Automated Sorting of X-ray Scattering Patterns with Convolutional Neural Networks

Daniela Ushizima[1,2,†], Flavio Araujo[3], Romuere Silva[3],
Harinarayan Krishnan[1], Eric Roberts[1] and Alexander Hexemer[1]
[1]Lawrence Berkeley National Laboratory, Berkeley CA, USA, 94117, [2]University of California Berkeley,
Berkeley CA, USA, 94117 [3]Federal University of Piaui, Picos PI, Brazil, 64607-670
{dushizima, flavio, romuere, hkrishnan, eroberts, ahexemer}@lbl.gov

Abstract

X-ray scattering is an experimental technique that generates patterns conveying material structural information. This technique presents four main modes: small and wide-angle X-ray scattering (SAXS and WAXS) and their respective surface-sensitive variation due to grazing-incidence known as GISAXS and GIWAXS. During a single, high-throughput experiment, these modes can be used interchangeably, therefore sorting is challenging. This paper proposes machine learning tools to sort datasets with thousands of patterns from experiments performed at a synchrotron-light beamline. Each pattern undergoes featurization with convnet, then the extracted features serve as input to AutoML, an automated paradigm for classification and performance evaluation. Our analysis shows that different convnet architectures lead to sorting schemes with high accuracy rates, achieving accuracy of 99.11%.

Index Terms

X-ray scattering, deep learning, convnet, AutoML, HPC

I. Introduction

X-rays have been broadly applied in the structure characterization of matter [13, 38, 36], for example, to inspect noncrystalline samples using X-ray scattering [2], which is an experimental technique that collects images by shining collimated X-ray beams through a material of interest. A two-dimensional detector captures the scattering patterns that result from interference between elements composing the sample structure [16]. The resulting scattering patterns reveal details about the physical structure and properties of targeted materials on the molecular and nano-scale, including crystalline unit cells, molecular and atomic spacing [26], and surface roughness [39]. More recently, this imaging modality has been used to provide information about materials on a sub-nanometer scale, such as the stability and flexibility of sub-nanometer polymer-like wires used in sub-nanometric material design [31, 27] and the effects of sub-nanometer biomembrane thickness on intracellular transport [19] and cellular reactions to pathogens and drug designs [32].

In order to capture the diffuse scattering patterns from disordered systems, distinct techniques have been developed, including small- and wide-angle X-ray scattering, SAXS and WAXS, and their respective grazing-incidence variations known as GISAXS and GIWAXS, whose geometries differ due to the surface-sensitive variation. During a single high-throughput experiment [2, 28], these different techniques can be used interchangeably. The data collection rates are challenging for manual interaction; real-time screening and analysis of the acquired patterns are required to keep pace. While human-interaction will continue to play an important role in the analysis process, the massive amounts of data, e.g. 10 petabytes per year [36], produced at some experimental facilities necessitates automation.

This paper proposes a set of algorithms for sorting scattering data from experiments into the different X-ray scattering techniques of SAXS, WAXS, GISAXS and GIWAXS, using deep learning and AutoML, an end-to-end automated machine learning tool [1], as illustrated in Figure 1. Our dataset includes thousands of patterns from real experiments performed by different users at the Lawrence Berkeley National Laboratory synchrotron light source. Each scattering pattern undergoes featurization using 8 methods for pattern description, including 6 deep learning architectures, as discussed in Section III-A. Different combinations of training sets and feature extraction schemes are responsible for generating a total of 25 possible representations per pattern, once different training strategies are considered. These are input to AutoML (Section III-B) that creates 5 different classification models for each of the featurizations, which are individually evaluated using 7 metrics of performance. Due to the combinatorial complexity of these different approaches, we use `dask` to parallelize the classification using AutoML, as in Section III-C. In order to improve interpretability of the results (Section III-D), we show the Gradient-weighted Class Activation Mapping (Grad-CAM) for scattering patterns in each class. Finally, we summarize performance results in Section IV.

Our exploration shows that different choices of convolutional neural network (CNN) architectures lead to sorting schemes with similar accuracy rates (over 97%) at diverse computing times. It means that deeper neural networks performed as well as their shallower counterparts, but shallow ones are faster to retrain and require less labeled data. These are promising results and represent a first step toward the automation of sorting of patterns for metadata recovery and/or verification, and enabling autonomous experiments for film design.

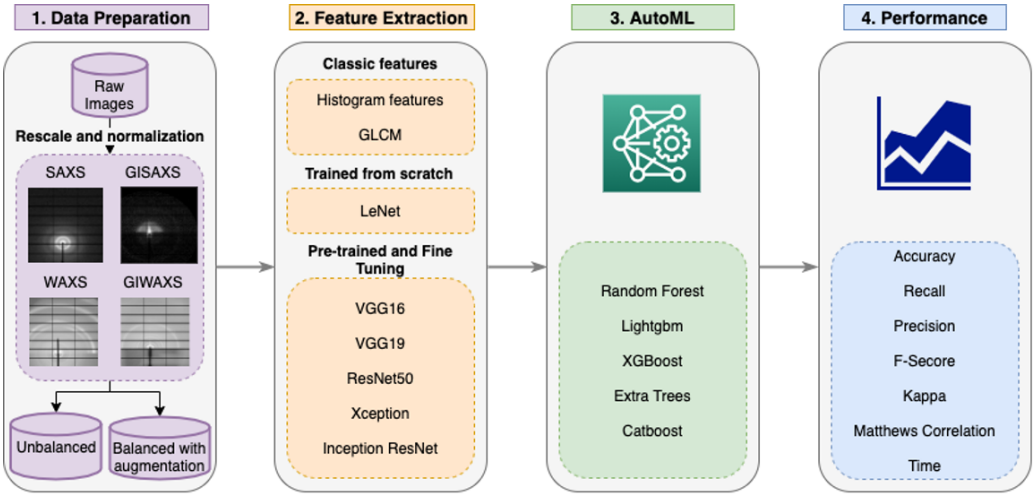


Fig. 1: Diagram of camSortXS with main steps in sorting X-ray scattering data.

