THE UNIVERSITY of TENNESSEE KNOXVILLE

Day 2. Linear Unmixing Methods for Image Analysis

Sergei V. Kalinin

High Dimensional Data – what should we do?

Examples of high D data:

- Face recognition
- Image compression
- Gene expression analysis
- Spectroscopy
- 4D STEM
- X-Ray scattering

What do we want to accomplish?

- Reduce number of dimensions in data
- Find patterns in high-dimensional data
- Visualize data of high dimensionality

High Dimensional Data is often redundant

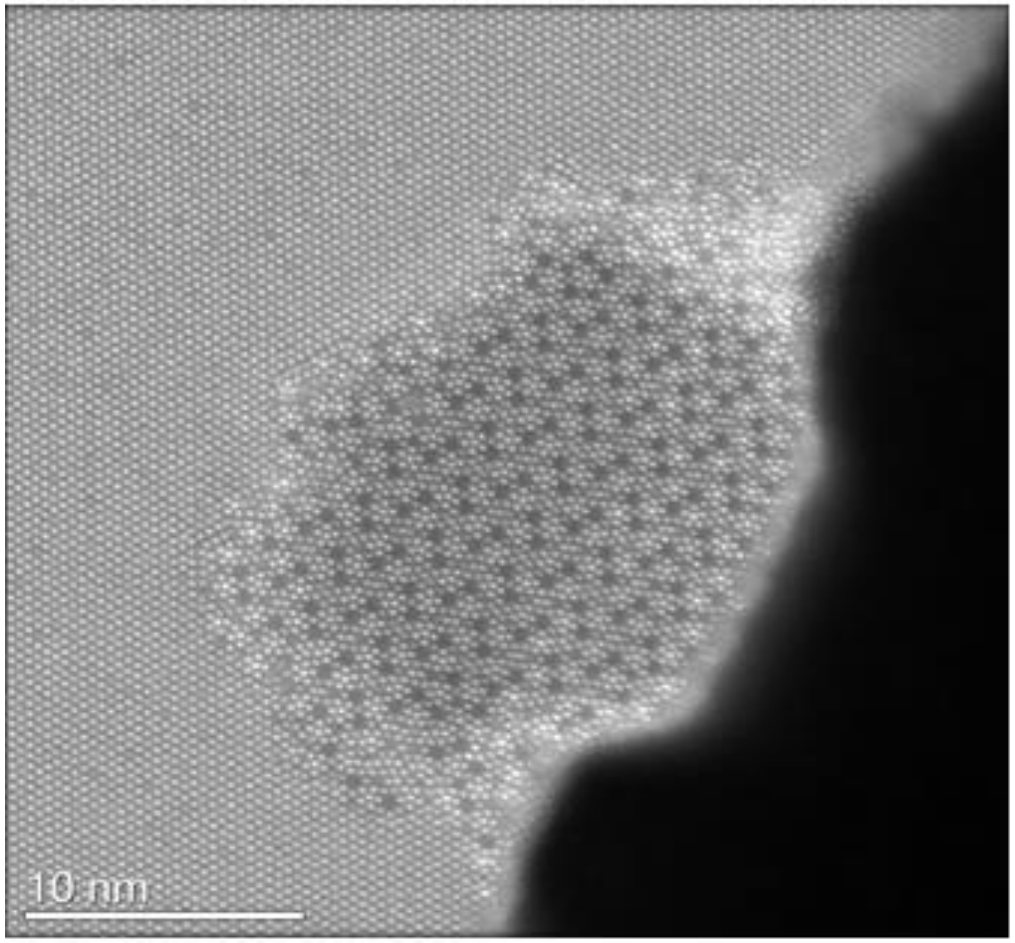
- We often need a method to simplify data on a large number of variables, and believe that there is some redundancy in those variables.
- Redundancy means that some of the variables are correlated with one another, possibly because they are measuring the same object or phenomenon.
- Because of redundancy, we believe that it should be possible to reduce the observed variables into a smaller number of artificial variables that will account for most of the variance in the observed variables.

Dimensionality Reduction Methods

- PCA (Principal Component Analysis):
  - Find projection that maximize the variance
- ICA (Independent Component Analysis):
  - Very similar to PCA except that it assumes non-Gaussian features
- Multidimensional Scaling:
  - Find projection that best preserves inter-point distances
- LDA(Linear Discriminant Analysis):
  - Maximizing the component axes for class-separation
- Bayesian Linear Unmixing
  - Linear unmixing, non-negative, sum to one
- … constrained linear unmixing methods

