OAK RIDGE National Laboratory

Deep Networks for Imaging: Applications towards Automated Experiments

Ayana Ghosh

Postdoctoral Research Associate

Center for Nanophase Materials Sciences & Computational Sciences & Engineering Division, Oak Ridge National Laboratory, USA

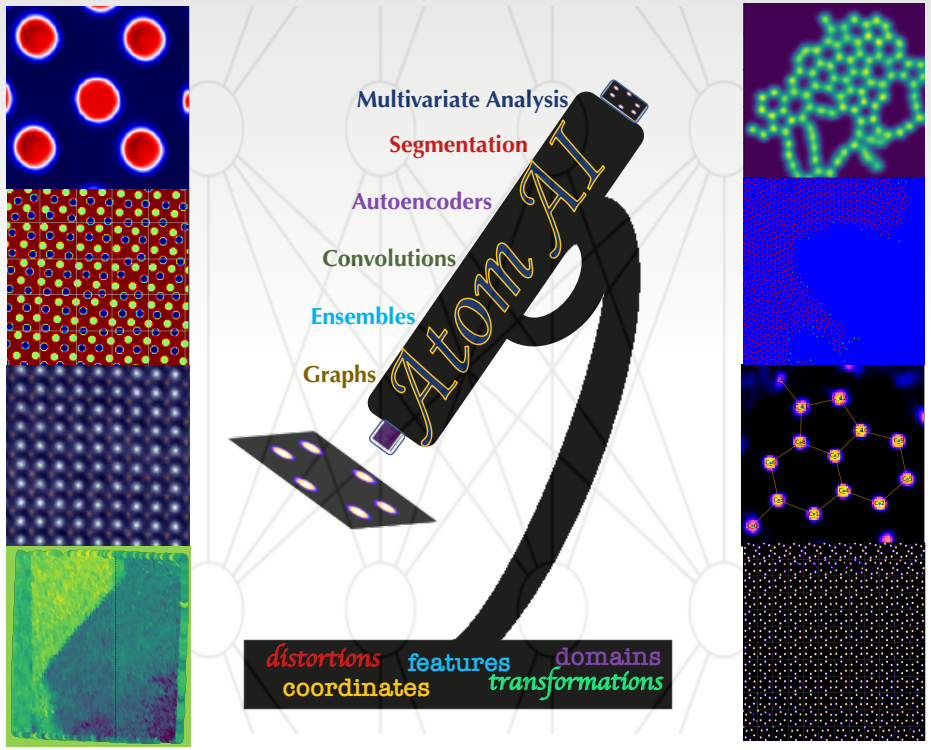
Acknowledgments: Maxim Ziatdinov, Sergei V. Kalinin

In this presentation…

Introduction to Deep Networks for Imaging

- Traditional Vs. Deep learning
- Usefulness of deep networks
- Working principles
- Semantic segmentation
- Popular deep learning frameworks

