nature machine intelligence

Article

https://doi.org/10.1038/s42256-022-00555-8

AtomAI framework for deep learning analysis of image and spectroscopy data in electron and scanning probe microscopy

In the format provided by the authors and unedited

Supplementary Note 1:

Complementary to the data analysis methods are the direct modeling methods, including QSTEM,[1] μSTEM,[2] Dr. Probe,[3] MULTEM,[4] STEMsalabim,[5] abTEM,[6] Prismatic,[7] and others. QSTEM utilizes the multislice algorithm to perform STEM simulations, as well as TEM and convergent beam electron diffraction (CBED) calculations. QSTEM features frozen phonon approximation for the thermal diffuse scattering simulations, the ability to create extremely thin slices, the correct treatment of chromatic aberration, and the efficient implementation of electron source size effects, amongh other things. QSTEM is available in python as pyqstem in addition to its GUI-based software. μSTEM is another software package for simulating STEM images. The package features a command-line input and has developed capabilities beyond the simulation of STEM images. Version 5.0 of μSTEM is capable of simulations in the absorptive model, as well as modeling elemental mapping using EELS and EDX analyses based on inner-shell ionization. MULTEM is a C++/MATLAB implementation which similarly performs multislice simulations of STEM images by treating as amorphous material and using the frozen phonon approximation. The accuracy is ensured by including higher order expansion of the multislice solution of the high energy Schrödinger equation and the correct subslicing of the three-dimensional potential and the top-bottom surfaces. The speed of calculation and scalability is improved by parallelization using GPUs and calculating the generalized transmission function of each slice when needed. STEMsalabim is a multislice simulation implementation which aims to improve the parallelization across HPC clusters and scalability. To achieve parallelization efficiency, a combination of the message passing interface (MPI) and (POSIX) threads was used while maintaining a small memory footprint. As with most STEM simulation package, STEMsalabim supports TDS via the frozen lattice approximation, as well as chromatic abberations simulations and the implementation of lens aberrations up to fifth order. abTEM simulates TEM images by generating *ab initio* electrostatic potentials when describing the core electrons by projector functions. This results in a comparable accuracy with earlier all-electron calculations at a much lower computational cost. abTEM is implemented in Python (with application-specific GUIs) and integrates directly with the Atomic Simulation Environment (ASE) for setting up atomistics models and GPAW for calculating electrostatic potentials. It also features GPU-acceleration via the plane-wave reciprocal-space interpolated scattering matrix (PRISM) algorithm for STEM simulations. Prismatic simulates STEM images and is also based on PRISM. Using an asynchronous, multi-GPU streaming framework, Prismatic significantly outperforms traditional multislice algorithms in terms of computation speed (as high as 1000 ´ with a 4-GPU machine). Prismatic is available as an oper-source C++/CUDA package with a GUI, and is also available in Python via PyPrismatic and as a command line interface.

Supplementary Note 2:

Studying any physical, chemical, or biological processes can expand over multiple length and time scales of many orders of magnitude. These may range from electronic (few Å), atomic or nanoscale (1-300 nm), microscale (0.1 – 15 m), mesoscale (1 - 10 mm) to macro or structural scale. While specific phenomena may occur at a particular length or time scale and system size, bridging the knowledge acquired across these ranges, leading to multiscale studies have been of interest for a long time in the scientific community. Moreover, development and availability of more computational capabilities including accessible CPU/GPUs, efficient algorithms and corresponding implementations have significantly advanced the field of physical simulations over the past couple of decades. While quantum-mechanical calculations can accurately describe

electronic properties driving the energetics and dynamics of materials, solving partial differential equations to model materials behavior at a larger continuum scale is also possible. Here, we include a brief (not inclusive) of some of the open-source packages to perform such simulations.

We note that the parent codes of the above-mentioned packages and others could be developed using different programming languages such as C++, Fortran, Java, Python. But that does not necessarily limit users to construct suitable interfaces using an environment of choice. This becomes very important to not only analyze the data generated by simulations or experiments individually but also build a bridge to learn from both nuances at the same time. There exist several such open-source post-processing codes that can be utilized to analyze the results generated from simulations performed at varied scales. Few of the popular frameworks (not inclusive) including visualization tools are p4vasp, vasptools, OVITO,[8] Atomsk,[9] Packmol,[10] Avogadro,[11] ASE,[12, 13] LIGGHTS,[14] ParaView,[15, 16] PyMol,[17] VMD,[18] Vesta[19] and many more. The applicability of these interface-type frameworks also is dependent on the flexibility and extendibility of those to process and interact with outputs generated from simulations performed using different packages. This may pose additional challenges where information from experiments, such as microscopic images, needs to be analyzed and further processed to perform reasonable simulations such that a comprehensive understanding of a material or material property can be developed using both theoretical and experimental knowledge. With recent development in data analytics and machine learning capabilities useful for materials research, several databases have also become popular allowing to perform simulations alongside data-driven studies. A non-exhaustive list of such databases include Materials Project,[20] AFLOW,[21, 22] OQMD,[23, 24] NOMAD.[25]