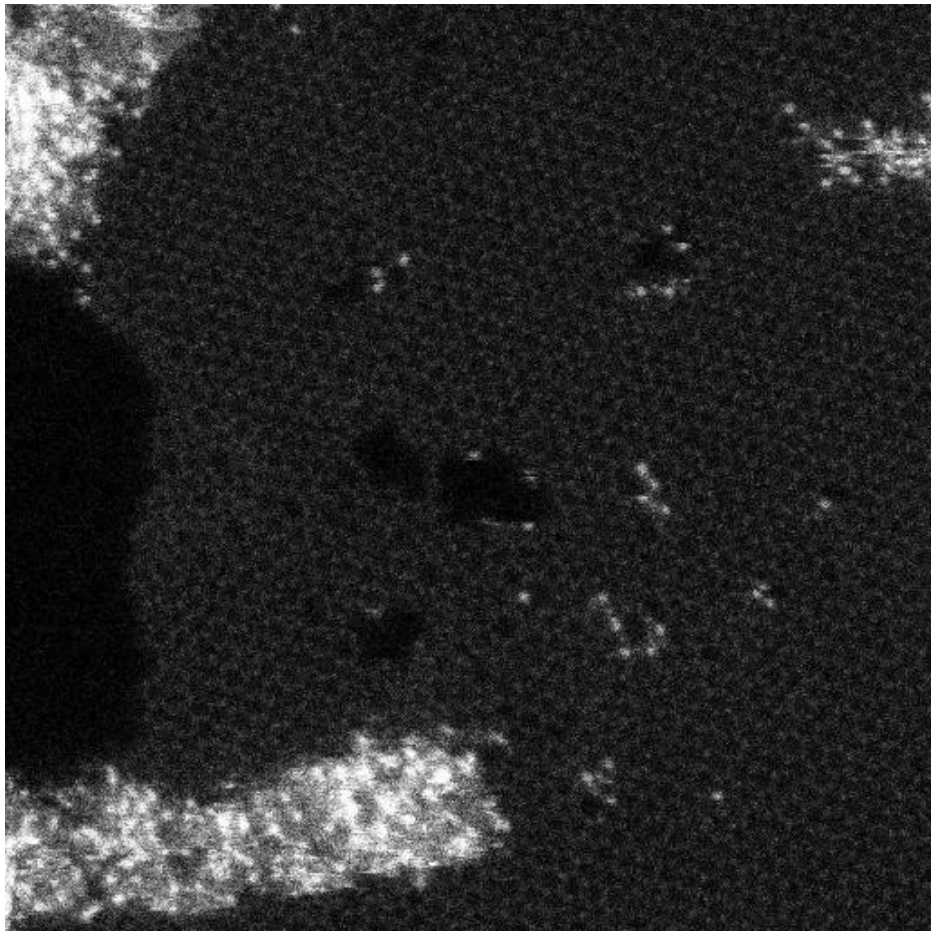
Chemically disordered systems

Mo-V-Ta complex oxide



Q. He et al, ACS Nano 9, 3470-3478

Si in graphene

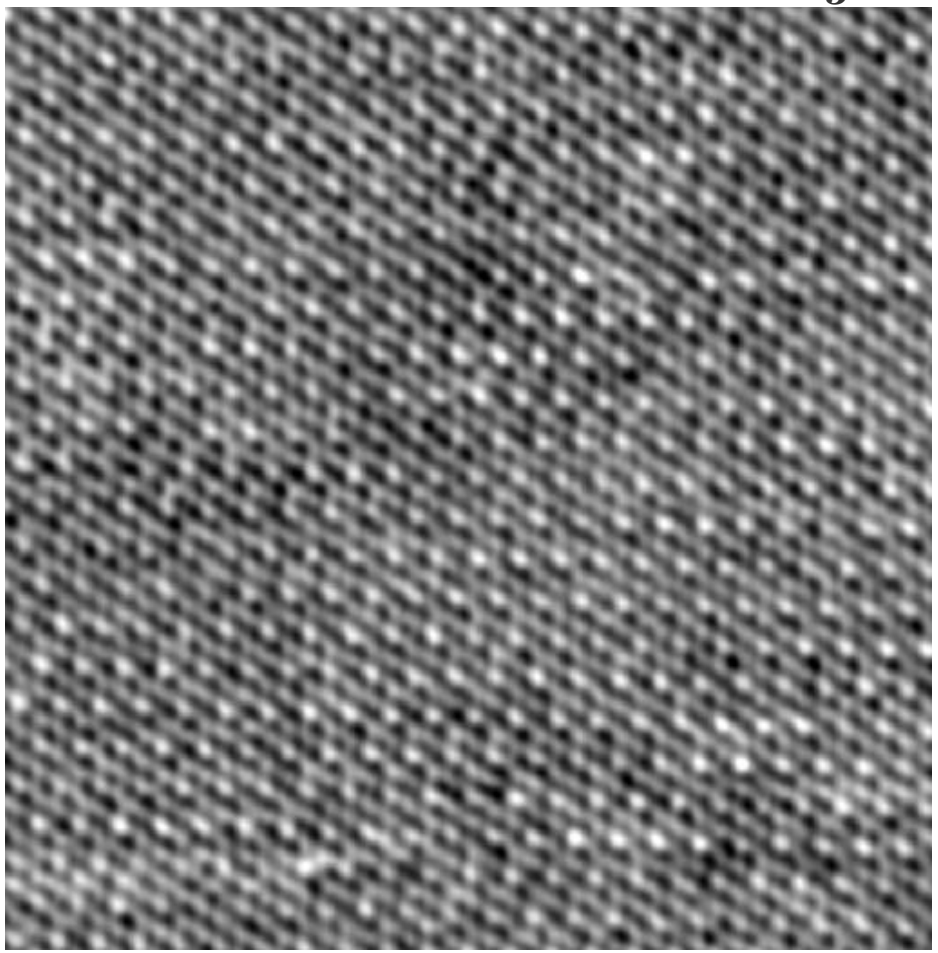


Data collected by O. Dyck (ORNL)