Package we are going to use

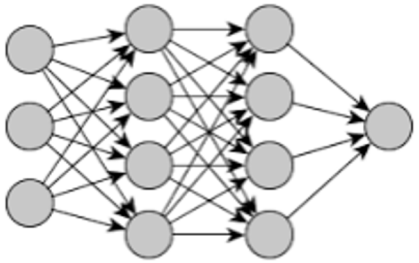


Traditional Vs. Deep Learning

Features Extraction

+

Classification algorithm



Large number of layers

Both could be supervised or unsupervised

Small Datasets

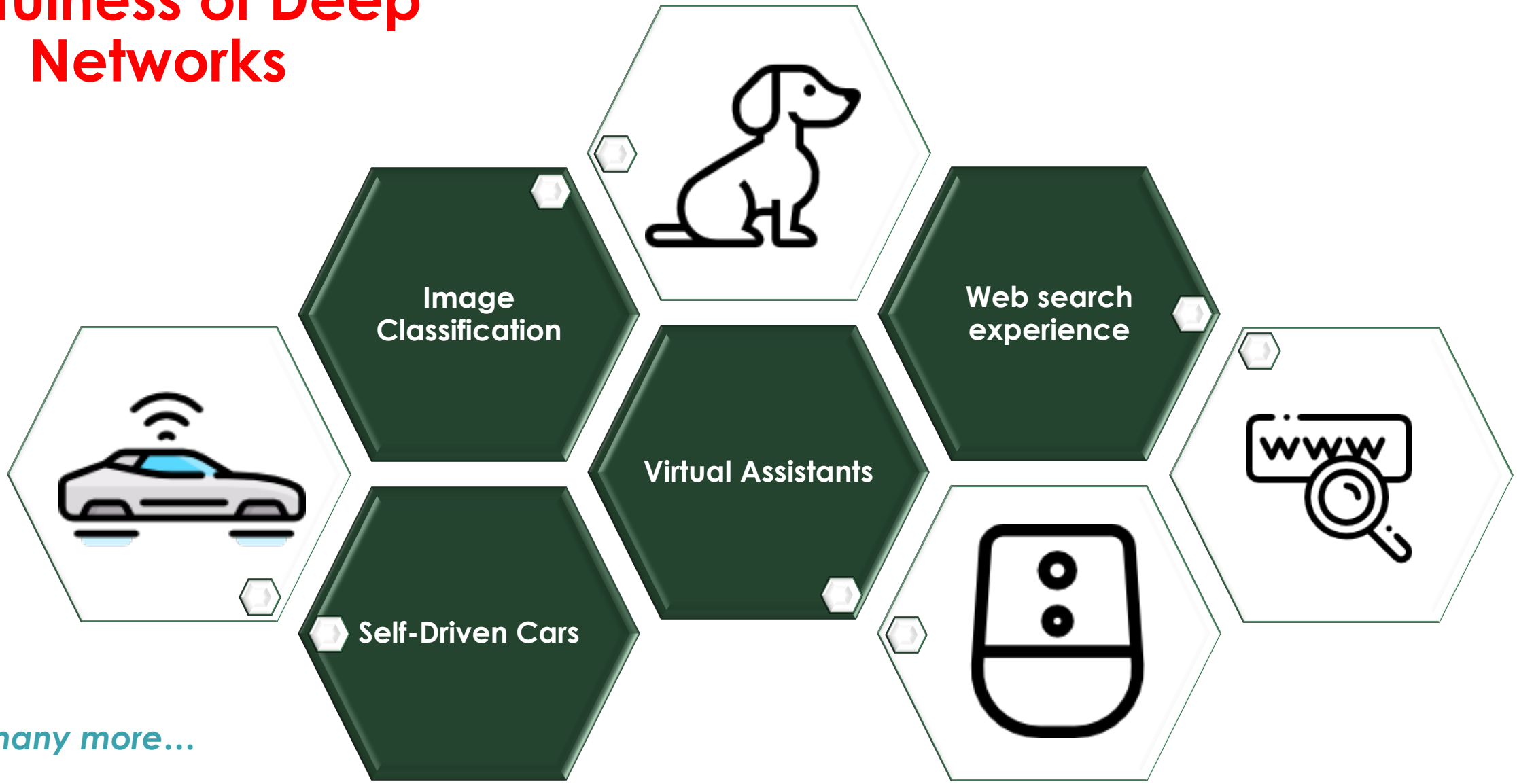
Feature Engineering

Interpretability

Large Datasets

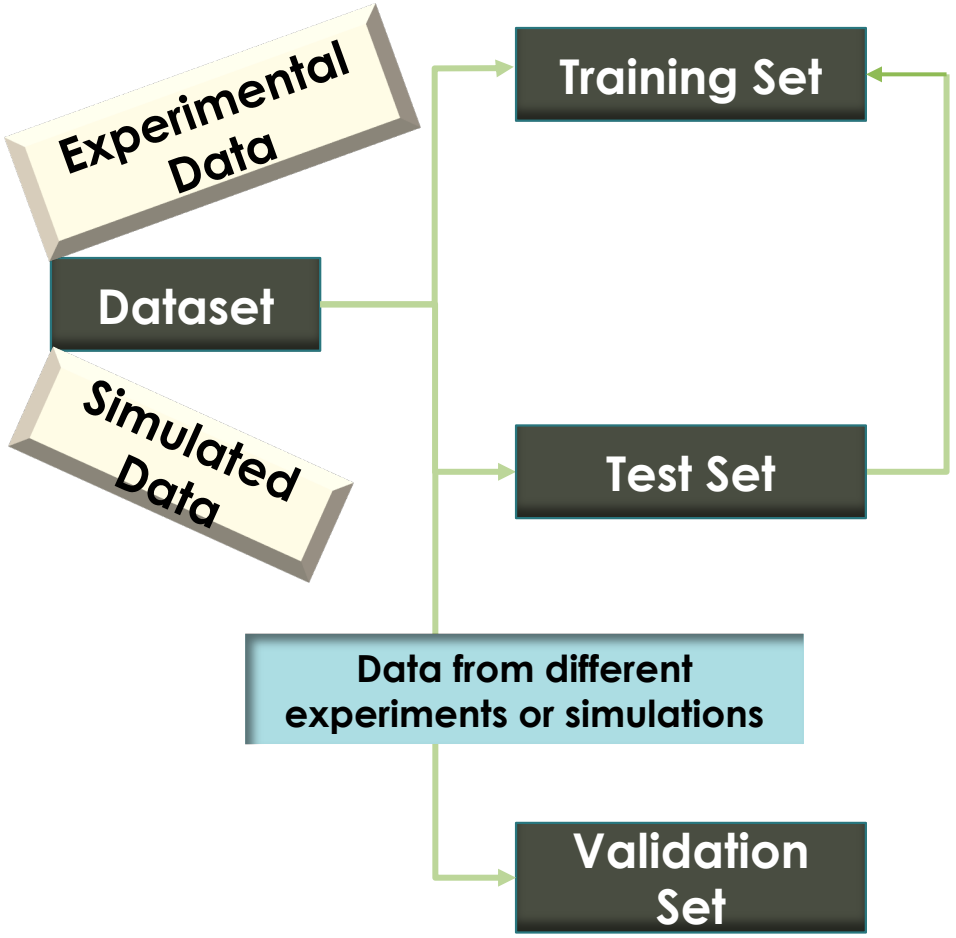
Avoids the crucial step of feature engineering