Supplementary Note 3: Additional code examples

Example of using low-level API to train a U-Net model:

```
import atomai as aoi
# Initialize model
model = aoi.nets.UNnet()
# Initialize trainer
t = aoi.trainers.BaseTrainer()
# Pass model to trainer
t.set_model(model, nb_classes=1)
# Compile trainer
t.compile_trainer(
       (images, labels, images_test, labels_test),
       loss="ce", training_cycles=500, swa=True)
# Train
t.fit()
# Save trained model
t.save_model("my_model")
```

Load the trained model and obtain a prediction on new data:

```
import atomai as aoi
# Load model
trained_model = aoi.load_model("my_model.pt")
# Initialize predictor
p = aoi.predictors.SegPredictor(trained_model)
# Apply to new data
nn_output, cords = p.run(expdata)
```

Example of training ensemble of models:

```
# Initialize ensemble trainer
etrainer = aoi.trainers.EnsembleTrainer("Unet", nb_classes=3)
# Compile
trainer.compile_ensemble_trainer(training_cycles=50, swa=True)
# Train ensemble of 12 models
smodel, ensemble = etrainer.train_ensemble_from_scratch(
       images, labels, images_test, labels_test, n_models=10)
```

Use trained ensemble to obtain prediction on new data:

```
p = aoi.predictors.EnsemblePredictor(smodel, ensemble, nb_classes=3)
nn_out_mean, nn_out_var = p.predict(expdata)
```

Supplementary Note 4: Case Studies

Here we illustrate several case studies using AtomAI. For more examples, please refer to the GitHub page of the project.[26]

4.A. Semantic segmentation of the atom-resolved images.

Semantic segmentation of atomically resolved microscopic images involves classifying every pixel in the image as belonging to specific atom and/or defect classes. This is different compared to classifying natural images where we categorize one image as a whole. Because this is a supervised method, it requires training data where the atoms and/or defects are labelled. Once trained, a *Segmentor* model can be used to predict the position of the atoms and/or defects in previously unseen data. Below we show an example of how one can train a Segmentor model using labeled experimental images and apply the trained model to data from a different experiment alongside performing multivariate statistical analysis on the semantically segmented output.

To prepare the training data, we used a single labeled experimental STEM image from Sm-doped $BiFeO_3$ with the resolution of 3000 × 3000 pixels containing ~20,000 atomic unit cells.[27] The image is part of a publicly available dataset.[28] The corresponding mask (*aka* ground truth) was generated using the atomic coordinates from a Gaussian fit. Here, there are three different classes in the labelled data corresponding to atomic columns (hereafter referred to simply as "atoms") in A-lattice (center atom) and B-lattice (4 corner atoms). About 2000 patches of image-masks pairs with the size of 256 × 256 pixels were cropped and further "augmented" by applying different

levels of noise, blurring, as well as changing scale (zooming-in) and 90°-rotations. The purpose of such augmentation is to account for variations in imaging conditions between different experiments. Figure S1 shows five different augmented images and corresponding ground truths.

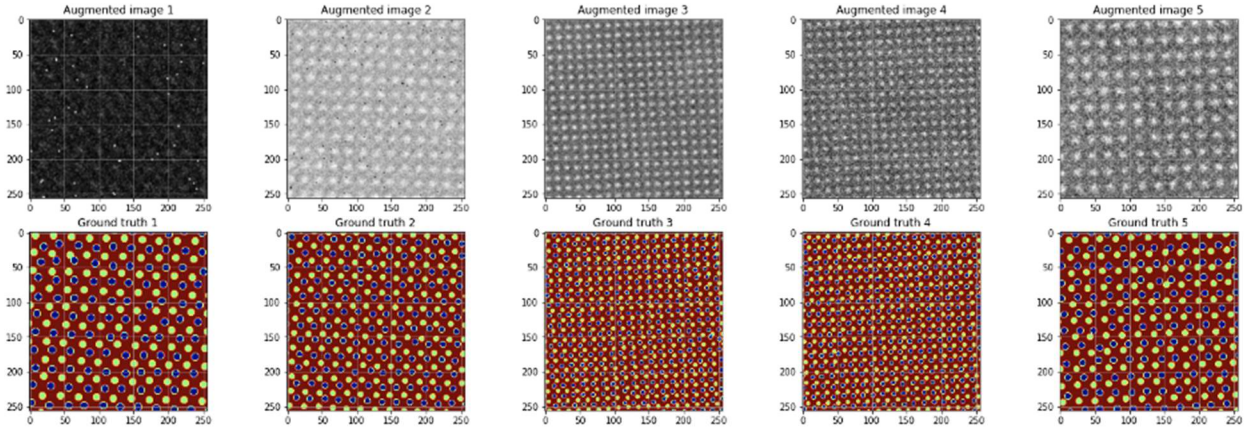

Figure S1. The top row represents the input to the Segmentor model and the bottom row represents the ground truth to which the output generated by the model will be compared during the model training.

