Latent-Constrained Autoencoder for 4D-STEM Clustering

Martin Eriksen, Joaquin Oton, Mauricio Matta

Abstract

This project explores the use of unsupervised learning to distinguish crystalline and amorphous domains in 4D-STEM data from metallographic samples. An autoencoder is trained on experimental diffraction patterns to compress each 128×128 pattern into a 16-dimensional latent vector, corresponding to a reduction of approximately three orders of magnitude. These latent representations are analyzed using HDBSCAN for unsupervised clustering, enabling the detection of grain boundaries. However, the initial clustering results tend to produce an excessive number of fragmented clusters. To address this, we train an autoencoder that is insensitive to rotations in the diffraction patterns. Enforcing this physically motivated constraint leads to improved clustering results and more accurately captures the finite width of grain boundaries in the experimental data.

1 Data

The analysed 4D-STEM sample, which details are intentionally not included, has been acquired at the JEMCA microscopy facility at the ALBA synchrotron. While we have access to three different samples, only one sample is used in this work. The data has the dimension 256×256 in physical space and 128×128 in reciprocal space. Converting the raw binary format to an array was done through reverse engineering the format.

2 Methodology

The data analysis consists of two parts. For the first part we are training an auto-encoder. The input is normalized by subtracting the median pixel intensity, then dividing upon an estimate of the spread of the distribution. Since the data has some outliers, instead of the standard deviation, we use σ_{68}, which is half the distance between the 86th and 16th percentile. **The authors previously trained a standard autoencoder on 4D-STEM data as part of an exploratory study on a different sample.**

The auto-encoder architecture is a convolutional architecture. The encoder uses multiple blocks of consisting of convolutions and batch normalization. Then, two fully connected layers reduce the data to 16 features. In the decoder, these features are first upsampled with two linear layers, before a few blocks of transpose convolutional layers. Details of the architecture can be seen in the corresponding Github respository[1].

For training the networks, we choose the mean absolute different (MAE) loss

$$\mathcal{L}_{\text{Recon}} = \left\| D_\theta(E_\phi(x)) - x \right\|_1 \tag{1}$$

where D_θ and E_ϕ denotes the decoder and encoder network, respectively. In addition, we are including some clipping to be less sensitive to outliers.

A challenge with auto-encoders are understanding the information they have encoded. The typical encoder loss in Eq.(1) ensures the data are compressed into the features, but does not ensure each features have a specific meaning. In this specific case, we would like to construct a feature space such that

$$F = F^{\text{Material}} \times F^{\text{Orientation}} \tag{2}$$

where the encoded feature (F) can be split into a component related to the material (F^{Material}) and a part related to the orientation ($F^{\text{Orientation}}$). This could potentially be done by enforcing the decoder to reconstruct to a reference orientation. However, this would require defining such a position. Furthermore, even if not required for the reconstruction, the encoder could still be encoding the rotation in the features. While this might seem theoretical, we have observed this auto-encoder behaviour in other applications.

Instead our proposed solution is including a loss to enforce that the first N components in the features are invariant under a rotation of the input. This works by applying a rotation R_ϕ to an input x. The loss is then

$$\mathcal{L}_{\text{Rot}} = \text{mean}(|E_\phi(x)_{1:N} - E_\phi(R_\psi(x))_{1:N}|) \tag{3}$$

where the subscript $1:N$ indicate the comparison is only done with the first N features and these tests used $N = 10$. The combined loss is then the combination

$$\mathcal{L}(x) = \mathcal{L}_{\text{Recon}}(x) + \mathcal{L}_{\text{Recon}}(R_\psi(x)) + \lambda \mathcal{L}_{\text{Rot},\psi}(x). \tag{4}$$

where the reconstruction loss is used both for the original and rotated image. The rotation angle ψ is drawn uniformly on the circle, using the same value for all images in the batch. The loss weight (λ) is set to 50 to balance the contributions.

Based upon these features, we have been running `HDBScan` from `scikit-contrib` to unsupervised define classes. For the result we have used a minimum cluster size of 60.