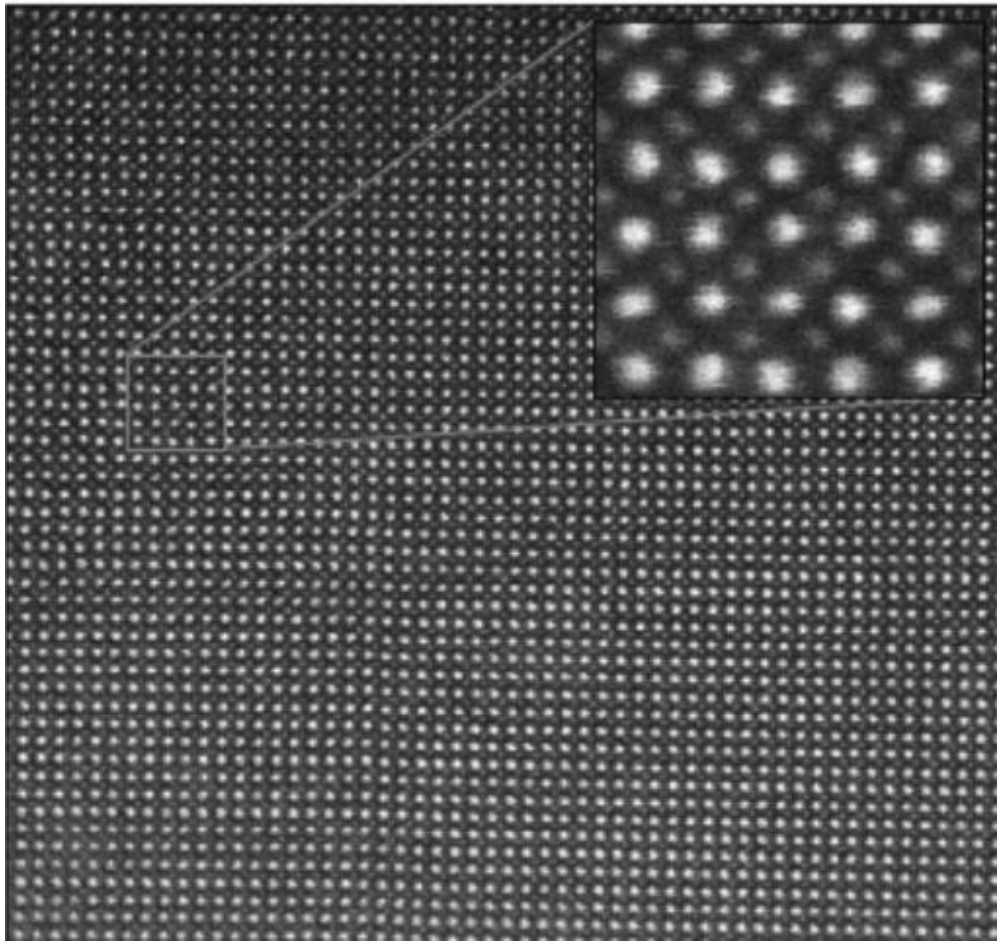
- What is the nature of the building blocks and relevant atomic configurations?
- Can we define single-phase regions and phase boundaries?

But what about subtle distortions?

Electronic structure in $RuCl_3$



$BiFeO_3$ on $SrRuO_3$



- Can we identify ferroelectric and ferroic variants and associated topological defects?
- What is the nature of the phases?

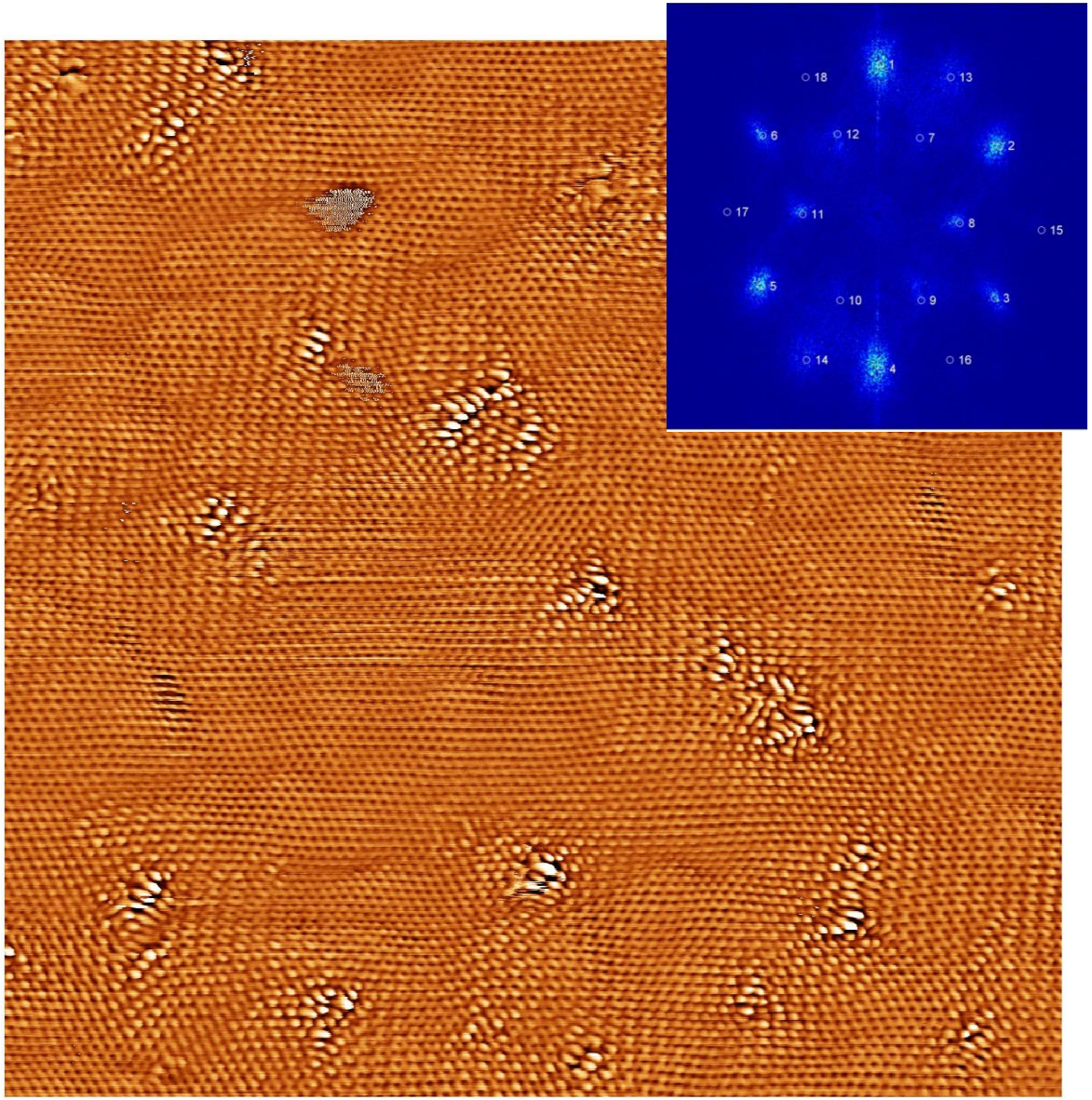
Can global FFT help?