The default Segmentor model is based on the U-Net neural network but one can also choose a different model or define a custom fully convolutional neural network. The Segmentor's raw output represents a set of well-defined (semantically segmented) blobs corresponding to different atomic types on a uniform background. For the trained model, the centers of the mass of the predicted blobs correspond to the atomic centers.

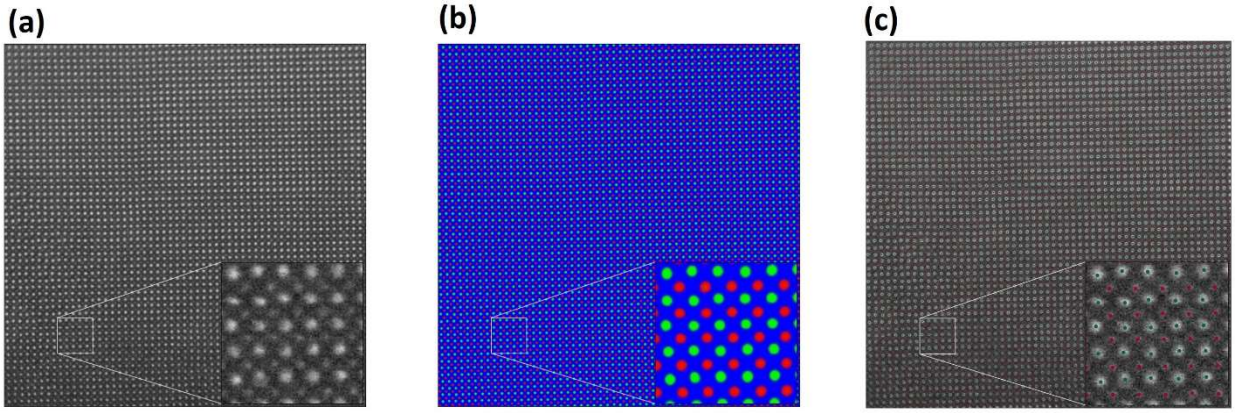

Figure S2. (a) STEM image from a similar material obtained in a different experiment (i.e., the Segmentor has not seen this image before). (b, c) Prediction of the trained Segmentor model: (b) semantically segmented raw output and (c) refined atomic coordinate (see Figure S3 for a comparison between refined and non-refined prediction). Note that the model is robust with respect to variations in sample thickness.