II. Background

A. Challenges in X-ray scattering

In energy-related research, thin-film polymeric materials play a major role in improving properties for batteries or organic photovoltaics (OPV). OPVs usually require thin layers of small molecules or polymers (usually below 100 nm thick) to optimize efficiency by balancing photon absorption versus the length that electrons and holes have to travel to reach an electrode. In addition, the structure inside the thin polymer layer requires an even smaller length scale on the order of just a few nanometers. The small structure allows the exciton created during the photon absorption, to diffuse to a charge separation area in the material [24].

These types of the materials are being investigate at the Advanced Light Source in Berkeley, one of brightest soft X-ray sources in the world, providing 40 beamlines with intense and coherent short-wavelength light for use in scientific experiments by researchers worldwide. As an example, grazing-incidence small angle X-ray scattering (GISAXS) enables the extraction of morphological information about both the thin film surface and the depth of embedded structures within the sample [21]. In tandem with extremely bright X-ray sources, the development of large and fast two-dimensional X-ray detectors provides an invaluable tool for in-situ time-resolved experiments, while creating a deluge in X-ray scattering data [35].

Roughly speaking, the choice of X-ray scattering experiment includes two different modes, namely small-angle scattering (S) and wide-angle diffraction (W). The approach depends on which structures, morphologies, or motifs are being targeted. For example, SAXS focuses on nanometer-size features while WAXS is more suitable for molecular and crystal structures of polymers. Most users usually employ more than one approach in order to understand the hierarchical structure of the compounds, e.g., combining SAXS and WAXS or GISAXS and WAXS. When the sample consists of a thin-film on a flat substrate, e.g. a silicon wafer, GISAXS or GIWAXS techniques are suitable choices to probe the surface as well as the internal structure of the material. Such experiments are essentially driven by the user, based on the properties of the sample; currently the manual mode of operation implies that the user will look at the recently acquired data and decide which scattering technique would work best next. In practice, a user adjusts the sample detector distance to a range that should suit the structure of the material. Initial experimental data are inspected by hand and a decision is made to continue and collect more data, or to change the experimental setup and adjust the sample detector distance and therefore the sold-angle of the collected data. The decision is driven by information content inside the initial data. For example, if the user notices several peaks (bright spots) just around the beam stop, but nothing else in the detector image, the user may choose a different small angle setup to better identify the peaks and extract more precise information for the experiment.

In order to begin the process of data reduction, it is important to identify the geometry of the experiment. However, when metadata is not available, the challenge is to automatically recover which technique was used. To exacerbate the problem of metadata recovery, multiple different techniques at the ALS have been intertwined for more than ten years, leading to a rich and invaluable, but somewhat chaotic data collection consisting of a mix of these image patterns.

To summarize, the main analysis tasks include:

- Indirect interpretation of physical properties from reciprocal-space data;
- The search for and recognition of features visually, such as peaks, arcs, rings, and rods;
- Verification of current X-ray scattering technique - if not appropriate, move detector; and
- Pattern measurements to infer sample properties, such as homogeneity, and decision about next steps in the experiment and analysis pipeline.

B. Machine Learning for Pattern Classification

Classification and ranking of scientific images [3], which includes the extraction of features and metrics from X-ray scattering patterns to infer the structure of probed materials, has become an unfeasible manual effort across laboratories. Most previous works using deep learning to automate classification and sorting report results on simulated data [2], in part due to the lack of publicly-labeled datasets from experimental settings.

Options for generating simulated data include several open source programs, which model the underlying physics associated with the generation of X-ray scattering patterns, such as BornAgain [33] that is a research software used to simulate and fit grazing-incident small angle scattering (GISAS) reflexometry using both neutrons and X-rays, generally used in Europe [20]. Similarly, HipGISAXs [11] is a pattern simulation software, based on the Distorted Wave Born Approximation (DWBA) theory, and it is broadly used across American institutions to generate synthetic data.

Despite providing realistic data by computing scattering intensity patterns for a variety of sample orientations, morphologies, etc., using simulated data present some challenges: for example, the images are unrealistically "clean", often misleading to impressive accuracy rates, which may not generalize when transferring learning to real data [2]. Later work addressed the challenge of increasing variability of synthetic data by exploring different sources of noise, such as varying smear effects, Gaussian noise levels, Poisson shot noise, and multiple image resolutions [28]. These studies provided insights regarding data fluctuation and opportunities for pattern compaction, however the high accuracy from such models was still restricted to simulated data.

While a wide spectrum of problems have been tackled with CNNs, to the best of the authors' knowledge, the work presented here is one of the first network designs for sorting X-ray scattering patterns into SAXS, WAXS, GISAXS and GIWAXS. Analogous data inputs were investigated by Zhong and Xu [43], who explored pattern representation to improve reconstruction from the reciprocal space; differently, and in contrast, we keep the scattering patterns in the reciprocal space, which are input to featurization using CNNs, followed by data compression. Similarly to Fountsop et. al. [12], we explore architectures with different depths: one key difference is that they seek compressed models by considering data quantization for fast training and testing, while our proposal focuses on the searching for the best possible image representation (Section III-A) to create robust models, and data compression (Section III-B) of feature vectors that can have more than 4,000 descriptors.

