed into the next layer and so on. Each neuron in a FC layer is connected to all neurons in the previous layer and the output function has no activation function. The model loss is defined as the l_2 norm, computed between the model output and the ground-truth (via Gaussian fitting). The gradient of each neuron's learnable weights are computed using back propagation and updated by the gradient descent method. Training iterates the feed-forward and back-propagation process on different (Bragg peak patch, ground truth

center) pairs many times, until the model no longer makes noticeable progress in minimizing the l_2-norm [1].

Results

For most of our project, we used a pre-trained BraggNN model and tested its performance on our dataset, compared to conventional 2D Gaussian fitting. As a first step, we simulated an 11x11 patch containing a Bragg peak and random noise, with a 100:1 signal to noise ratio. The pre-trained BraggNN model we used takes 11x11 sized patches with min-max normalization as input. Feeding our simulated patch to the model showed very precise results, with an error of only 0.062 pixels (Figure 4)

We then moved on to test it on a sample detector image from our data. Since the Bragg peaks in our data are significantly noisier and more irregular than the data used to train the model, we had to carefully optimize our pre-processing, as well as criteria used to identify peaks on a detector image.

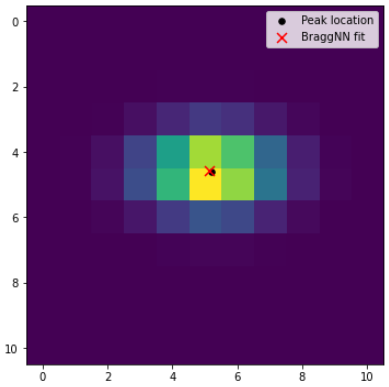

Figure 4: Model fit on simulated patch containing a Bragg peak

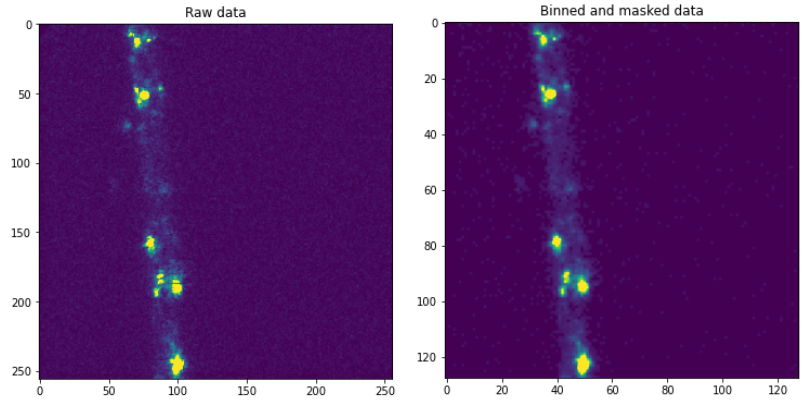

Figure 5: Raw data (left); Binned and Masked data (right)

Because the model works on 11x11 patches that contain a single peak, we needed to roughly locate Bragg peaks on our detector, and generate patches around those to feed to the model as input. To achieve this, we utilized the module 'findpeaks' (GitHub Link). The topology method of findpeaks returns a list of peak locations on a 2D image, as well as a persistence score that shows how prominent that peak is.