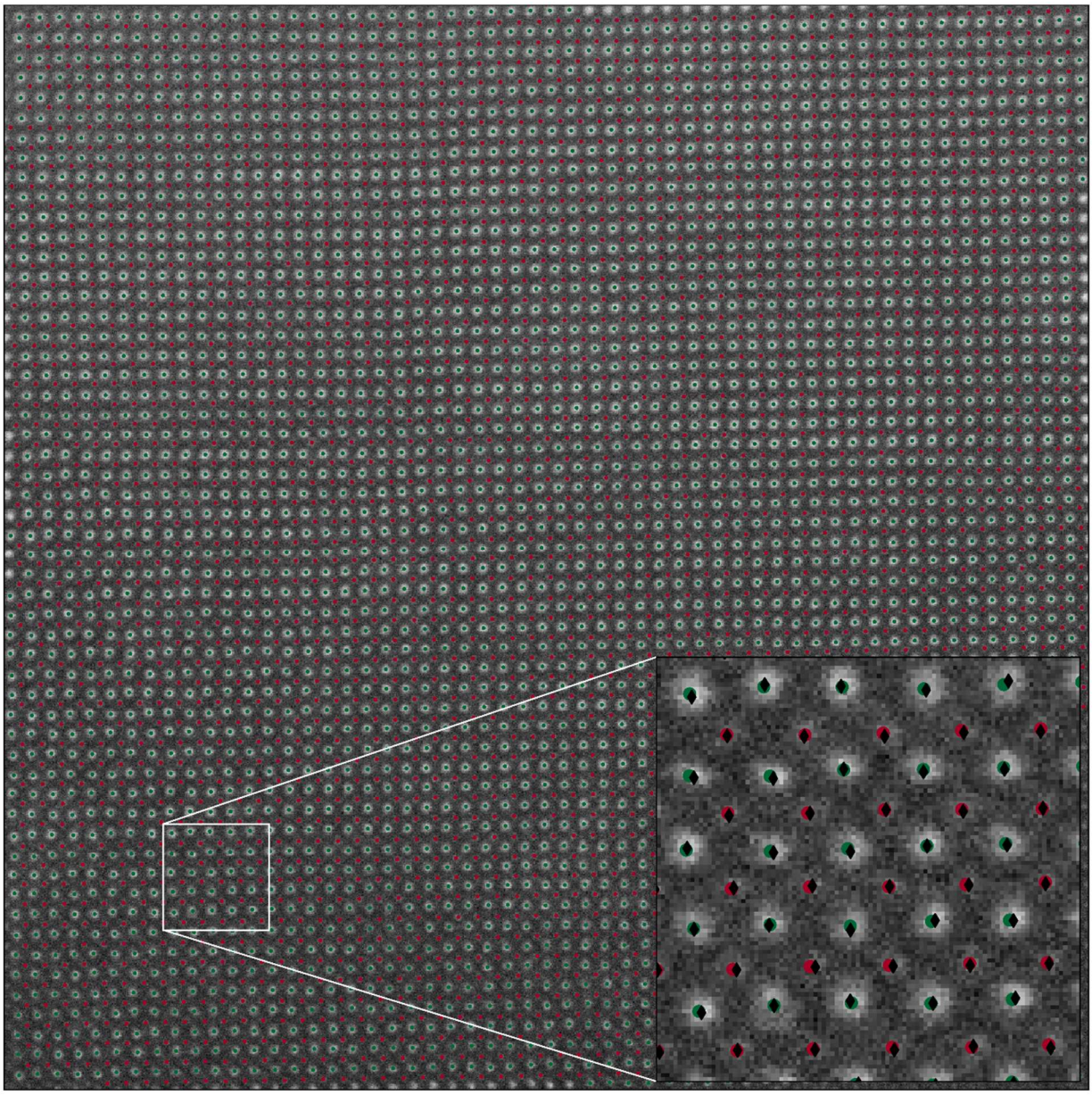

Figure S3. The prediction of AtomAI's *Segmentor* with (black diamonds) and without (green and red scatter points) Gaussian peak refinement. Implementation wise, the refined prediction is obtained by running `model.predict(expdata, refine=True)`. For this image, a prediction with a trained model without refinement takes a fraction of a second on a modern GPU, whereas the prediction with refinement may take up to 1 minute.

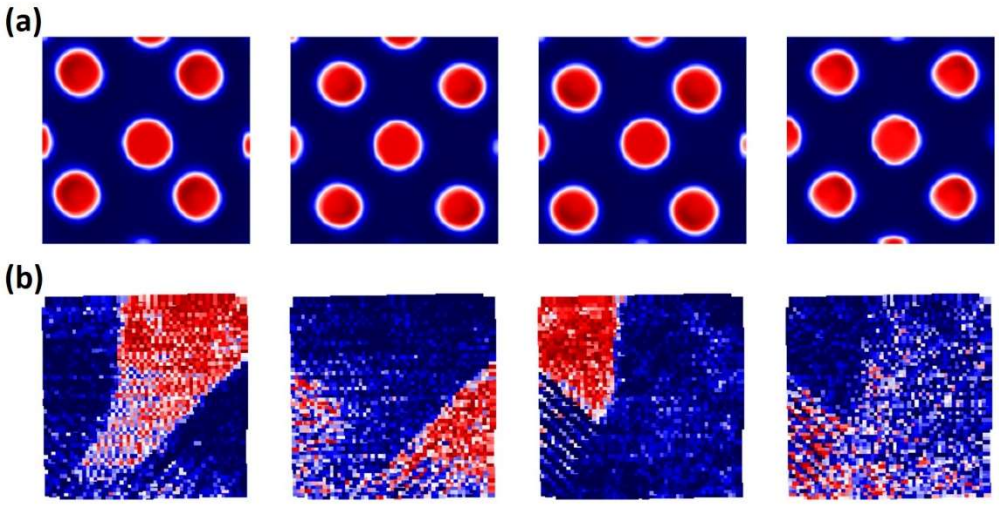

Figure S4. The results from the patch-based NMF analysis with the displacement components and associated loading maps shown in (a) and (b), respectively. Similar analysis was reported earlier in [29] using our AICrystallographer package (the AtomAI's predecessor).

The prediction of the trained Segmentor for a different experimental image (La-doped $BiFeO_3$) obtained in a different experiment[29] is shown in Figure S2 (b,c) where it was able to remove the experimental noise and separate two sublattices into different classes (red and green in Fig. S2b, c). We note that in this case the input image had a size of 1024 × 1024 pixels while our network was trained only using images of 256 × 256 pixels. This underscores a very important aspect of the fully convolutional neural networks, namely, that they are not sensitive to the size of input image as long as it can be divided by 2^n, where n is a number of max-pooling layers in the network (here it is equal to 3). However, there is always some optimal pixel-to-angstrom ratio (or, roughly, number of pixels per atom/defect/particle) for which a network will generate the best results. We note that although in this example the Segmentor was trained essentially on a single image, it is generally better to use a large(r) and diverse set of images.

Once we have all atomic coordinates and semantically segmented images, we can perform various forms of multivariate analysis on local image descriptors formed by extracting patches of a fixed size centered around one type of the lattice sites. The results of applying NMF to the Segmentor output are shown in Figure S4. The four NMF components provide clear pictures of atomic displacements whereas the corresponding loading maps show characteristics of the domain structure, all found in an unsupervised manner. We note that the NMF components are not required to be physical and generally one has to exercise caution when it comes to their interpretation. Strictly speaking, it is an exploratory analysis which can point to regions of interest for more in-depth exploration. In this case, the detailed inspection of these regions confirmed the NMF predictions.