To address these challenges, in the next section we describe our work on camSortXS, which is a set of essential methods to build digital infrastructure for automated identification of X-ray scattering imaging techniques: one important use is to apply the correct geometry change to the data when converting from pixels to q-space, which often requires knowledge about the sample detector distance and whether reflection or transmission geometry have been employed.

III. Materials and Methods

This paper creates deep learning models applied to a publicly-available dataset consisting of images of X-ray scattering patterns. The dataset contains ≈700 images per class without augmentation, with each image measuring 256x256 pixels and 64 KB in tiff file format. Access to materials for reproducibility are available on github [1].

There are 4 possible types of experimental techniques within the dataset, pictured below in Figure 2 and listed as follows: (a) GISAXS: grazing-incidence small angle X-ray scattering; (b) GIWAXS: grazing-incidence wide angle X-ray scattering; (c) WAXS: wide angle X-ray scattering; (d) SAXS: small angle X-ray scattering.

Each of these datasets contain patterns coming from three or more users, as well as different types of materials. Neither the user nor the material type were disclosed or included as prior information to our experiments.

A. Featurization: from Texture Descriptors to Deep Learning Fingerprints

1) Texture Descriptors:

a) Gray Level Co-occurrence Matrix (GLCM): This technique evaluates co-occurrences between pairs of pixels which follow a given pattern, distance, and angle [17]. The result is a co-occurrence matrix from which we compute the following image features: contrast, correlation, energy, homogeneity, angular second moment (ASM), and entropy. GLCM has been broadly used in cervical cell description, cytology, mammogram, and lung cancer detection.

b) Histogram Features (HF): Histogram-based features consist of values for the average, entropy, energy, variance, skewness, kurtosis, and roughness computed from the image histogram. Despite its simplicity, histogram features have been broadly used for brain tumor detection, breast cancer detection, and vibration signal analysis for mechanical systems [22].

2) Deep Learning Fingerprints: Recent works typically employ three main different ways of using CNNs for pattern featurization with the intent of providing data fingerprints and classification. The first is a more time-consuming approach, in which the training is performed from scratch with a large set of data. The second is through transfer learning using pre-trained networks trained in a large natural image database, such as ImageNet, which contains over 14 million images from 1,000 classes. Thus, the neural network can assimilate generic features, facilitating its application to small databases. The third is the fine-tuning technique that consists of continuing the hyperparameter adjustment training of a pre-trained network by using data

[1] Source code and documentation for camSortXS: http://bit.ly/camSortXS

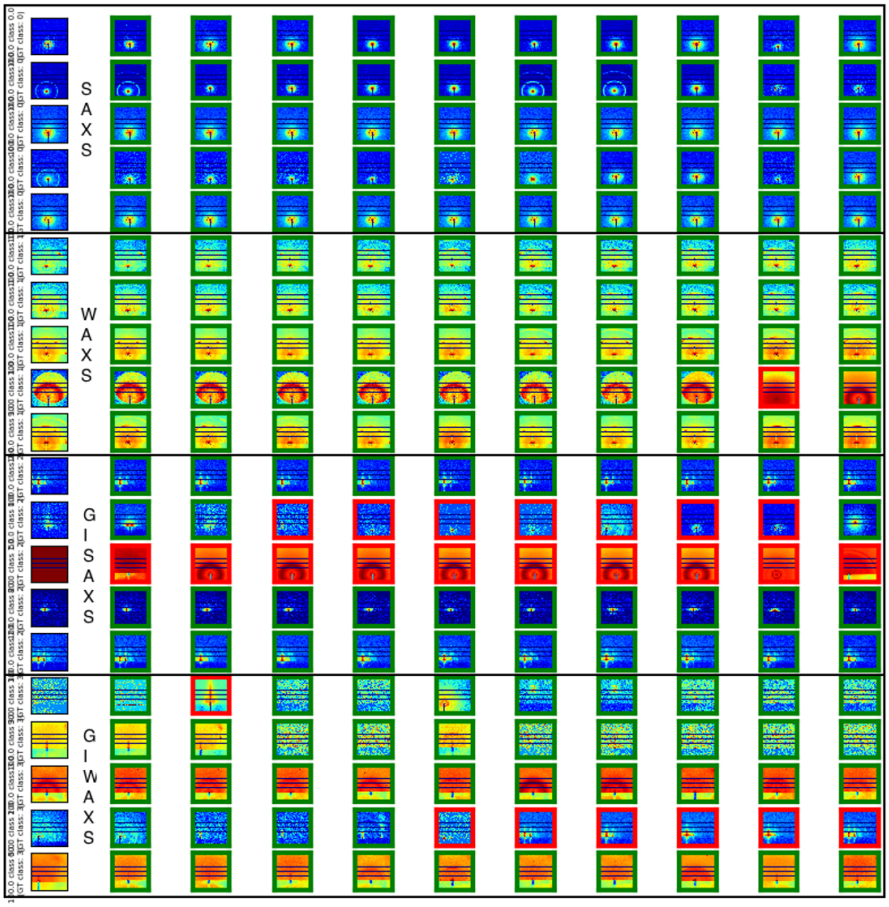


Fig. 2: Randomly chosen X-ray scattering patterns (first column) are input to a similarity search with retrieval of top-10 matches: featurization is performed by LeNet and similarity by approximate KNN [2]. We emphasize successful cases with green borders, and failures with red borders.