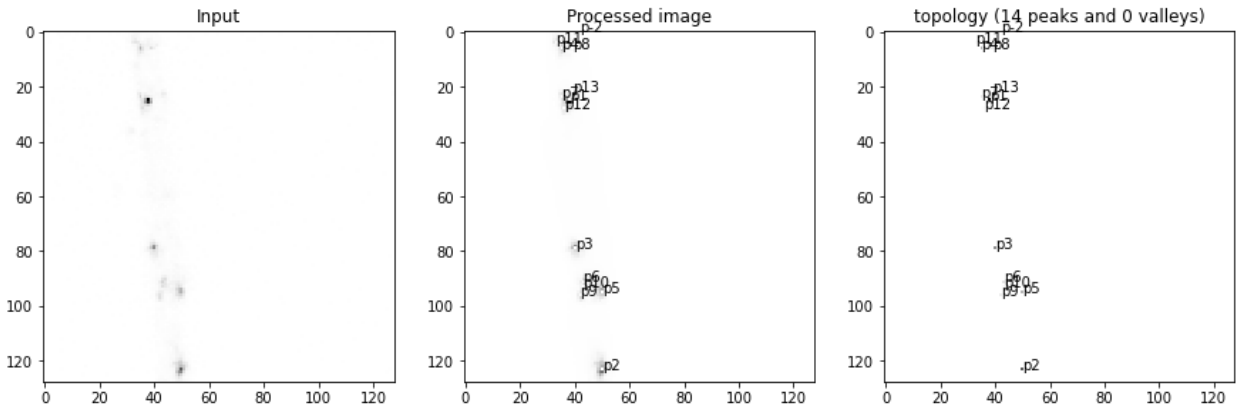

Figure 6: Example output from findpeaks's topology method on our detector image, showing 14 peak locations

We then filter the peaks based on their persistence score, and generate 11x11 patches centered on each peak. We then pass these patches as input to the model, which outputs a more accurate peak location with sub-pixel precision. Figure 7 shows some examples of input patches from our real data, together with the model fit. Values within each patch are normalized to (0-1) range.

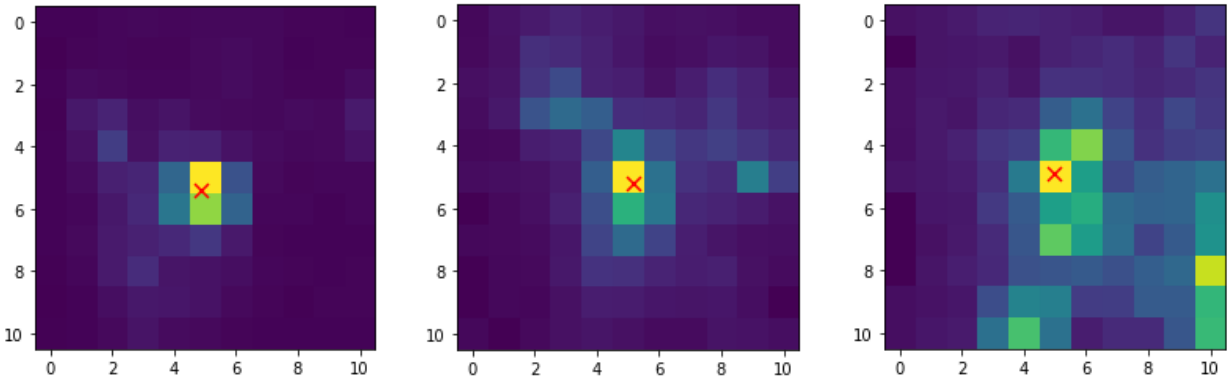

Figure 7: BraggNN output using real data

Figure 8 shows the entire detector image with peak positions roughly determined by findpeaks, and more precisely fitted with the BraggNN model.