4.B. Variational Autoencoders (VAE): analysis of structural order parameters

An autoencoder generally refers to a special class of neural networks where the original data set is compressed to a small number of continuous latent variables and is then expanded back to the original data set. In the process, the network learns how to optimally describe the data in terms of the latent variables. This allows the autoencoder to discover the optimal representation(s), while also rejecting the noise present in the data. The variational autoencoder (VAE) builds upon this concept by making the reconstruction process probabilistic. In this case, the latent variables are drawn from a certain (typically standard Gaussian) distribution, and the training process seeks to optimize both the reconstruction loss and the Kullback-Leibler divergence between the encoded distribution and the chosen prior. The advantages of the VAE over classical autoencoders include enforcement of structure and smoothness in the latent space, the ability to learn both continuous and discrete distributions, and state-of-the-art performance for the weakly- and semi-supervised learning (when only a small part of data is labeled). In fact, VAEs have been actively used for discovering trends in various high-dimensional datasets.[30-32] Examples of such trends include emotional expressions in facial databases and writing styles in hand-written digits databases.

AtomAI expands the classical VAE architecture to disentangle both discrete and continuous representations of the data while accounting for translational and rotational invariances. Recently, the VAE module in AtomAI has been used to identify an order parameter in disordered system from atom-resolved movies[33] and to (re)discover molecular building blocks and chemical reaction pathways directly from electron microscopy data in an unsupervised fashion.[34] It has also been used to probe atomic-scale symmetry breaking from CBED patterns.[35]

Here we briefly illustrate the application of the rotationally invariant VAE (*r*VAE) to an atomic force microscopy movie of protein self-organization[36] as shown schematically in Figure S6. We start by applying a pre-trained Segmentor to the AFM movie frames. The Segmentor was trained on just a few labeled images from a "stable" phase of the process characterized by relatively low noise and absence of large scars along the fast scan direction. The Segmentor output then serves as the input (and output) for the *r*VAE which automatically separates the orientation of the particles from other degrees of freedom thereby allowing to encode (and analyze) a rich spectrum of local transitions into the latent space. Note that in classical VAE the rotational factor of variation gets admixed into the factors associated with physical mechanisms preventing from learning a proper disentangled representation for any physically meaningful number of latent dimensions.

4.C. Predictability of localized functional responses

The AtomAI's ImSpec models can be trained to predict localized functional responses in the form of 1D spectra from structural 2D images. The method is based on the idea that local structures and functional phenomena are (cor)related and the relationship is parsimonious, that is, it can be explained by a relatively small number of (latent) mechanisms. The ImSpec consists of encoder in the form of convolutional neural network that that maps the 2D images into the low-dimensional latent vector and the decoder in the form of 1D convolutional neural network that reconstructs the spectra from this latent representation. Note that this is different from the (variational) autoencoder approach where inputs and outputs are the same. The correlative structure-property relationships can be analyzed by projecting the learned latent vector representation (for a particular dataset) to the spectral and image domains. As with all supervised ML methods, the caveat is that ImSpec shows good/reliable predictive performance only on the data obtained under similar experimental conditions (see however the next section on some ways of addressing this limitation).

4.D. Ensemble Learning-Iterative Training (ELIT)

In general, the performance of deep neural networks varies significantly based on the choice of training data and experimental noise, bias, and data acquisition parameters. One way to (partially) account for these issues is to utilize Ensemble Learning (EL) which combines the predictions of multiple models. Here, multiple learners are trained to solve a given problem as compared to commonly constructed ML models that depend on only a single learner. The utilization of a combination of learners allows for a flexibility in the search processes of an algorithm (in this case, a deep learning model), and selection of a broad hypothesis space. Hence, the generalization capability of EL is superior compared to that of a single model, leading to higher prediction accuracies on new data. This approach has already been widely used in various applications[37, 38] ranging from character recognition, text categorization, face recognition to computer-aided medical diagnosis and gene expression analysis.

One such application of interest is feature finding in image data. For image datasets where the tens of thousands of samples with large variabilities are commonly available (such as MNIST and CIFAR), the common deep learning classification strategies work very well. In such cases, the training and validation data are drawn from the (more or less) *i.i.d.* data making the feature-based classifications perform with reasonable accuracies.

In contrast, for datasets consisting of atomically resolved STEM or STM images, the presence of almost identical atoms or objects is common, even when the imaging or simulation conditions are varied. The disparities in detected features for many experimental or atomistic simulation conditions are often negligible which makes the task of feature finding much more challenging. As a result, deep neural networks trained to account for a large variety of experimental conditions are not always good enough to recognize subtle distinctions in atomic features present in a particular experimental dataset. In addition, the applications of deep neural networks to such problems must be able to deal with out-of-distribution effects by rapidly adapting to changes in the imaging conditions.

Ensemble learning combined with iterative training (ELIT) offers a strategy to surpass the above-mentioned challenges. The EL part of the workflow allows for selection of artifact-free features and pixel-wise uncertainty maps by combining multiple networks. The IT part retrains the ensemble networks with already detected features, focusing its attention on features present in the (heavily degenerate) data and thus increasing the detection limit of the network on the dataset(s) of interest.