from a new image set [2]. Yet another variation of each of these schemes involves the featurization portion of the CNN used as input to more sophisticated non-linear mappings; for example, one could substitute the several max-pooling operations and soft-max activation function for a support vector machine [42] to define a mapping from inputs to outputs, therefore unveiling a class prediction of an unseen object – this is the approached adopted in this paper, combined with the following featurization schemes: (a) LeNet model trained from scratch, (b) VGG16, VGG19, ResNet50 [8], Xception [10] and Inception Resnet [41] as pre-trained models on ImageNet, (c) their variants with fine-tuning. In both LeNet training and deep models fine-tuning, Stochastic Gradient Descent (SGD) was used for the optimization and the loss function was taken as the categorical cross-entropy, which have become straightforward approaches in multi-class classification problems. The next sections summarize important aspects of each CNN available within camSortXS, and how to obtain the feature vectors for each architecture.

a) Lenet: The Lenet architecture switches from fully connected to sparsely connected neurons, often allowing feedback in real-time. Due to its simplicity, it can manage recognition problems with a smaller number of examples during training in comparison with deeper CNNs, however it tends to be less accurate when dealing with complex recognition tasks. In this work, we used a Lenet with three convolutional layers with 64, 64, and 48 convolutional filters of size 3×3. There is a max-pooling with kernel size of 2×2 and a batch normalization layer after each convolutional layer. The classification network often has two fully connected layers with 192 and 64 neurons, with dropout value of 50% between them to generalize the learning process [40], followed by an output layer with k neurons whose response is modulated by a soft-max function. In this paper, we intercept the feature vector after the first fully connected layer.

b) VGG: The Visual Geometry Group (VGG) at the University of Oxford proposed the idea of decomposing big convolution kernels (11×11, 7×7 and 5×5) into smaller 3×3 convolution kernels. According to the VGG architecture, multiple 3×3 convolutions stacked together can replicate larger convolution kernels, and there are more non-linear features (in terms of activation function) between them, even larger than with convolution. The difference between VGG16 and VGG19 CNNs reflects the number of layers: the first has 16, while the second has 19. In this architecture, the input size of the images is $224\times224\times3$ and the classification network has two fully connected layers with 4,096 neurons followed by an output layer with soft-max function. The pattern representation follows a scheme similar to that described for LeNet, but here the feature

vector is the output of the second fully connected layer.

c) ResNet50: ResNet50 is a 50-layer residual network. A common problem with deep neural networks is the repeated multiplication that occurs as the network is traversed deeper, resulting in an infinitely small gradient [5]. To address this problem, the residual module introduces a direct step from one layer to the next. Intuitively, these skip steps form a gradient highway where the computed gradients can directly affect the weights in the first layer, allowing updates to have a more meaningful effect [18]. As in VGG19, the input size of this architecture is 224×224×3. The ResNet50 does not have fully connected hidden layers. The sequence of convolutional and pooling layers results in 2,048 features which serve as input to the output layer with the soft-max function: we used the input to the soft-max as our 2,048-length feature vector.

d) Inception ResNet: Inception ResNet contains multiple sub-networks and a much deeper and wider architecture than the ResNet50. These hierarchical layers promote many levels of non-linearity needed for deeper, more extensive pattern classification, generally required for more complex inputs. This model is formed by inception modules combined with residual modules. An inception module is conceptually similar to convolutions (they are convolutional feature extractors), and empirically appears to be capable of learning richer representations with fewer parameters. The input size for this architecture is 299×299×3. As with ResNet50, the Inception ResNet does not have fully connected hidden layers. The sequence of convolutional and pooling layers results in 1,536 features before the soft-max function during model creation: we used these 1,536 values as features afterwards.

e) Xception: Xception is based on the Inception architecture, where separable convolutions replace the inception modules in depth. This process is named "depth-wise separable convolution", often referred as an inception module with a maximally large number of towers. As with Inception ResNet, the input size of this architecture is 299×299×3. The Xception does not have fully connected hidden layers, and the sequence of convolutional and pooling layers results in 2,048 features.

B. AutoML

Automated Machine Learning or AutoML frameworks offer strategies to generate robust data-driven models while minimizing numerous possible choices during key tasks, such as data preparation, model hyperparameter tuning, model selection, and model evaluation. Currently, there are several AutoML frameworks available for model deployment, including Auto Sklearn and Auto Keras. Our paper leverages an open-source AutoML package called PyCaret [1], which is a well-maintained framework that accelerates much of the exploration of possible models, with rapid testing of different statistical models, such as LightGBM, XGBoost, Random Forest, Extra trees, and Catboost, among others.

Before running such classification models, we perform dimensionality-reduction by running principal component analysis (PCA), enforcing all resulting feature vectors to have dimensionality equal to 5. PCA compresses the dataset onto a lower-dimensional feature subspace, which is empirically determined based on the realization that the top 5 components maintain most (over 95%) of the relevant information about the scattering patterns. Additional tests used fast independent component analysis (FastICA), whose results were similar to those using PCA, but at least twice as slow.

In Algorithm 1, each item of List_of_feature_vector_matrix is transformed using PCA. The next sections describe the classification models provided by the AutoML, which are responsible for statistical inference of different classes using the compressed pattern representation.

1) LightGBM: LightGBM or Light Gradient Boosting Machine [23] implements an ensemble method that combines simple learners in a stage-wise approach to obtain a single prediction model. This gradient boosting framework uses tree-based learning algorithms to obtain high-quality prediction models. The main motivations to use this framework are the training speed, the higher efficiency associated with preserving suitable accuracy despite low memory usage, and the ability to handle large-scale data.