- Global FFT – everything is averaged:
  - Drift
  - Extended defects
  - Multiple grains

We are averaging out all interesting phenomena except for small spatially uniform structural distortions

- Solution – sliding window approaches:
  - Fit FFT peaks: amplitudes, positions
  - Multivariate analysis

Note: window can be also tied to a specific feature, such a selected atom. Then we explore atomic neighborhood



Sliding FFT:

- We always have a problem of window size:
  - too large – loose spatial resolution,
  - too small – FFT behaves poorly due to edge effects
- Interpretation of FFT data is complicated (too much data if fit each peak, unclear meaning of the unmixing components)
- Natural descriptor for atomically resolved images – atomic coordinates!

General linear unmixing

$$S(\mathbf{x}, \mathbf{R}) = \sum_i a_i(\mathbf{x})\, w_i(\mathbf{R}) + \mathrm{N}$$

We start with:

- **x** is the spatial variable, x = (*x*,*y*)
- R is the (vector) parameter variable

We aim to get:

- $a_i(\mathbf{x})$ are loading maps
- $w_i(\mathbf{R})$ are endmembers/eigenvectors
- N is noise

The M pixel 2D image is transformed to M/N pixel image of more complex structure.

Our loading map is 2D image, and endmembers/eigenvectors are 2D images

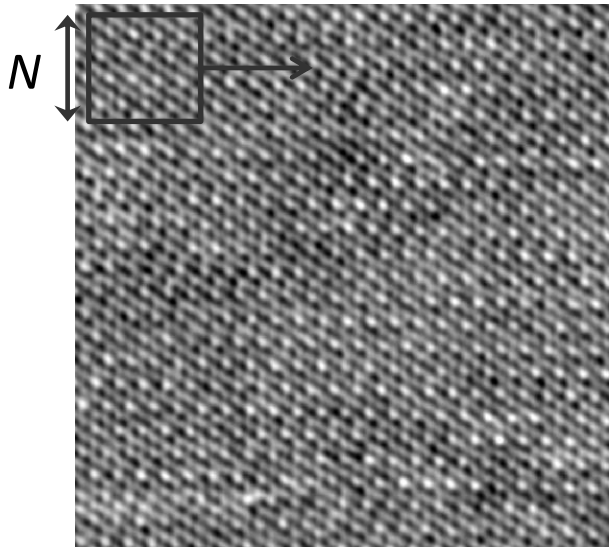


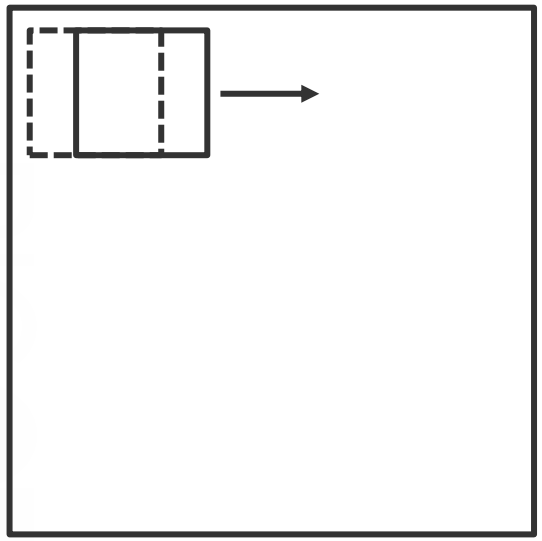
Figure by M. Ziatdinov

Sliding image transforms:

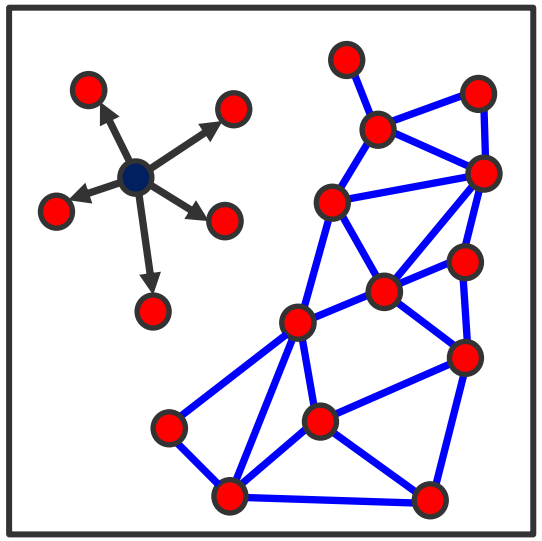
- Fast Fourier Transforms
- Correlation functions
- Intensity histograms
- Structural descriptors