We have recently successfully applied a ELIT framework to multiple datasets, including the dynamic STEM data from graphene with impurities and static STEM data from NiO-LSMO.[39] In both cases, the learning begins with training the models on data generated by simulations and consequently applying it to the experimental images followed by iterative reevaluations and modifications to the training sets to obtain a model that can successfully identify all atoms and impurities, or defects present in the system. The ELIT framework works reasonably well for feature finding in new experimental data and allows a generation of meaningful uncertainty maps on the level of individual pixels. This method also allows for rapid correction for out of distribution effects during automated experiments where the imaging conditions change compared to the training set.

4.E Connection to ab-initio simulations

As described in the main text, AtomAI allows using deep fully convolutional neural networks (Segmentor models) for finding atoms and/or other relevant features (particles, defects, etc.) from microscopic images. The output contains a list of coordinates with corresponding classes for all detected atoms, which can easily be converted to Cartesian or fractional coordinates as necessary. We note that one of the key steps before performing any atomistic simulation is to build the system's "cell". It can be of different types such as bulk conventional unit cell, supercell or surface depending on the type of simulations and material properties of interest. Hence, once we obtain the coordinates corresponding to all atoms present in the system from the Segmentor model, our next step is to construct one of these cells. This is done using the AtomAI's `ase_obj_basic` function. Once these cells are constructed, the next task is to optimize the geometry. One can obtain the relaxed geometry using quasi-Newton algorithms or first-principles methods before proceeding to perform other advanced simulations such as computing the electronic properties.

Within the quasi-Newton methods as implemented in ASE, the algorithms decide the positions of atoms at every iteration based on the forces and second derivative of the total energy of the atoms. This may not work well in all cases, where electronic interactions can play a major role to acquire relaxed geometry. In such cases, we can employ the objects obtained using AtomAI utility functions in electronic structure codes to first relax atom positions, cell shape, and volume such that forces and stresses between all atoms are negligible. Consequently, atomistic simulations with various classical potentials such Lennard-Jones using built-in ASE calculators or other ab-initio DFT (using pseudopotentials) or MD codes can be employed to evaluate material properties.

Supplementary Note 5: From feature extraction to generative models

In any type of solids, regardless of whether the long-range periodicity[40] is present, features extracted from microscopic observations[41] are useful to appropriately depict the material, even in the non-observed regions. There are few directions that are interesting to further elaborate such views. Generative models based on sample generations representing collation of microstructures can potentially serve as an alternative of describing the full solid structure. These models rely on selection of atomic configurations that will be enough to produce an average description of the material via statistical means. In addition, there are physical generative models that provide us with the scope to include prior physical knowledge of the systems from the observational data and associated uncertainties. For example, the latent representations as uncovered by the VAE can also serve as a generative model, demonstrating physical behavior of parts of material structure, variation in those with respect to the changes in the environment. For solids where the conventional deterministic rules such as rotation, tilting of building blocks (unit cells) of materials break, reconstruction of the Hamiltonian[42] can become the objective. Parameters and physical variables from mesoscopic models can be directly matched with local imaging data, to achieve such goals. Optimization schemes such as Bayesian optimization in parametric, semi-parametric and non-parametric regimes are employed to explore and exploit wide parameter space in such scenarios. Such approaches are also advantageous for phase-field models that rely on solving partial different equations or assuming the inversion[43-47] approach to describe complex physical behavior and processes, applicable mainly for atomically resolved[48-50] systems.

Supplementary Note 6: Tips for model selection and training data preparation.

Application of a deep neural network trained on simulated data or on data from previous experiments to a new experiment performed under different conditions often leads to the unreliable performance. The reason for this is that a neural network cannot generalize to data outside the domain of training examples, the behavior generally referred to as the out-of-distribution shift. These out of distribution effects have been broadly recognized in the deep learning community and present a significant barrier for practical applications of the deep learning methods in medicine or fully autonomous driving.[51-58] Hence, it is critical to ensure that i) the new data is acquired using the same range of acquisition/imaging parameters as used for model training; ii) the prediction on new data comes with robust uncertainty estimates allowing to detect the out-of-distribution shift (on which the uncertainty is high signaling that the prediction is not reliable). The carefully crafted data augmentation pipelines can partially mitigate the out-of-distribution issues by accounting for a variation in instrumental parameters during the acquisition (different sample orientation, magnification, noise level, etc.). AtomAI provides utility functions for creating such data augmentation pipelines that can be used either during the training ("on-the-fly") or prior to training. For the uncertainty estimates, we recommend using the ensemble training available from AtomAI.

Supplementary Table I: A short list of open-source packages for physical simulations at different length scales.