Usefulness of Deep Networks



And many more…

Input data



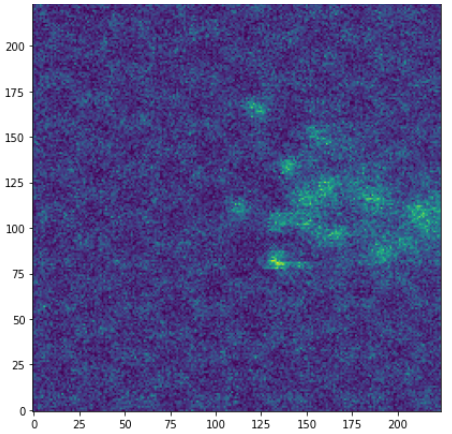
Types of data labels

Unlabeled

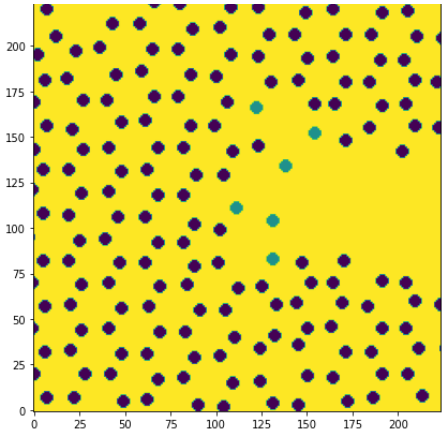
Weakly-labeled

Fully-labeled

Image

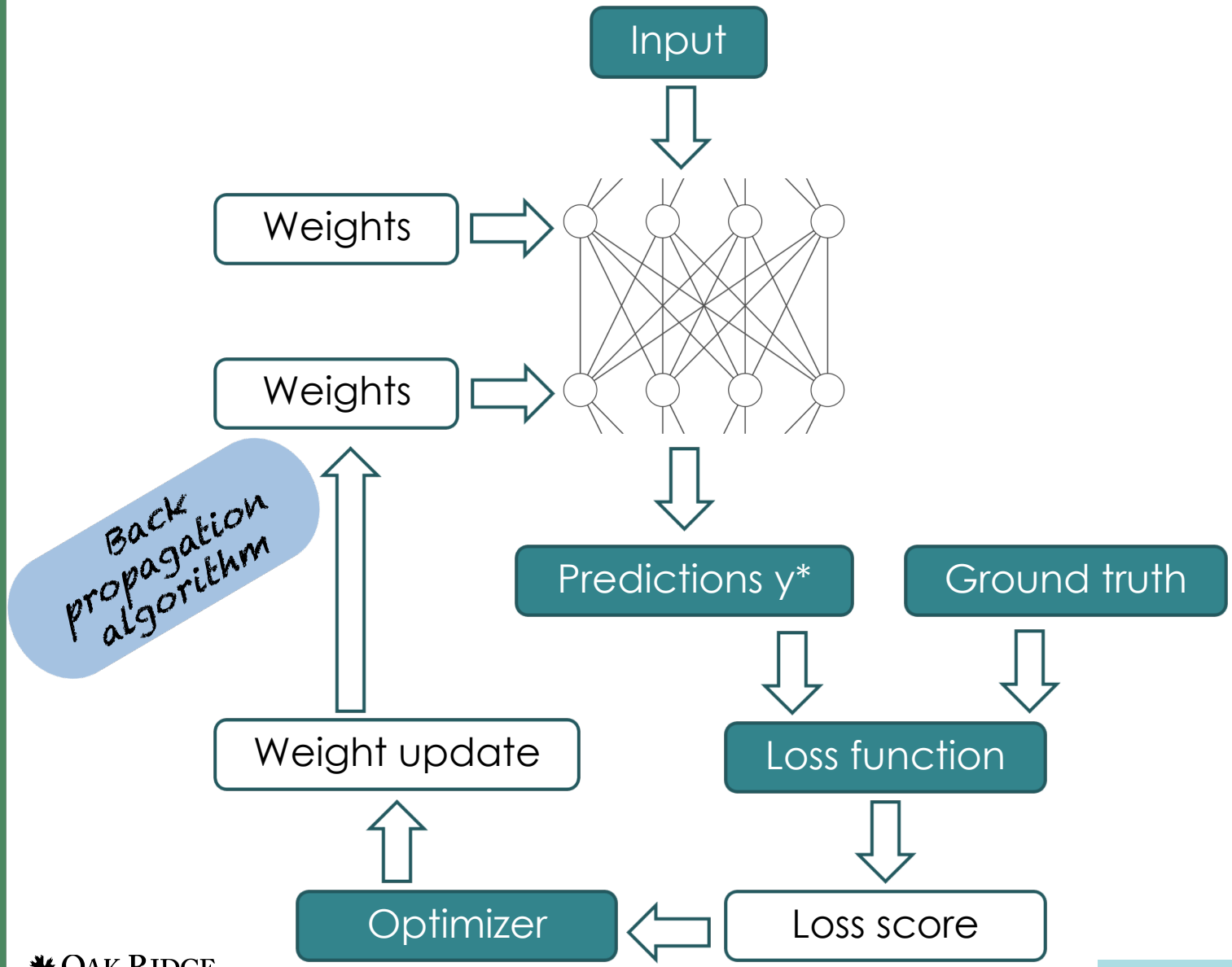


Ground truth



For models constructed on simulated data, it then becomes important to perform experiment-theory matching !