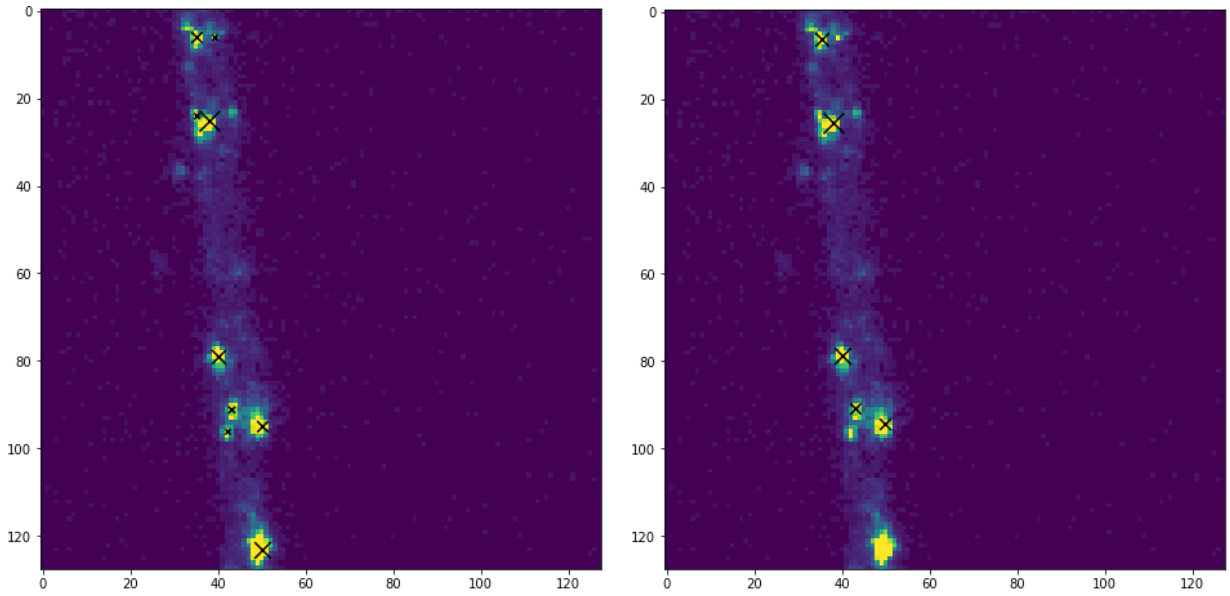

Figure 8: Detector image with 'findpeaks' output (left); Detector image with BraggNN output (right). The size of the black cross corresponds to the peak's persistence score. Note that not all peaks from findpeaks() could be fitted with BraggNN successfully, either due to being too close to the detector edge, or due to containing more than one peak within a patch

In order to evaluate the performance of the pre-trained model on our data, we also implemented 2D Gaussian fitting on patches to compare against the model fit. Figure 9 shows some examples of those results - as expected, the model performs best on sharp, well-defined peaks since those resemble the data it was trained on the most.

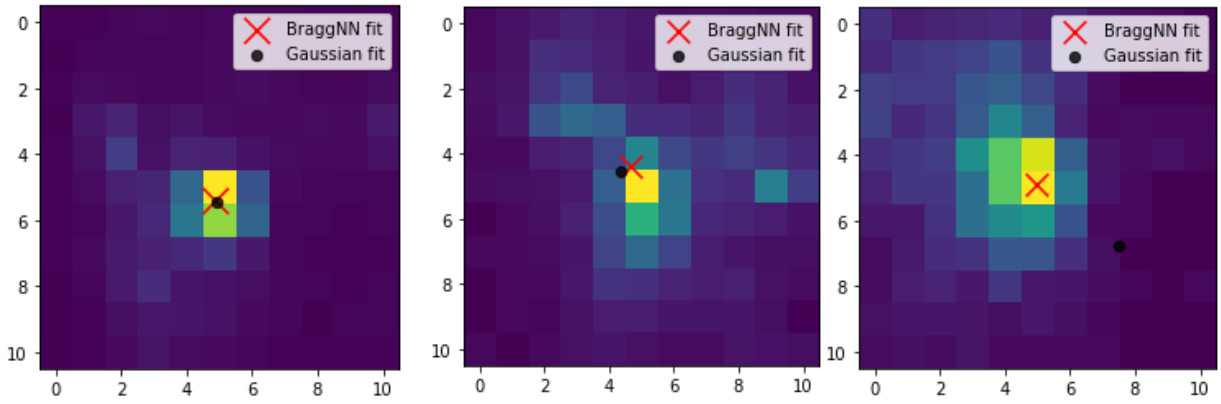

Figure 9: Comparison between BraggNN and Gaussian fits on a sharp peak (left), elongated peak (middle) and broad peak (right). As expected, the BraggNN model performs best on sharp, symmetrical peaks since it was trained on a very clean dataset. For the right patch containing a broad peak, both BraggNN and especially the Gaussian fit failed to accurately capture the peak location.