Package	Scale	Notes
Materials Studio[59]	multiscale	Structure generator, quantum-mechanical, semi-empirical, classical simulations
CP2K[60]	Multiscale	Atomistic simulations, molecular dynamics, metadynamics, Monte Carlo, Ehrenfest dynamics, vibrational analysis, core level spectroscopy, energy minimization
MOOSE[61]	Meso-macroscale	Finite element framework, phase-field modeling
CALCULIX[62]	Meso-macroscale	Finite element framework
WARP3D[63]	Macroscale	3D Nonlinear analysis of solids
ELMER[64]	Macroscale	Multi-physics finite element framework
LAMMPS[65]	Nanoscale	Molecular dynamics simulations
ASE[12, 13]	Electronic - nanoscale	Structure generator, Atomistic simulations
SIESTA[66, 67]	Electronic	Density functional theory (DFT) simulations using atomic orbitals basis functions
Quantum Espresso[68-70]	Electronic	DFT simulations using plane waves basis functions

References:

1. C. T. Koch, Ph. D. Thesis (2002).
2. L. J. Allen, A. J. D'Alfonso and S. D. Findlay, Ultramicroscopy **151**, 11-22 (2015).
3. J. Barthel, Ultramicroscopy **193**, 1-11 (2018).
4. I. Lobato and D. Van Dyck, Ultramicroscopy **156**, 9-17 (2015).
5. J. O. Oelerich, L. Duschek, J. Belz, A. Beyer, S. D. Baranovskii and K. Volz, Ultramicroscopy **177**, 91-96 (2017).
6. J. Madsen and T. Susi, Microsc. microanal. **26** (S2), 448-450 (2020).
7. A. Pryor, C. Ophus and J. Miao, Adv. Struct. Chem. Imag. **3** (1), 15 (2017).
8. A. Stukowski, Modelling Simul. Mater. Sci. Eng. **18**, 015012 (2010).
9. P. Hirel, Comput. Phys, Comm. **197**, 212-219 (2015).
10. R. A. L. Martínez, E. G. Birgin, J. M. Martínez. , Journal of Computational Chemistry **30(13)**, 2157-2164 (2009).
11. D. E. C. M.D. Hanwell, D.C. Lonie, T. Vandermeersch, E. Zurek and G. R. Hutchison Journal of Cheminformatics **4(1)**, 1-17 (2012).
12. A. H. L. e. al., Journal of Physics: Condensed Matter **29** (27), 273002 (2017).
13. T. Y. David B. Lingerfelt, Anthony Yoshimura, Panchapakesan Ganesh, Jacek Jakowski, Bobby G. Sumpter, Nano Letters **21**, 236-242 (2021).
14. C. G. C. Kloss, A. Hager, S. Amberger, S. Pirker, Progress in Computational Fluid Dynamics **12**, 140-152 (2012).
15. B. G. J. Ahrens, C. Law, (Elsevier, 2015).
16. A. Utkarsh, (Kitware, 2015).
17. W. L. DeLano, CCP4 Newsletter on protein crystallography **40**, 82-92 (2002).
18. A. D. W. Humphrey, and K. Schulten, J. Molec. Graphics **14**, 33-38 (1996).
19. K. M. a. F. Izumi, Journal of applied crystallography **44(6)**, 1272-1276 (2011).
20. S. P. O. A. Jain, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson, APL Materials **1** (1) (2013).
21. D. H. M. J. Mehl, C. Toher, O. Levy, R. M. Hanson, G. L. W. Hart, and S. Curtarolo, Comp. Mat. Sci. **136**, S1-S828 (2017).
22. M. J. M. D. Hicks, E. Gossett, C. Toher, O. Levy, R. M. Hanson, G. L. W. Hart, and S. Curtarolo, Comp. Mat. Sci. **161**, S1-S1011 (2019).
23. S. K. J. E. Saal, M. Aykol, B. Meredig, and C. Wolverton, JOM **65**, 1501-1509 (2013).
24. S. J. E. S. S. Kirklin, B. Meredig, B, A. Thompson, J.W. Doak, M. Aykol, S. Rühl and C. Wolverton. , npj Computational Materials **1** (2015).
25. C. D. a. M. Scheffler, J. Phys. Mater **2**, 036001 (2019).
26. M. Ziatdinov, GitHub repository, https://github.com/pycroscopy/atomai (2020).
27. M. Ziatdinov, N. Creange, X. Zhang, A. Morozovska, E. Eliseev, R. K. Vasudevan, I. Takeuchi, C. Nelson and S. V. Kalinin, Appl. Phys. Rev. **8** (1), 011403 (2021).
28. C. Nelson, A. Ghosh, M. Ziatdinov and S. Kalinin V, (Zenodo, 2021).
29. M. Ziatdinov, C. Nelson, R. K. Vasudevan, D. Y. Chen and S. V. Kalinin, Appl. Phys. Lett. **115** (5), 5 (2019).
30. D. Celis and M. Rao, in *Proceedings of the 1st International Workshop on Fairness, Accountability, and Transparency in MultiMedia* (Association for Computing Machinery, Nice, France, 2019), pp. 26–32.