Constructing the descriptors

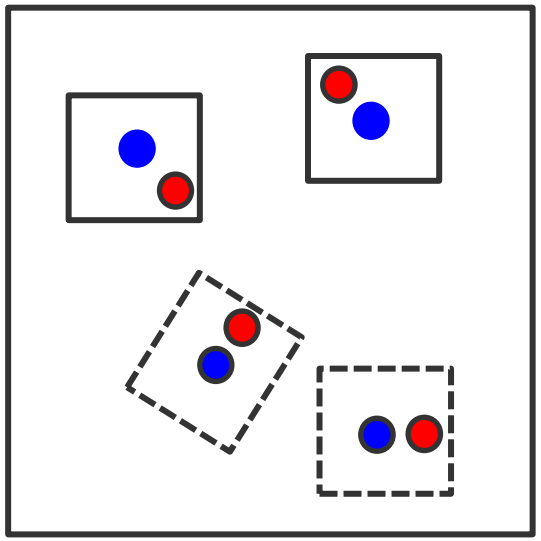
Continuous translational symmetry



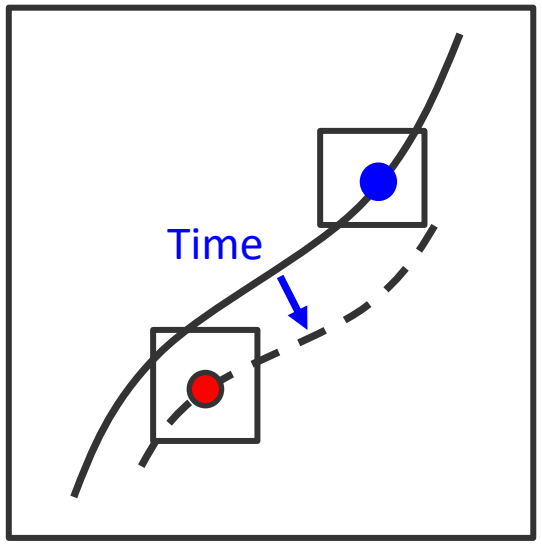
Atom based descriptions



Localized sub-images



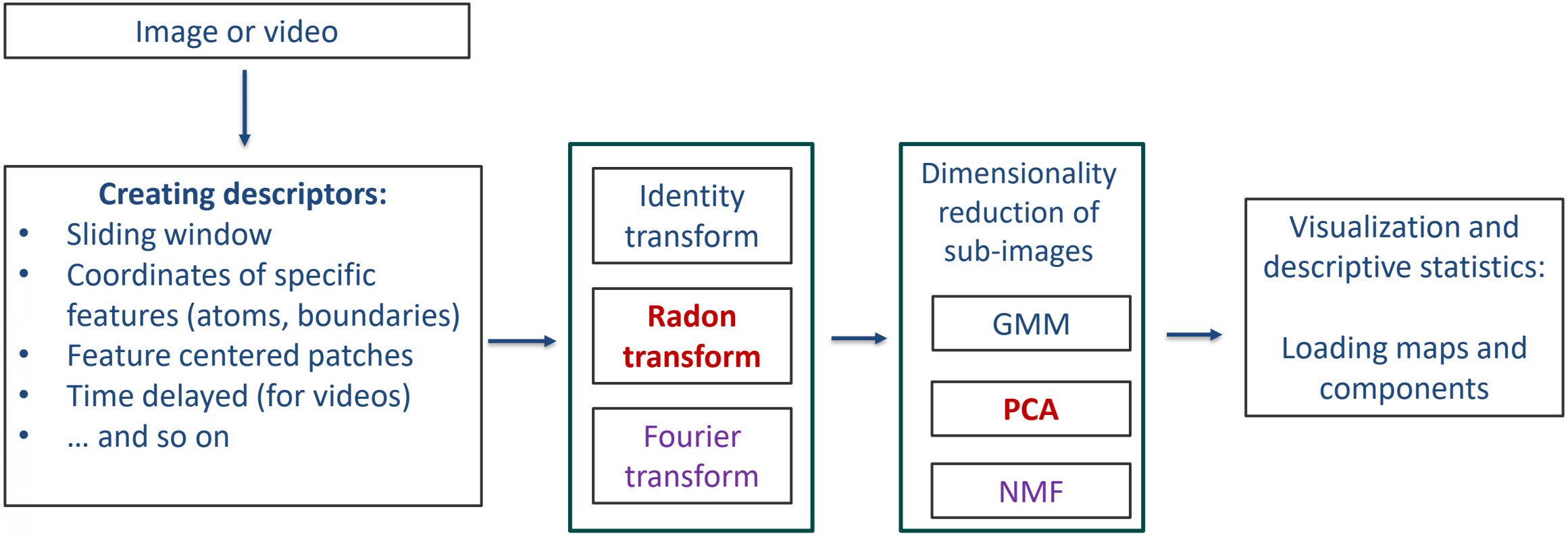
Time-delayed descriptors



The choice of the descriptor:

- Defines physical inferential biases and allows to introduce prior knowledge
- Determines the physical meaning of the analysis
- Establishes the analysis pipeline

Example of analysis pipeline



Pipelines are defined to

- Make analysis traceable, repeatable, explainable, and transferable
- Allow for hyperparameter tuning and optimization
- Efficiently use the memory

How general should you be: depends on applications

Principal Component Analysis

$$S(\mathbf{x}, \mathbf{R}) = \sum_i a_i(\mathbf{x})\, w_i(\mathbf{R})$$

In PCA, the eigenvectors $w_i(\mathbf{R})$ are orthonormal and are arranged such that corresponding eigenvalues are placed in descending order by variance

- Reveals internal structure of the data that best explains variance in the data set
- Since data often moves in clusters, PCA reveals those variables that drive the variance
- PCA transforms the data such that the greatest variance by any projection lies on the first coordinate

Sliding PCA-FFT

Can we use PCA of FFT transform in sliding windows to find periodicity?