[1] https://github.com/marberi/mic_hackathon

3 Results

We trained the network using a batch size of 32 and the Adam optimizer with a learning rate of 10^{-4}. The network was trained for 10 epoch, with epoch taking about 3.5 minutes on a single NVIDIA L40S. After training the networks, we evaluated the network on the same 4D-STEM datasets, using the same data as has been used for training. While this should not be done for supervised training, this can be done when working with an auto-encoder. When evaluating in batches of 128, the total evaluation time on a single GPU is 40 seconds.

Visually the input and auto-encoder reconstruction shows a very good agreement (figure not included). This demonstrates that the auto-encoder can learn to compress the 128×128 numbers into 16 parameters, representing a three order of magnitude compression. Also the auto-encoder has been removing noise in the pixel.

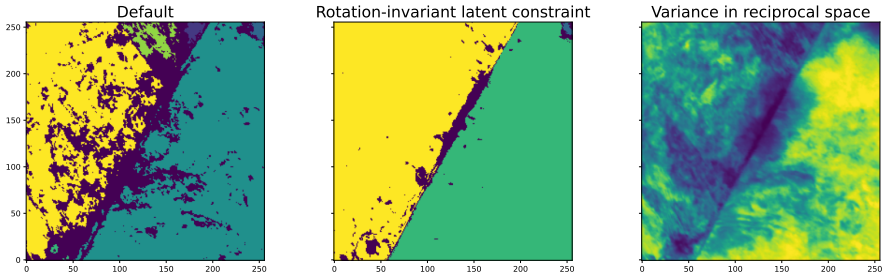

Figure 1: Clusters defined by running HDBScan over the auto-encoder features and variance in reciprocal space. *Left:* Using only the reconstruction loss. *Middle:* After adding the rotation invariance loss. *Right:* Variance over the reciprocal space.

Figure 1 shows the detected cluster using `HDBScan` algorithm using either the standard loss (left, Eq.1) or when also including the rotation-invariance loss (middle, Eq.4). For comparison we have also shown the variance over the reciprocal space. With the normal loss, the classes are more fragmented than after adding the additional loss, since it removes the rotation as a degree of freedom.

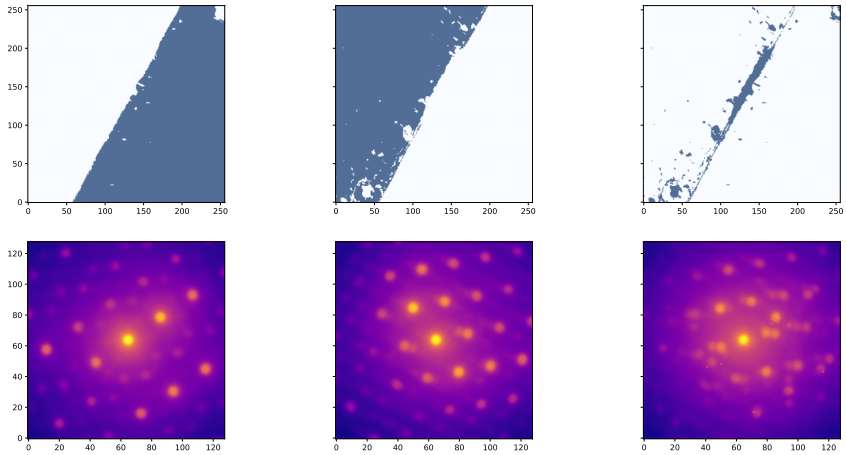

Figure 2: Three different clusters (*top row*) and their mean diffraction patterns (*bottom row*), using the auto-encoder trained with the rotation invariance loss and HDBScan.

Figure 2 validates the results further. On the top row is the clusters found with the additional loss, while the bottom rows shows the mean diffraction pattern in those regions. The algorithm is separating two different grains (left and middle panel), which are expected to be the same material, but has an out of plane rotation. On the right is the grain boundary, which exhibits a combination of both grain neighbours structure with a background component due to the amorphous or multi-crystal composition.

4 Conclusion

In this hackathon, we performed an unsupervised clustering of 4D-STEM data using an auto-encoder. We introduced an additional loss in auto-encoder to remove the effect of rotations in the diffraction patterns. Removing this degree of freedom resulted in clearer defined classes. For future work, we plan better understanding additional degrees of freedom using simulations and analyze the other samples we have available.