Primary Working Principles

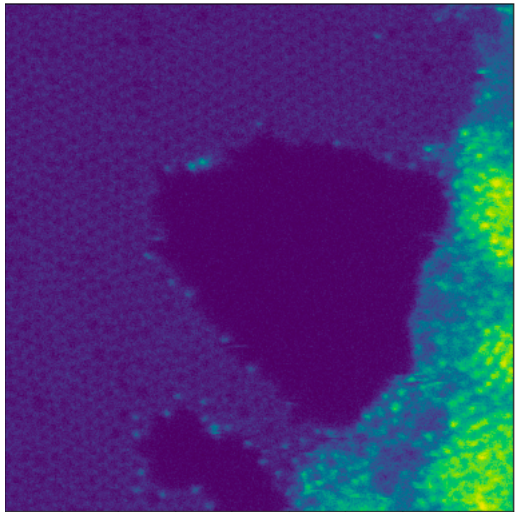


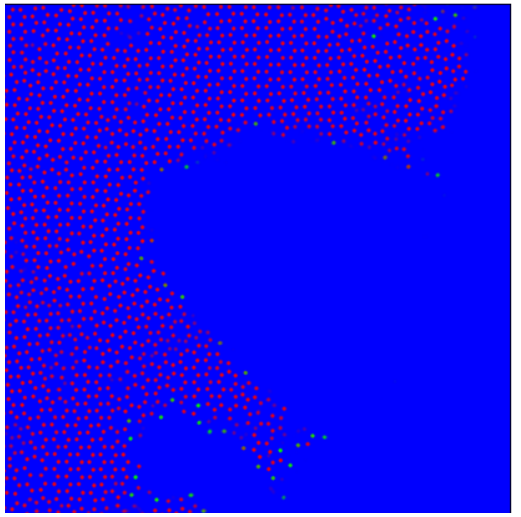
- The network is parameterized by its weights
- A loss function measures the quality of the network's output
- The gradient of the loss is computed for the network's parameters using back propagation algorithm
- The loss score is used as a feedback to adjust the weights

Chollet, F.: *Deep Learning with Python*; Manning Publication Co., 2018

Semantic Segmentation

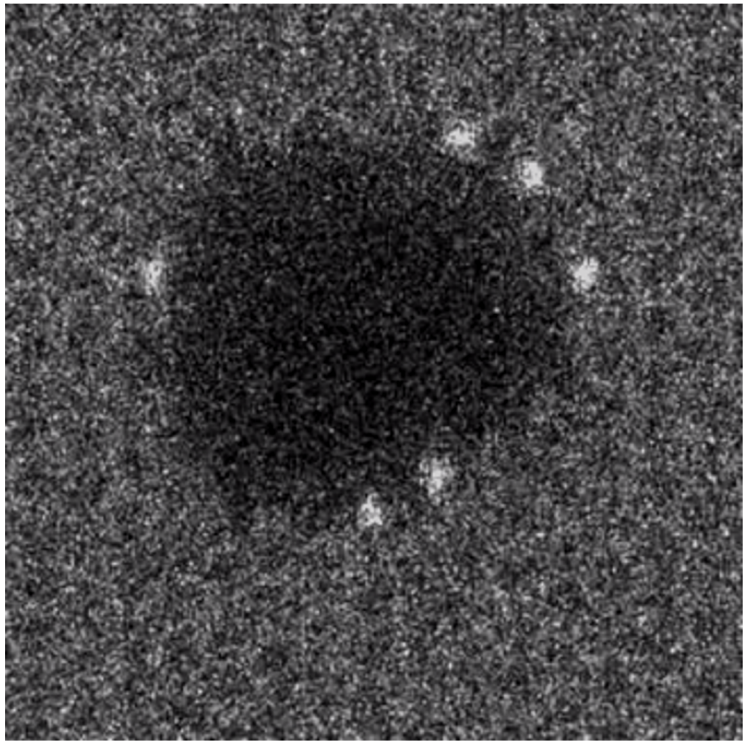
- Deep learning strategies applied to categorize natural images into different classes, e.g., CIFAR, MNIST datasets
- Each class generally contains thousands of samples with large variability
- Image segmentation: Atom/particle/defect identification in STEM/SPM
  - Find nearly identical objects
  - Significant variations in image acquisition parameters
  - Difficult for a neural network trained on a broad distribution of parameters to recognize subtle features
  - Feature finding is much challenging



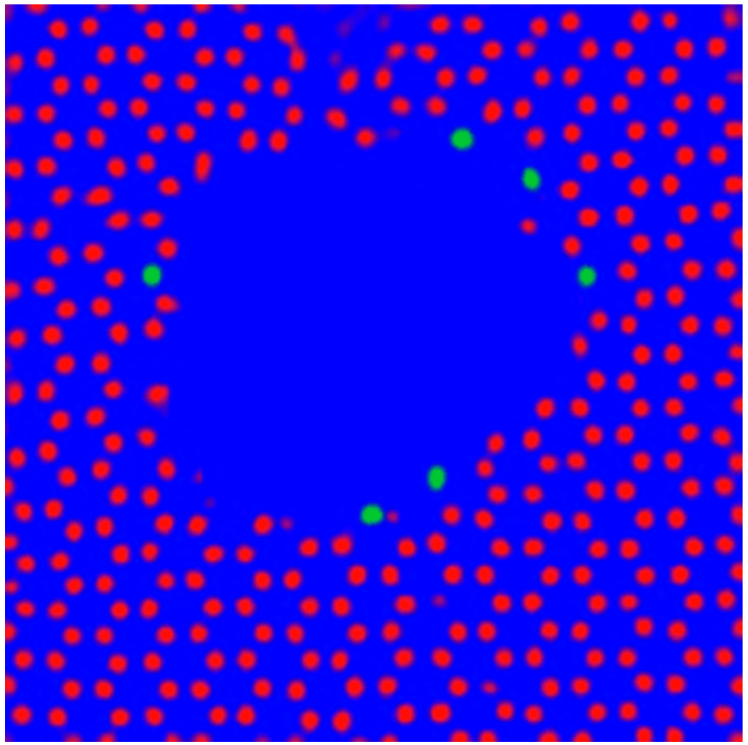


Application to real data: Semantic segmentation of noisy images ("denoising" + atom finding)