2) XGBoost: "Extreme Gradient Boosting" or simply XGBoost is another scalable tree-boosting framework invented by Tianqi Chen [7]. The term "Gradient Boosting" refers to previous work by Friedman [14] on greedy function approximation.

3) Random Forests (RF): One of the motivations to use random forests is the high classification accuracy obtained through the creation of ensembles of trees [6]. RF drives the generation and selection of feature vectors that govern the growth of each tree in the ensemble, and it is known for preventing data overfitting, even with relatively small datasets.

4) Extra Trees (ET): Extra trees, also know as Extremely Randomized Trees Classifiers [29], are ensemble methods similar to random forests on the basis that both consider a large number of decision trees to obtain the best prediction. The main motivation for using the ET method is its higher performance in the presence of noisy features, which is the case of most of scattering patterns.

5) Catboost: Catboost is a state-of-the-art open-sourced machine learning algorithm that implements gradient boosting on decision trees library [34]. This method generally yields competitive performance without the need for as much data for training as other methods. Our investigations have shown that this approach was the most time consuming of the ensemble methods adopted in this paper.

C. Parallelization and AutoML

Dask provides functions to scale python-based workflows in an easy way, distributing our computations across multiple cores. To create different prediction models using AutoML, we execute the `distributed` module to set up a scheduler with

Algorithm 1: Python Pseudocode.

```
Input :
        Dask Configuration:
        - host config: "NERSC CORI".
        - nodes: "number of nodes".
        - cores: "number of cores".
        - conda_env: "Python Dask/Distributed + PyCaret environment".
begin
    Dask Client launches Dask Scheduler and Workers using Dask Configuration;
    classic = [hist,glcm];
    unbalanced = [LeNet, VGG16, VGG19, Resnet, Xception, Inception];
    balanced = [LeNet, VGG16, VGG19, Resnet, Xception, Inception];
    augmented = [LeNet, VGG16, VGG19, Resnet, Xception, Inception];
    pre_trained = [VGG16, VGG19, Resnet, Xception, Inception];
    List_of_feature_vector_matrix = [classic, unbalanced, balanced, augmented, pre_trained];
    def dask_kernel(Feature_vector_matrix):
        List_of_models = [rf, lightgbm, xgboost, et, catboost];
        for each model in List_of_models do
            for each iteration of the model do
                perform operations: create(), tune(), update() -¿ new model;
            end
            write_stats_results_to_disk()
        end
        return None
    # Allocate and assign dask workers to resources;
    # Submit features to workers;
    Map List_of_feature_vector_matrix to remote workers;
    Dask Client [Parallel %Map] to Remote Cluster with (Dask Configuration, feature_vector_matrix);
end
```

several worker processes. In order to speed up a portion of our computations, we created a specific routine (Algorithm 1) that runs on a high performance supercomputer: we used the National Energy Research Scientific Computing facility (NERSC) supercomputer Cori, which is a Cray XC40 with a peak performance of about 30 petaflops, comprised of 2,388 Intel Xeon "Haswell" processor nodes and 9,688 Intel Xeon Phi "Knight's Landing" (KNL) nodes.

D. Grad-CAM

Gradient-weighted Class Activation Mapping or Grad-CAM [37] is a technique used to visualize the activation maps of each class and explain which image parts or pixel sets contributed more to the final output of the CNN model. In other words, CAM verifies if specific image areas are associated with a particular class, allowing better model interpretability by the material scientists. It calculates the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map highlighting "important regions" associated with prediction.

To preview the X-ray scattering image classification and demonstrate Grad-Cam, Figure 3 shows heatmaps corresponding to the class activation maps of the four different scattering techniques. The maps show the well-known concepts expected in these imaging modalities, such as peaks, arcs, and rings, which is described in Section IV.

IV. Results and Discussion

This section describes how to use the X-ray scattering data (Table I) as input to the several architectures for feature extraction, including time for training the models, and fingerprinting each data sample (Table II). Next, we show boxplots (Figure 4- 5), which summarize the comparisons among the different featurization schemes: notice that each interquartile (IQR) shows the metric dispersion over the 5 different AutoML classifiers. Finally, Table III highlights the best results in terms of six different metrics given each featurization scheme and the best AutoML classifier, as detailed below.

A. Dataset Creation and Augmentation

In order to compare the different classification approaches, we created models using X-ray scattering images as those shown in Figure 2. The database consists of 3,512 GISAXS, 746 GIWAXS, 746 WAXS, and 645 SAXS images, and we generate

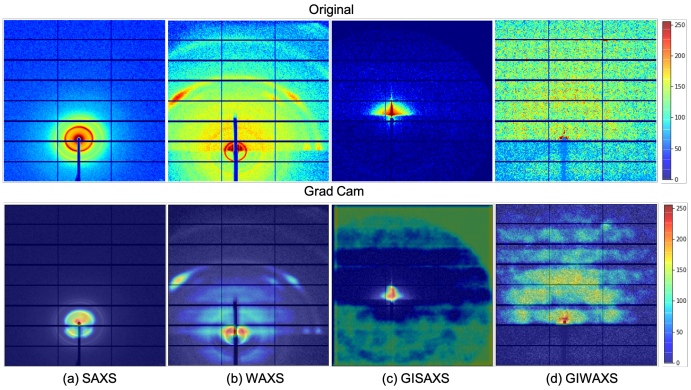


Fig. 3: Visualization of the class activation maps using Grad-CAM: SAXS, WAXS, GISAXS and GIWAXS. The first row shows the original images, their respective Grad-CAM counterparts in the second row.