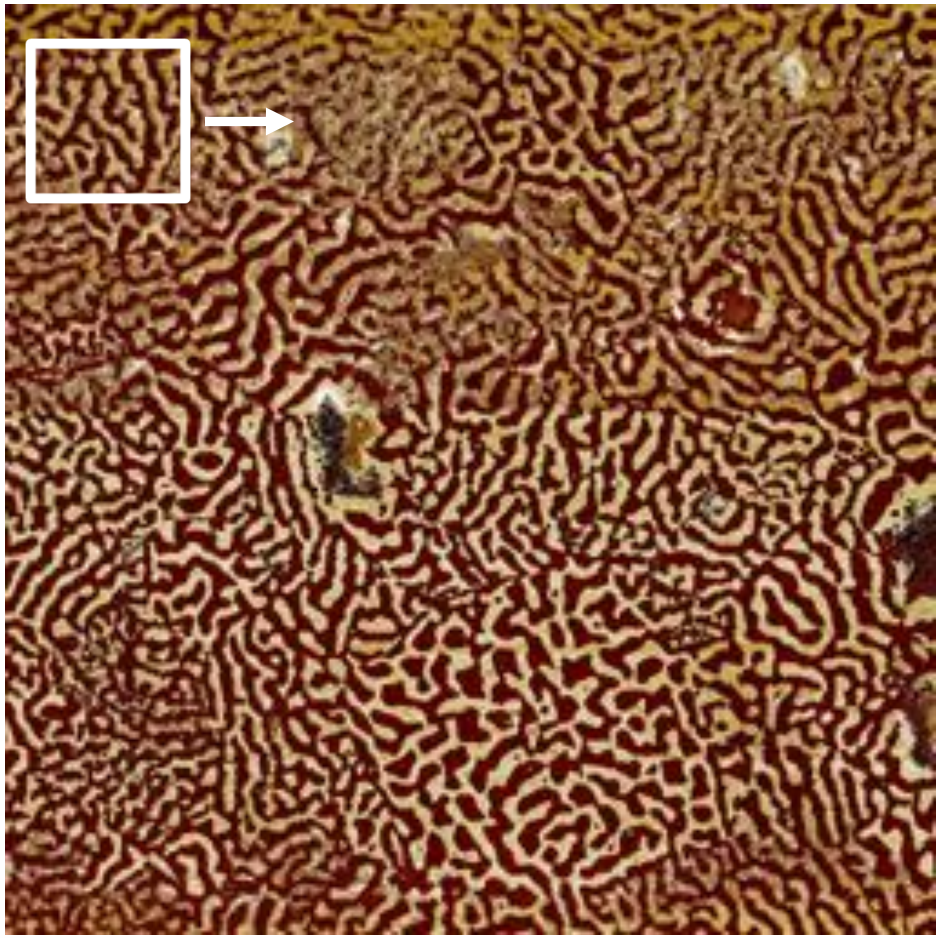
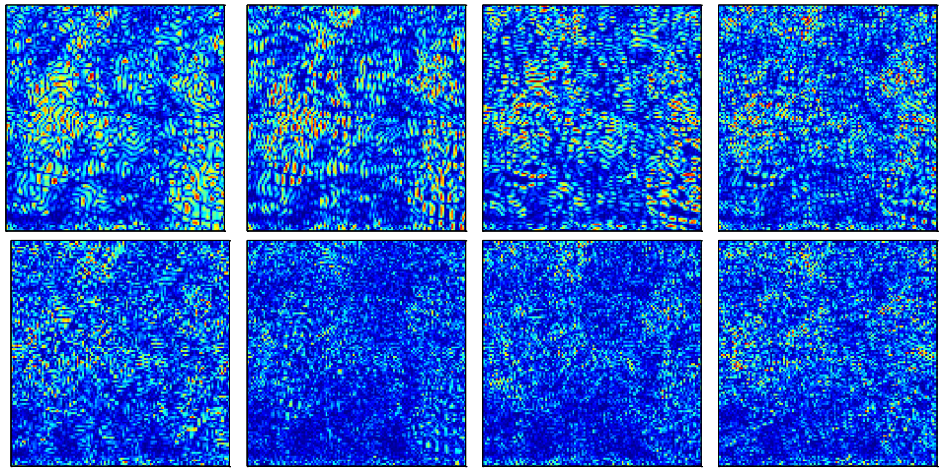
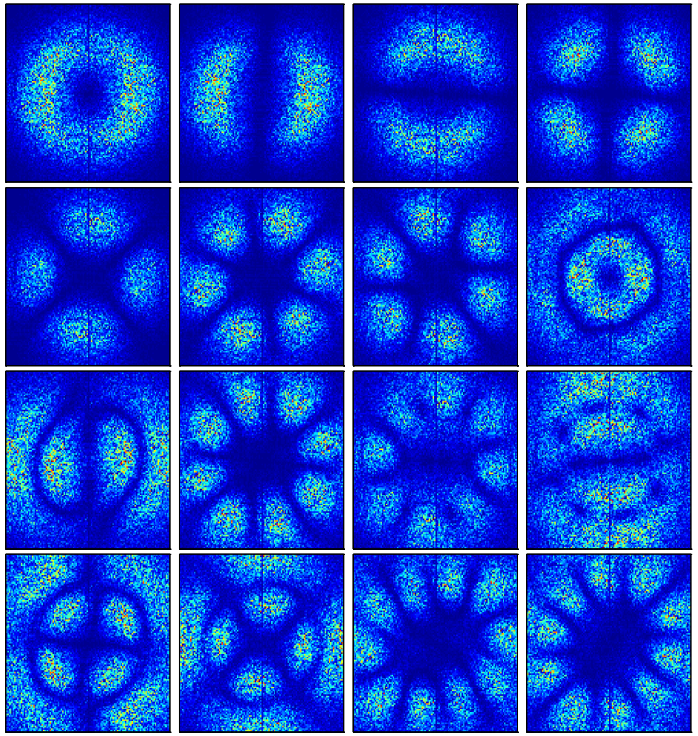