Experiment



Neural network output

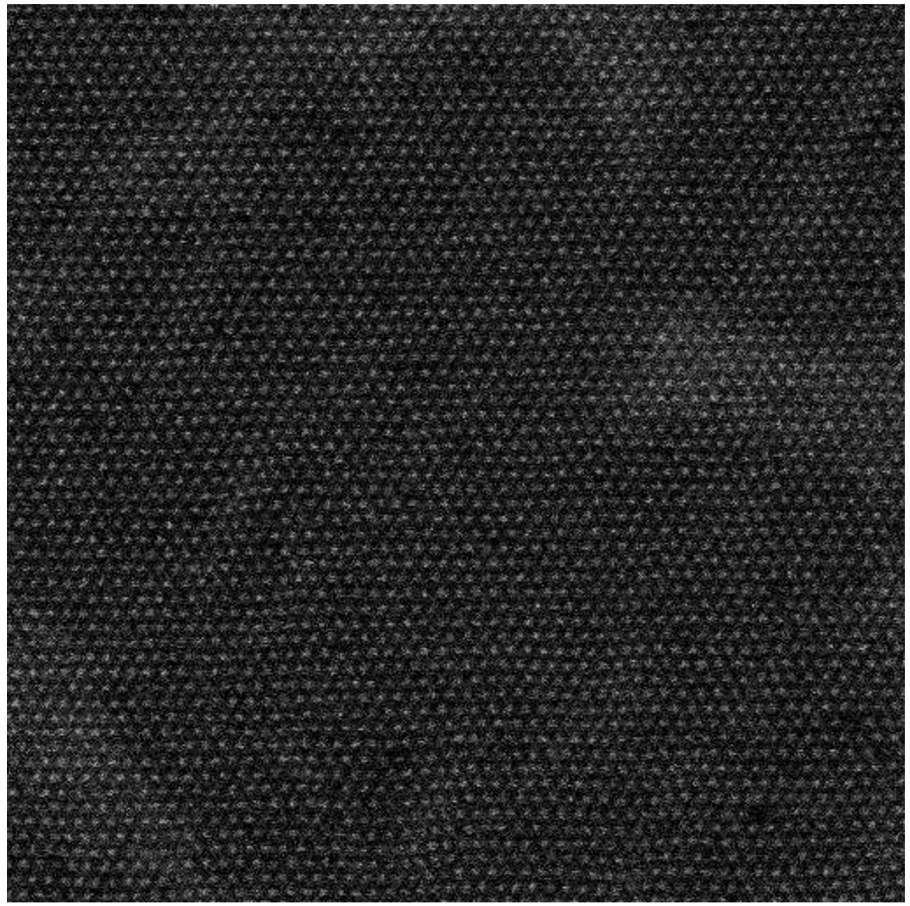


- U-Net-based models typically work well for noisy data
- The neural network accomplishes two goals:
  - Removes noise
  - Separate different atom/defect types into different classes

https://arxiv.org/abs/2105.07485

Application to real data: Finding complex defects

Experimental

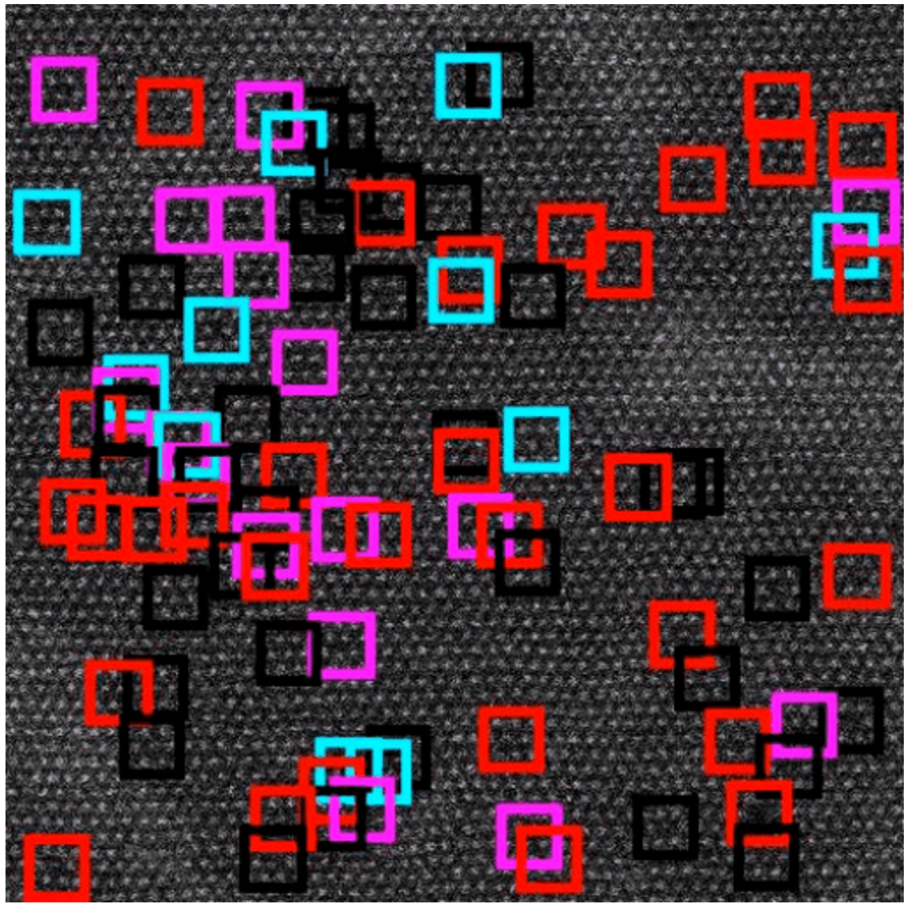


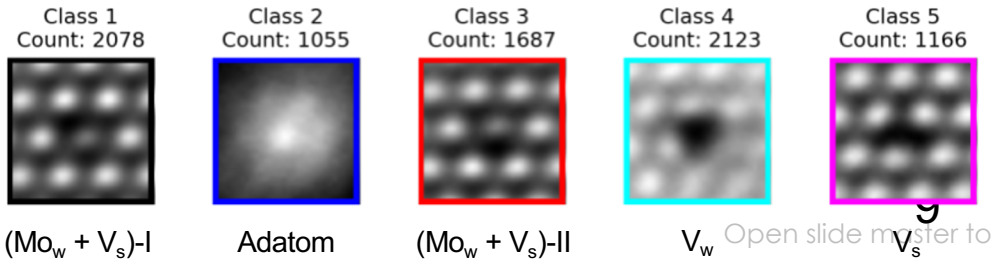
Sample: WS2
E-beam energy: 60 kV

Data collected by Ondrej Dyck (CNMS/ORNL)