TABLE I: Number of images for each class of generated datasets for training and testing of deep learning models: unbalanced (UB), balanced (BL) and balanced augmented (BLA).

Dataset name		WAXS	SAXS	GIWAXS	GISAXS
Train	UB	373	322	373	1756
	BL	322	322	322	322
	BLA	1288	1288	1288	1288
Test	UB	373	323	373	1756

three distinct datasets from this database: (a) the Unbalanced (UB) dataset, which randomly splits samples such that 50% of the dataset is used for training and the other 50% for testing (number of images are listed in the first and last rows in Table I); (b) the Balanced (BL) dataset, created to prevent training with uneven sizes, therefore selecting 322 images from each class, with the value 322 chosen because it evenly divides the maximum number of examples from the smallest class (second row in Table I); (c) the Balanced Augmented (BL), an augmented training dataset, which increases the number of training samples to 1,288 by rotating each image in the BL dataset (third row in Table I) with the angles: 0, 90, 180, and 270.

TABLE II: Processing time for training and fine-tuning architectures.

Architecture	Train time (sec)			Feature extraction time (sec)
	Unbalanced	Balanced	Balanced augmentated	
LeNet	1283	587	2758	0.010
VGG16	1587	744	2822	0.020
VGG19	1814	846	3272	0.021
Resnet	2097	984	3872	0.022
Xception	2967	2251	5520	0.034
Inception Resnet	3832	2932	10546	0.052

B. Model Evaluation

These three datasets, UB, BL, and BLA are input to featurization as in Section III-A, whose results are evaluated using 5 classifiers provided by the AutoML. Such evaluation considered cross-validation with k-folds ($k = 5$) to avoid biases on specific data subsets. Table II shows the processing time for training and fine-tuning during architectures evaluation.

Regarding the classic feature extraction methods, we computed the histogram-based features, which is a parameter-free approach, and the GLCM for distance values from 1 to 10, achieving the best results for distance equal to 1. The parameters used in the training of the LeNet are as follows: learning rate equal to 0.001, decay equal to 0.0001, number of epochs equal to 50, and batch size equal to 64. For fine-tuning of the deep models, the parameters are: learning rate = 0.0001, decay = 0.00001, number of epochs = 50 and batch size = 32. After the featurization using each of the CNN architectures in Section III-A, we assessed the AutoML performance in terms of six broadly-accepted metrics available in literature for our multi-class problem: accuracy (Acc), recall (Rec), precision (Prec), F-Score (FS) [9], Kappa coefficient (κ) [25], and the Matthews correlation coefficient (MCC) [30], the later three recommended as more appropriate metrics for data exhibiting an imbalanced representation of classes [15, 4].

Figure 4 presents the boxplots displaying the mean and variance of κ for the five deep architectures pre-trained with Imagenet, the histogram features, and GLCM. We focused on the κ coefficient, which is considered a vital metric to evaluate multiclass problems [25], and it allows easy interpretation, for example, a value less than 0 indicates non-agreement, between 0 and

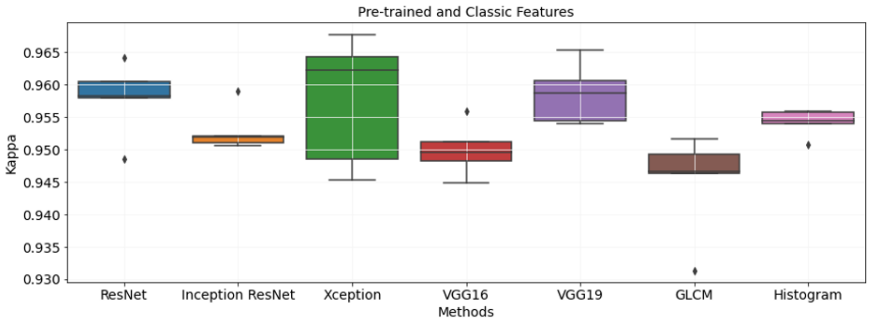
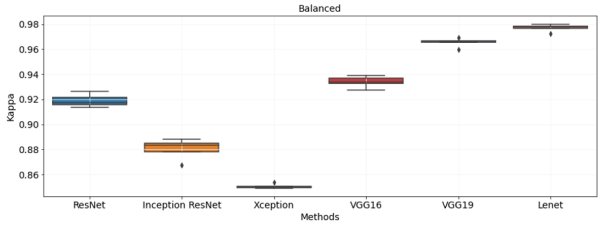


Fig. 4: Comparison among different CNN architectures in terms of Kappa. All CNNs were pre-trained using ImageNet, but without fine-tuning. GLCM, and Histogram-based features bypass training.

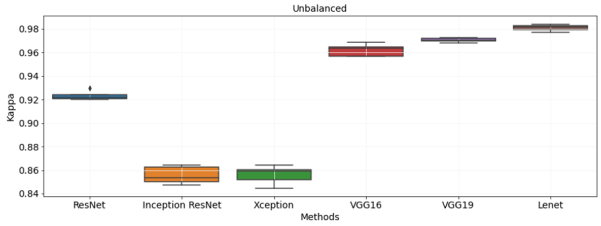
0.20 as light agreement, between 0.21 and 0.40 as reasonable, between 0.41 and 0.60 as moderate, between 0.61 and 0.80 as substantial and between 0.81 and 1 as almost perfect agreement.