We then ran the pipeline described above on an entire dataset consisting 480 frames. We included two tunable parameters: `minval` was the threshold below which detector pixel values were set to 0, and `minscore` was the threshold for persistence score determined by findpeaks to pass a particular peak as input to BraggNN. Figure 10 shows the errors between the BraggNN and Gaussian fits for two pairs of `minval` and `minscore` values - as expected, the error is smaller when using higher threshold values, since those will filter out noisier patches and focus on ones containing sharp, well-defined peaks.

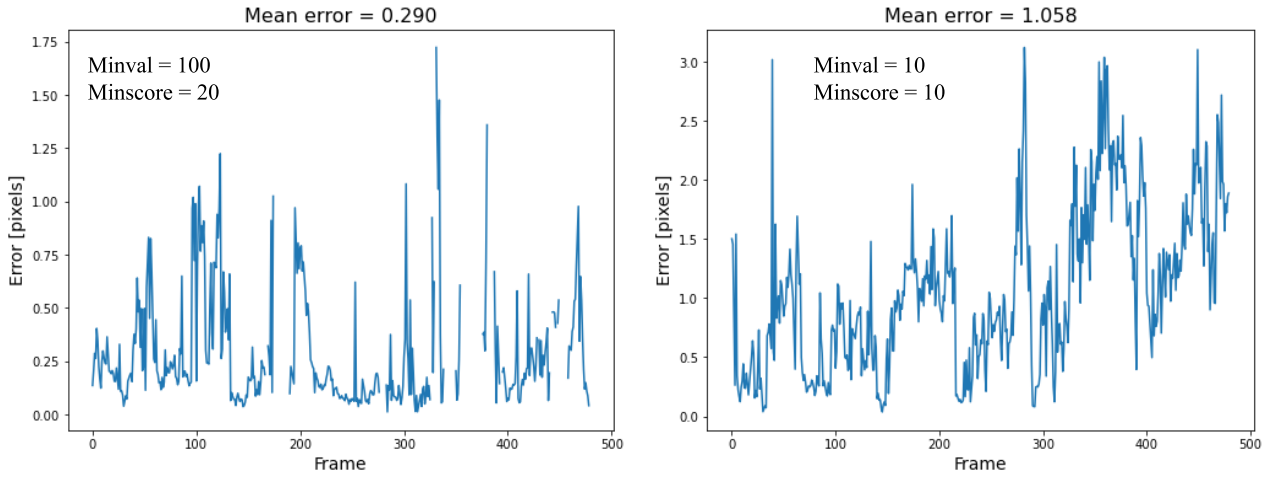

Figure 10: Errors as a function of frame number for two different sets of parameter values.

For the lower threshold values, we present our results using an animation that overlays each detector frame from the dataset with the peak locations as determined by the model. (link)

Conclusion

The goal of our project was to understand how the BraggNN model works by using a pre-trained model to fit Bragg peaks in our data, as well as re-train it on our data. We successfully developed a pipeline to feed our data into the pre-trained model, and saw that it can perform very well on clean, sharp Bragg peaks, but its performance deteriorates significantly when the peaks are less symmetrical and sharp. Our attempts to re-train the model were plagued with multiple issues that we ran out of time to troubleshoot, but a potential future solution would be to re-create the model architecture from scratch. More future work would include further refinement to filtering what we classify as a peak to feed to the model, as well as some padding methods at detector edges to allow for fitting of peaks near the edge.

References

[1] Liu, Z., Sharma, H., Park, J.-S., Kenesei, P., Miceli, A., Almer, J., Kettimuthu, R. & Foster, I. (2022). BraggNN: fast X-ray Bragg peak analysis using deep learning. IUCrJ, 9, 104–113.

[2] Link to our GitHub repository: https://github.com/miwan1/PHYS139_Group6_Project