Slide Credit: Maxim Ziatdinov

Decoded





Popular Deep Learning Frameworks

TensorFlow (2.x)

```python
# Import tensorflow
import tensorflow as tf

# Define a model
class NNRegressor(tf.keras.Model):
    """2-layer neural network for regression"""
    def __init__(self, nb_neurons):
        """Initialization of model parameters"""
        super(NNRegressor, self).__init__()
        self.layer1 = tf.keras.layers.Dense(nb_neurons)
        self.layer2 = tf.keras.layers.Dense(nb_neurons)
        self.output_layer = tf.keras.layers.Dense(1)
        self.relu = tf.keras.activations.relu

    def call(self, x):
        """Forward path"""
        l1 = self.relu(self.layer1(x))
        l2 = self.relu(self.layer2(l1))
        return self.output_layer(l2)

# Initialize model
model = NNRegressor(nb_neurons=100)

# Specify weights optimizer and loss function (criterion)
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-4)
criterion = tf.keras.losses.MeanSquaredError()
```

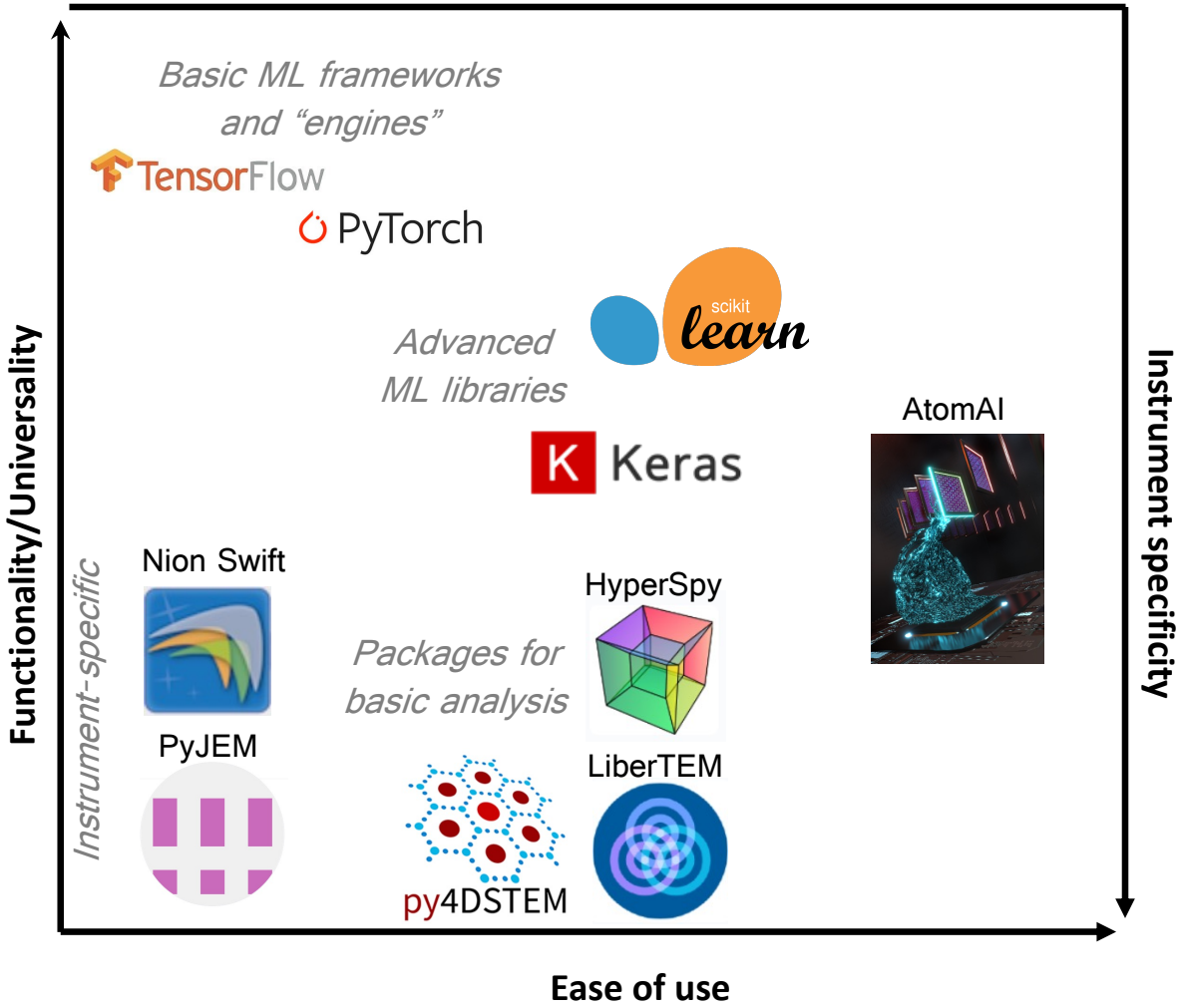
PyTorch

```python
# Import pytorch
import torch

# Define a model
class NNRegressor(torch.nn.Module):
    """2-layer neural network for regression"""
    def __init__(self, nb_neurons):
        """Initialization of model parameters"""
        super(NNRegressor, self).__init__()
        self.layer1 = torch.nn.Linear(1, nb_neurons)
        self.layer2 = torch.nn.Linear(nb_neurons, nb_neurons)
        self.output_layer = torch.nn.Linear(nb_neurons, 1)
        self.lrelu = torch.nn.functional.leaky_relu

    def forward(self, x):
        """Forward path"""
        l1 = self.lrelu(self.layer1(x))
        l2 = self.lrelu(self.layer2(l1))
        return self.output_layer(l2)

# Initialize model
model = NNRegressor(nb_neurons=100)

# Specify weights optimizer and loss function (criterion)
optimizer = torch.optim.Adam(model.parameters(), lr=5e-4)
criterion = torch.nn.MSELoss()
```