The IQR from each boxplots in Figure 4- 5 emphasizes the mean average κ over the 5 different AutoML classifiers. As an example of how to calculate each IQR, notice the blue IQR in Figure 4, which is the result of a 3-step computation: (a) calculate accuracy for the multiclass problem using ResNet associated to one AutoML classifier using k-fold cross-validation, therefore obtaining the average κ; (b) repeat the previous step for each one of the AutoML classifiers; (c) calculate the mean and variance over the average κ for each classifier.

Next, Figure 5 presents the boxplots summarizing the results when using fine-tuning for the deep architectures, and LeNet trained from scratch; notice that the κ dispersion indicates that they outperformed the pre-trained architectures in Figure 4. This was somewhat expected since the pre-trained architectures without fine-tuning extract features that are less customized to the problem at target. One of the hypotheses for such a divergence is that the ImageNet database, used in the pre-trained case, lacks images that follow patterns similar enough to those present in our experiments.



(a) Balanced training set.



(b) Unbalanced training set.

(c) Balanced training set due to augmentation.

Fig. 5: Comparison among CNN architectures in terms of κ: all CNNs used fine-tuning but LeNet.

When evaluating the datasets used to train and fine-tune the CNNs, we observed similar κ coefficient across different architectures. Nevertheless, Figure 5 (a) and (c) indicate a lower dispersion in the boxplot when training with balanced sets. In addition, we show Table III that reports on the classification results following several metrics, including κ. Here, instead of calculating the mean across the 5 AutoML classifiers, we report metrics only for those with best performance given each featurization, in this case random forest (RF) and extra trees (ET). Overall, these results

V. Conclusion and Future Work

This research work uses deep learning and AutoML algorithms encapsulated in camSortXS to explore thousands of patterns from real experiments emcompassing 4 X-ray scattering techniques: SAXS, WAXS, GISAXS, and GIWAXS. We were able

TABLE III: Result for each featurization, given the best classifiers (Clf) using different training sets: unbalanced (UB), balanced (BL), and augmented (BLA) datasets, and the pre-trained (PT) models after fine-tuning hyperparameters (*); best classifiers are random forest (RF) and extra trees (ET)

CNN	Train	Clf	Acc	Rec	Prec	FS	κ	MCC
HF	UB	RF	97.51±0.30	95.71±0.51	97.53±0.30	97.51±0.30	95.60±0.52	95.60±0.52
LeNet	UB	RF	99.10±0.13	98.80±0.13	99.11±0.12	99.10±0.13	98.40±0.23	98.41±0.23
VGG16*	BLA	ET	99.00±0.06	98.48±0.14	99.01±0.06	99.00±0.06	98.24±0.10	98.24±0.10
VGG19*	BLA	ET	99.01±0.11	98.45±0.25	99.02±0.11	99.01±0.11	98.25±0.20	98.25±0.19
Xception	PT	ET	98.17±0.10	97.10±0.29	98.21±0.11	98.17±0.10	96.78±0.18	96.78±0.18
ResNet	PT	ET	97.97±0.41	97.05±0.73	98.00±0.39	97.97±0.41	96.42±0.71	96.43±0.71
Inception	PT	ET	97.68±0.42	96.36±0.70	97.72±0.42	97.68±0.42	95.90±0.75	95.91±0.75

to sort this new large labeled image collection of energy critical materials, and showed that different featurization options led to high mean average κ (over 0.9), which indicates almost perfect agreement across the 4 different classes and the different classifiers. By checking on 25 possible representations, classified using 5 different methods and evaluated using 7 metrics of performance, we achieve three main goals: (a) Characterize: Learn effective feature representations by transforming raw experimental data into compact signatures with the most relevant components; (b) Link: Compare, merge and integrate records from different experiments; (c) Screen: Autonomously sort experimental data which will inform upcoming inferential control of processes. Together, these efforts will exploit information from previously performed experiments to better guide next investigations. We expect that camSortXS will allow for expansion of algorithmic and software design to build recommendation systems toward the construction of operators that can efficiently steer experiments. In this way, many calibration processes for obtaining successful experiments at the beamlines could be automated. Such intelligent guidance of complex experiments will accelerate tasks such as the acquisition of experimental data, as well as part of manufacturing processes, such as film production.

REFERENCES

[1] M. Ali. *PyCaret: An open source, low-code machine learning library in Python*, July 2020. PyCaret version 2.1.

[2] Araujo, Silva, Medeiros, Parkinson, Hexemer, Carneiro, and Ushizima. Reverse image search for scientific data within and beyond the visible spectrum. *Expert Systems with Applications*, 109:35–48, Nov 2018.

[3] F. H. Araujo, R. R. Silva, D. M. Ushizima, M. T. Rezende, C. M. Carneiro, A. G. C. Bianchi, and F. N. Medeiros. Deep learning for cell image segmentation and ranking. *Computerized Medical Imaging and Graphics*, 72:13 – 21, 2019.

[4] M. Bekkar, H. K. Djemaa, and T. A. Alitouche. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, 3(10), 2013.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[6] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, oct 2001.

[7] Y.-F. Chen, P.-C. Huang, K.-C. Lin, H.-H. Lin, L.-E. Wang, C.-C. Cheng, T.-P. Chen, Y.-K. Chan, and J. Chiang. Semi-automatic segmentation and classification of pap smear cells. *IEEE J. Biomed. Health Inform.*, 18(1):94–108, Jan 2014.

[8] Z. Chen, Z. Xie, W. Zhang, and X. Xu. Resnet and model fusion for automatic spoofing detection. In *INTERSPEECH*, pages 102–106, 2017.