31. T. Yamada, M. Hosoe, K. Kato and K. Yamamoto, presented at the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017 (unpublished).
32. C. Doersch, **2021**, arXiv:1606.05908. arXiv.org e-Print archive. https://arxiv.org/abs/1606.05908.
33. S. V. Kalinin, O. Dyck, S. Jesse and M. Ziatdinov, Science Advances **7** (17), eabd5084 (2021).
34. S. V. Kalinin, O. Dyck, A. Ghosh, B. G. Sumpter and M. Ziatdinov, arXiv preprint arXiv:2010.09196 (2020).
35. M. P. Oxley, M. Ziatdinov, O. Dyck, A. R. Lupini, R. Vasudevan and S. V. Kalinin, npj Computational Materials **7** (1), 65 (2021).
36. S. V. Kalinin, S. Zhang, M. Valleti, H. Pyles, D. Baker, J. J. De Yoreo and M. Ziatdinov, ACS Nano **15** (4), 6471-6480 (2021).
37. G. S. a. J. Elder, *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. (Morgan and Claypool Publishers, 2010).
38. C. Z. a. Y. Ma, *Ensemble machine learning: methods and applications*. (Springer Science & Business Media, 2012).
39. A. Ghosh, B. G. Sumpter, O. Dyck, S. V. Kalinin and M. Ziatdinov, arXiv preprint arXiv:2101.08449 (2021).
40. D. A. K. A. L. Goodwin, Nature **521**, 303-309 (2015).
41. M. Ziatdinov, O. Dyck, X. Li, B. G. Sumpter, S. Jesse, R. K. Vasudevan and S. V. Kalinin, Sci. Adv. **5** (9), 9 (2019).
42. Q. Li, C. T. Nelson, S. L. Hsu, A. R. Damodaran, L. L. Li, A. K. Yadav, M. McCarter, L. W. Martin, R. Ramesh and S. V. Kalinin, Nat. Commun. **8** (2017).
43. M. Z. B. Wei Li, Juner Zhu, Computer Methods in applied mechanics and engineering **383**, 113933 (2021).
44. J. S. a. M. Z. B. Surya Effendy, Journal of The Electrochemical Society **167**, 106508 (2020).
45. R. D. B. Hongbo Zhao , Martin Z. Bazant, Journal of Computational Physics **436**, 110279 (2021).
46. R. G. Anton L. Cottrill, Flavien Fremy, Johann Meulemans, Mackenzie R. Sheldon, Martin Z. Bazant, International Journal of Heat and Mass Transfer **163**, 120445 (2020).
47. V. B. John Newman, in *The Newman Lectures on Mathematics* (Jenny Stanford Publishing, Boca Raton, 2018).
48. L. Vlcek, S. Yang, M. Ziatdinov, S. Kalinin and R. Vasudevan, Microscopy and Microanalysis **25** (S2), 130-131 (2019).
49. S. Mani Prudhvi Valleti, L. Vlcek, R. K. Vasudevan and S. V. Kalinin, in *arXiv e-prints* (2019).
50. L. Vlcek, A. Maksov, M. Pan, R. K. Vasudevan and S. V. Kalinin, ACS Nano **11** (10), 10313-10320 (2017).
51. D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman and D. Mané, arXiv preprint arXiv:1606.06565 (2016).
52. A. Nguyen, J. Yosinski and J. Clune, Proceedings of the IEEE conference on computer vision and pattern recognition, 427-436 (2015).
53. I. Gulrajani and D. Lopez-Paz, arXiv preprint arXiv:2007.01434 (2020).
54. Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan and J. Snoek, Advances in Neural Information Processing Systems, 13991-14002 (2019).
55. J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano and E. K. Oermann, PLOS Medicine **15** (11), e1002683 (2018).
56. V. Nagarajan, A. Andreassen and B. Neyshabur, arXiv preprint arXiv:2010.15775 (2020).
57. S. Beery, G. Van Horn and P. Perona, Proceedings of the European conference on computer vision (ECCV), 456-473 (2018).
58. M. Arjovsky, L. Bottou, I. Gulrajani and D. Lopez-Paz, arXiv preprint arXiv:1907.02893 (2019).

59. https://www.3ds.com/products-services/biovia/products/molecular-modeling-simulation/biovia-materials-studio/.
60. T. D. K. e. al., J. Chem. Phys. **152**, 194103 (2020).
61. D. R. G. C.J. Permann, D. Andrš, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. W. Peterson, A. E. Slaughter, R. H. Stogner and R. C. Martineau, SoftwareX **11**.
62. D. Guido, *The Finite Element Method for Three-Dimensional Thermomechanical Applications*. (John Wiley & Sons, 2004).
63. A. S. G. K. C. Koppenhoefer, C. Ruggieri, R. H. Dodds, Robert H. Jr and B. E. Healey, Report No. Civil Engineering Studies SRS-596, 1994.
64. P. M. Malinen and P. Råback, Appl. Mater. Sci. **19**, 101-113 (2013).
65. https://lammps.sandia.gov/.
66. E. A. J.M. Soler, J. D. Gale, A. García, J. Junquera, P. Ordejón and D. Sánchez-Portal, Journal of Physics: Condensed Matter **14**, 2745 (2002).
67. A. G. e. al., Journal of Chemical Physics **152**, 204108.
68. P. G. e. al., Journal of Physics: Condensed Matter **21**, 395502 (2009).
69. P. G. e. al., Journal of Physics: Condensed Matter **29(46)**, 465901 (2017).
70. P. G. e. al., J. Chem. Phys **152**, 154105 (2020).