Slide Credit: Maxim Ziatdinov

AtomAI: Microscopic Images to Atomistic Simulations to Learning and Discovering Physics

- Python-based infrastructure bridging instruments and HPC
- Microscopic and mesoscopic data analysis and beyond
- Neural Networks and other machine learning methods



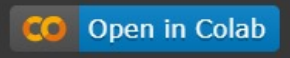
https://arxiv.org/abs/2105.07485

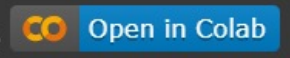
OAK RIDGE National Laboratory

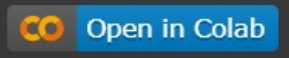
AtomAI: Extensive List of Interactive Examples & Tutorials

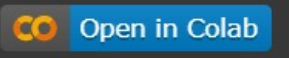
Quickstart: AtomAI in the Cloud

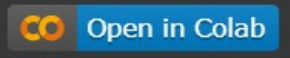
The easiest way to start using AtomAI is via Google Colab, which is a free research tool from Google for machine learning education and research built on top of Jupyter Notebook. The following notebooks can be executed in Google Colab by simply clicking on "Open in Colab" icon:

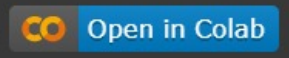
1) Train a Deep Fully Convolutional Neural Network for Atom Finding 

2) Multivariate Statistical Analysis of Distortion Domains in a Single Atomic Image 

3) Variational Autoencoders I: Learning Disentangled Representations of Arbitrarily Rotated Handwritten Digits 

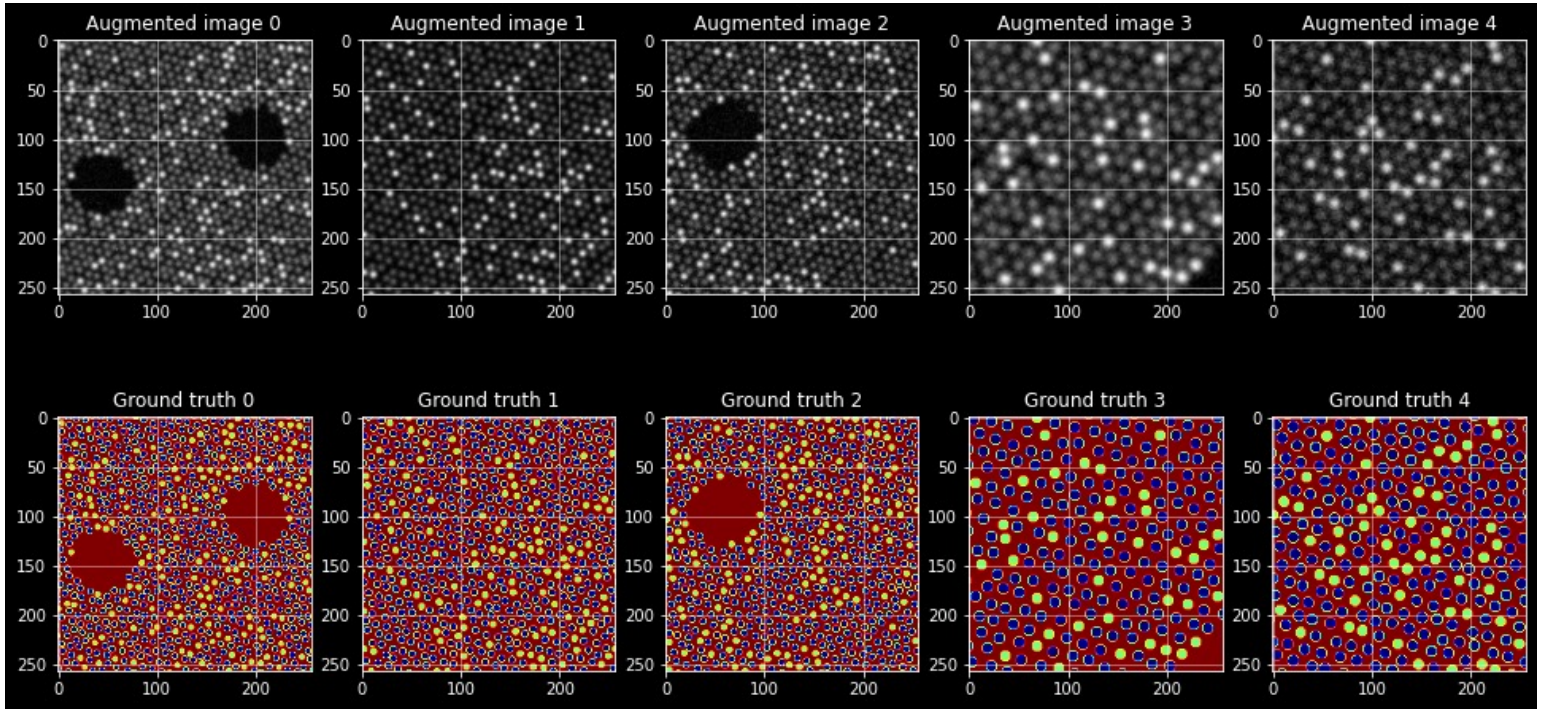
4) Variational Autoencoders II: Simple Analysis of Structural Transformations in Atomic Movies 