[9] N. Chinchor and B. M. Sundheim. Muc-5 evaluation metrics. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993.

[10] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[11] S. T. Chourou, A. Sarje, X. S. Li, E. R. Chan, and A. Hexemer. *HipGISAXS*: a high-performance computing code for simulating grazing-incidence X-ray scattering data. *Journal of Applied Crystallography*, 46(6):1781–1795, Dec 2013.

[12] A. N. Fountsop, J. L. Ebongue Kedieng Fendji, and M. Atemkeng. Deep learning models compression for agricultural plants. *Applied Sciences*, 10(19), 2020.

[13] G. Freychet, D. Kumar, R. J. Pandolfi, P. Naulleau, I. Cordova, P. Ercius, C. Song, J. Strzalka, and A. Hexemer. Estimation of line cross sections using critical-dimension grazing-incidence small-angle x-ray scattering. *Phys. Rev. Applied*, 12:044026, Oct 2019.

[14] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[15] Q. Gu, L. Zhu, and Z. Cai. Evaluation measures of the classification performance of imbalanced data sets. In *International symposium on intelligence computation and applications*, pages 461–471. Springer, 2009.

[16] A. Guinier. *X-ray diffraction in crystals, imperfect crystals, and amorphous bodies*. Courier Corporation, 1994.

[17] R. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans. Syst., Man, Cybern.*, SMC-3(6):610–621, Nov 1973.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] F. A. Heberle and G. Pabst. Complex biomembrane mimetics on the sub-nanometer scale. *Biophysical reviews*, 9(4):353–373, 2017.

[20] W. V. Herck, J. Fisher, and M. Ganeva. Deep learning for x-ray or neutron scattering under grazing-incidence: extraction of distributions. *Materials Research Express*, 2020.

[21] A. Hexemer and P. Müller-Buschbaum. Advanced grazing-incidence techniques for modern soft-matter materials analysis. *IUCrJ*, 2(1):106–125, Jan 2015.

[22] R. Jegadeeshwaran and V. Sugumaran. Fault diagnosis of automobile hydraulic brake system using statistical features and support vector machines. *Mechanical Systems and Signal Processing*, 52–53:436 – 446, 2015.

[23] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3149–3157, 2017.

[24] D. R. Kozub, K. Vakhshouri, L. M. Orme, C. Wang, A. Hexemer, and E. D. Gomez. Polymer crystallization of partially miscible polythiophene/fullerene mixtures controls morphology. *Macromolecules*, 44(14):5722–5726, 2011.

[25] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, 33(1):159–174, 1977.

[26] R. Lazzari, F. Leroy, and G. Renaud. Grazing-incidence small-angle x-ray scattering from dense packing of islands on surfaces: Development of distorted wave born approximation and correlation between particle sizes and spacing. *Physical Review B*, 76(12):125411, 2007.

[27] Q. Liu and X. Wang. Polyoxometalate clusters: Sub-nanometer building blocks for construction of advanced materials. *Matter*, 2(4):816–841, 2020.

[28] S. Liu, C. N. Melton, S. Venkatakrishnan, R. J. Pandolfi, G. Freychet, D. Kumar, H. Tang, A. Hexemer, and D. M. Ushizima. Convolutional neural networks for grazing incidence x-ray scattering patterns: thin film structure identification. *MRS Communications*, pages 1–7, 2019.

[29] R. Maree, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 34–40 vol. 1, 2005.

[30] B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[31] B. Ni, Y. Shi, and X. Wang. The sub-nanometer scale as a new focus in nanoscience. *Advanced Materials*, 30(43), 2018.

[32] G. Pabst, M. Rappolt, H. Amenitsch, and P. Laggner. Structural information from multilamellar liposomes at full hydration: full q-range fitting with high quality x-ray data. *Physical Review E*, 62(3):4000, 2000.

[33] G. Pospelov, W. Van Herck, J. Burle, J. M. Carmona Loaiza, C. Durniak, J. M. Fisher, M. Ganeva, D. Yurov, and J. Wuttke. *BornAgain*: software for simulating and fitting grazing-incidence small-angle scattering. *Journal of Applied Crystallography*, 53(1):262–276, Feb 2020.

[34] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: Unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6639–6649, Red Hook, NY, USA, 2018. Curran Associates Inc.

[35] G. Santoro and S. Yu. Grazing incidence small angle x-ray scattering as a tool for in- situ time-resolved studies. In *X-ray Scattering*. IntechOpen, 2017.

[36] N. Schwarz, S. Veseli, and D. Jarosz. Data management at the advanced photon source. *Synchrotron Radiation News*, 32(3), 5 2019.

[37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019.

[38] B. Shyam, K. H. Stone, R. Bassiri, M. M. Fejer, M. F. Toney, and A. Mehta. Measurement and modeling of short and medium range order in amorphous ta2o5 thin films. *Scientific Reports*, 6(1):2045–2322, 2016.

[39] S. Sinha, E. Sirota, S. Garoff, and H. Stanley. X-ray and neutron scattering from rough surfaces. *Physical Review B*, 38(4):2297, 1988.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[41] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *Computing Research Repository*, abs/1602.07261, 2016.

[42] Y. Tang. Deep learning using linear support vector machines. In *Proc. of International Conference on Machine Learning*, 2013.

[43] Y. Zhong and K. Xu. Contraction integral equation for three-dimensional electromagnetic inverse scattering problems. *Journal of Imaging*, 5(2):27, 2019.