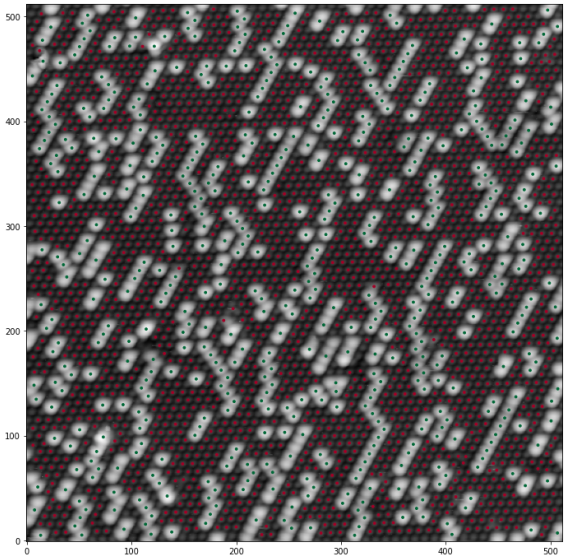
5) Implementation of Custom Image Denoiser in AtomAI 

6) Prepare Training Data from Experimental Image with Atomic Coordinates 

In Today's Tutorials

- Perform semantic segmentation on an STM image
- We use simulated data to train a DL U-Net type of model
- Use that to find atoms, coordinates from the experimental image

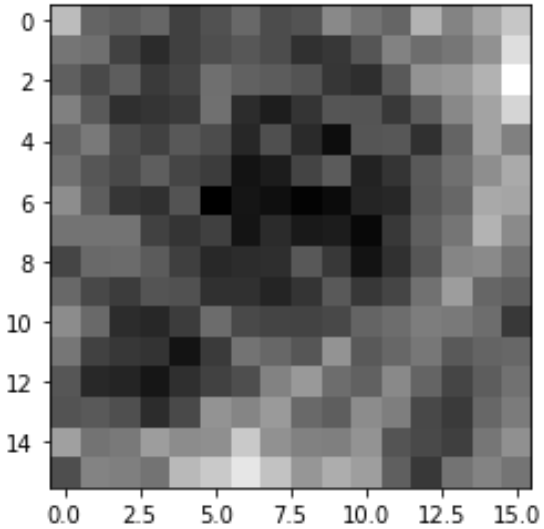


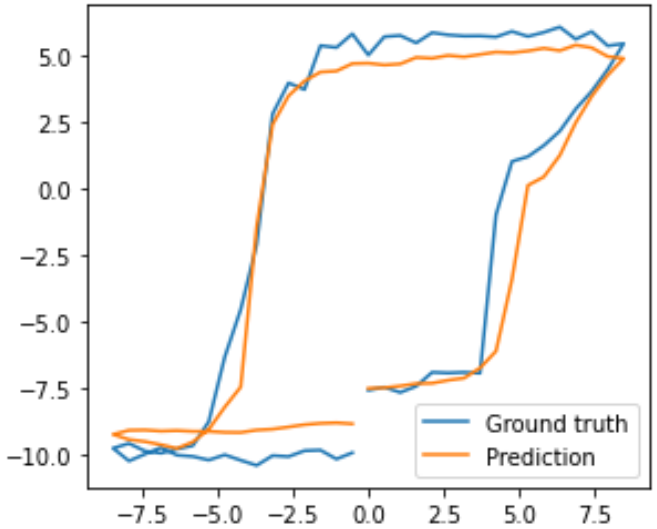


OAK RIDGE National Laboratory

In Today's Tutorials

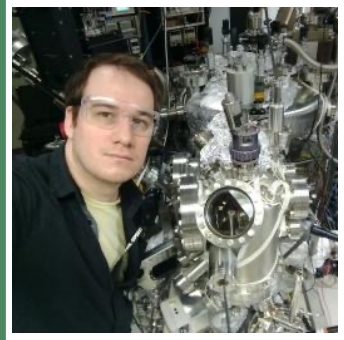
- Predict functional property (spectrum) from input image
- im2spec models
  - Use (2D CNN layers) to embed input images into a latent vector
  - Decoder for generating one-dimensional signals from the embedded features
  - Encoder part consists of three back-to-back two-dimensional convolutional layers
  - Decoder part represents a cascade of one-dimensional dilated convolutions





https://pubs.acs.org/doi/pdf/10.1021/acsami.0c15085

Acknowledgments



Thank you for your attention

- U.S. Department of Energy (DOE), Office of Science, Office of Basic Energy Sciences Data, Artificial Intelligence and Machine Learning at DOE Scientific User Facilities
- DOE, Office of Science, Basic Energy Sciences (BES), Materials Sciences and Engineering Division
- Oak Ridge National Laboratory's Center for Nanophase Materials Sciences (CNMS)

OAK RIDGE National Laboratory

Link to Tutorials!

https://github.com/pycroscopy/SPM_ML_School_2021/tree/main/Day01/Notebooks