Figure by S. Jesse, data D. Gobelic

First 8 maps



First 16 eigenvectors



Spectral Unmixing: N-FINDR

Spectra for a given pixel is assumed to be a linear combination of the end-member spectra (+ Gaussian noise). The mixing proportions sum to 1

Physics constraint

$$p_{ij} = \sum_k e_{ik} c_{kj} + \varepsilon \qquad \sum_k c_{kj} = 1$$

- Let E be the matrix of end-members (here, 3).

$$\boldsymbol{E} = \left[\overrightarrow{e_1} \quad \overrightarrow{e_2} \quad \overrightarrow{e_3}\right] \qquad V\left(\frac{\mathbf{1}}{(\boldsymbol{l}-1)!}\right)|\det(E)|$$

- Iteratively select endmembers, accepting the new selection if the volume increases

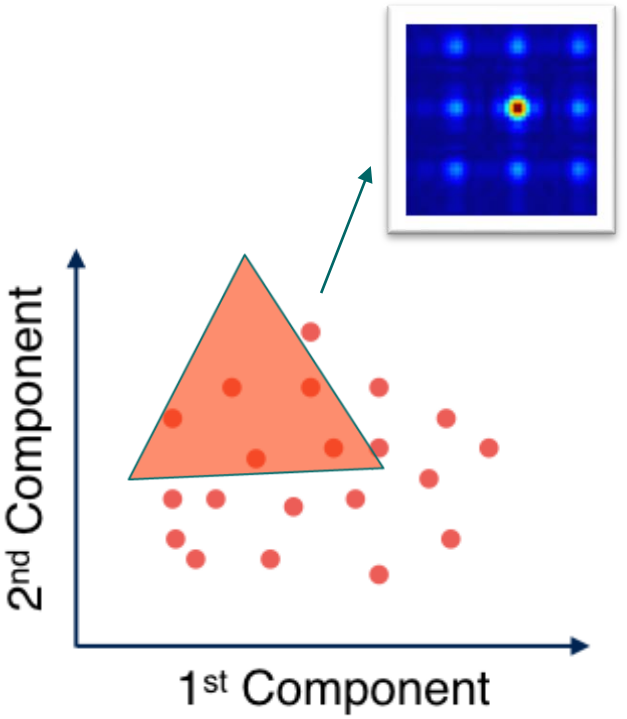


Figure by R. Vasudevan

Ideal test case

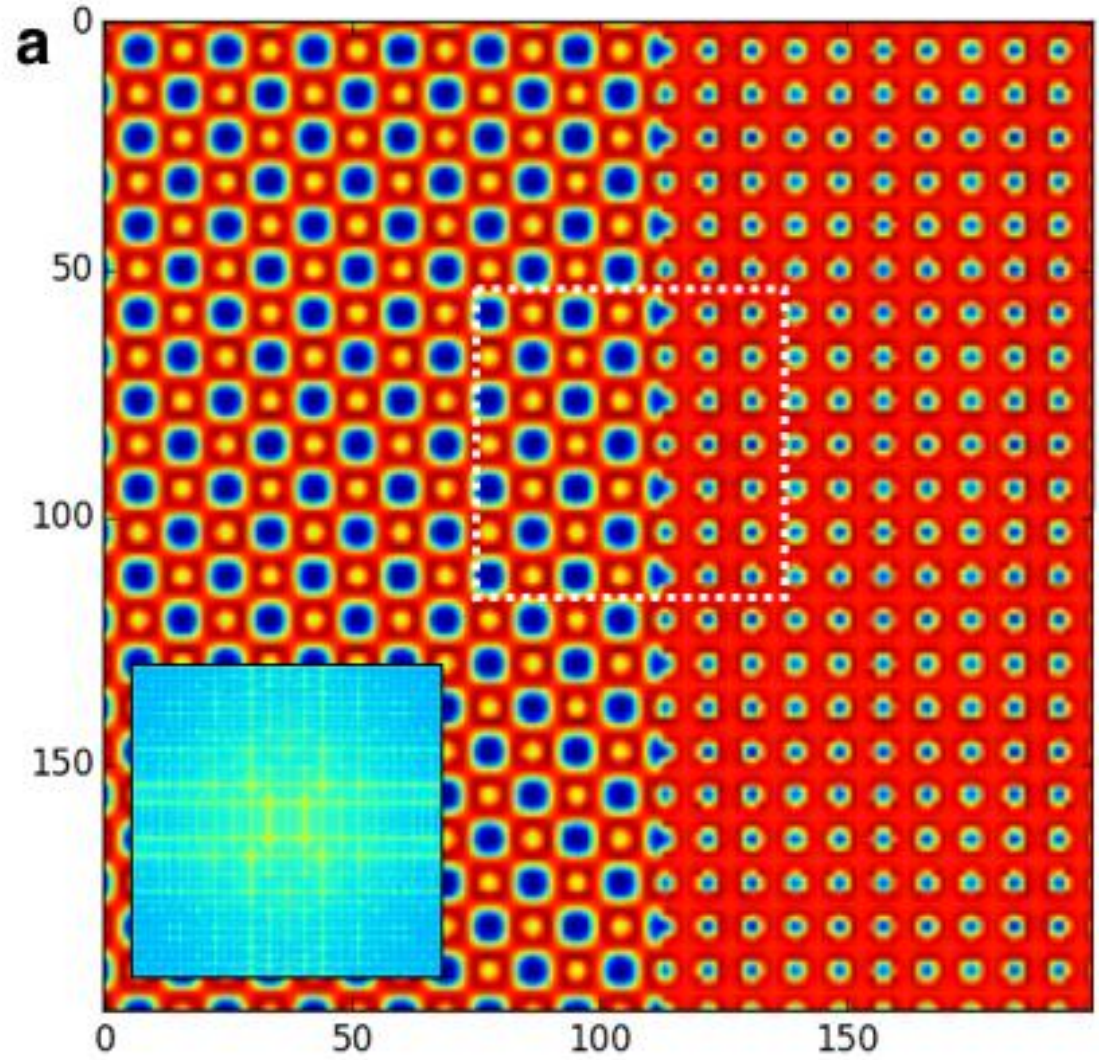
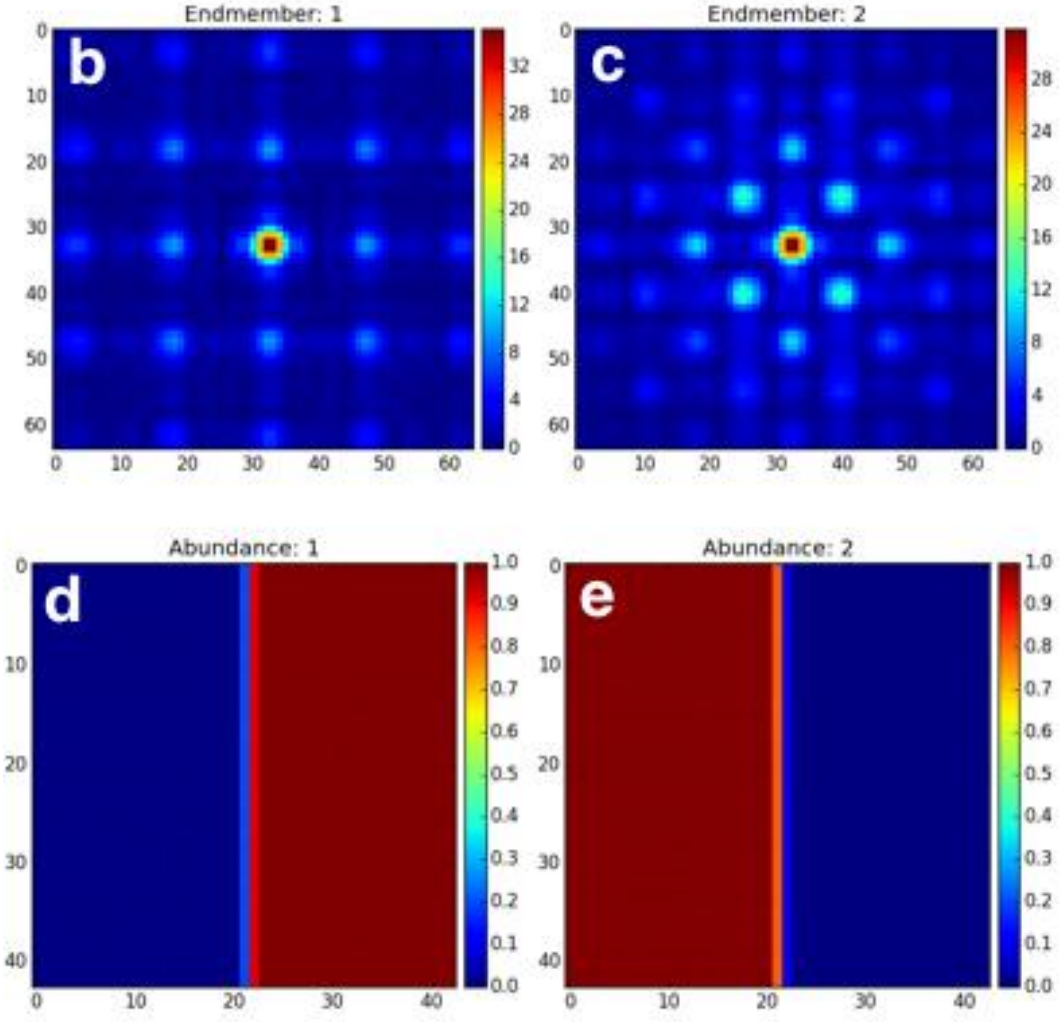


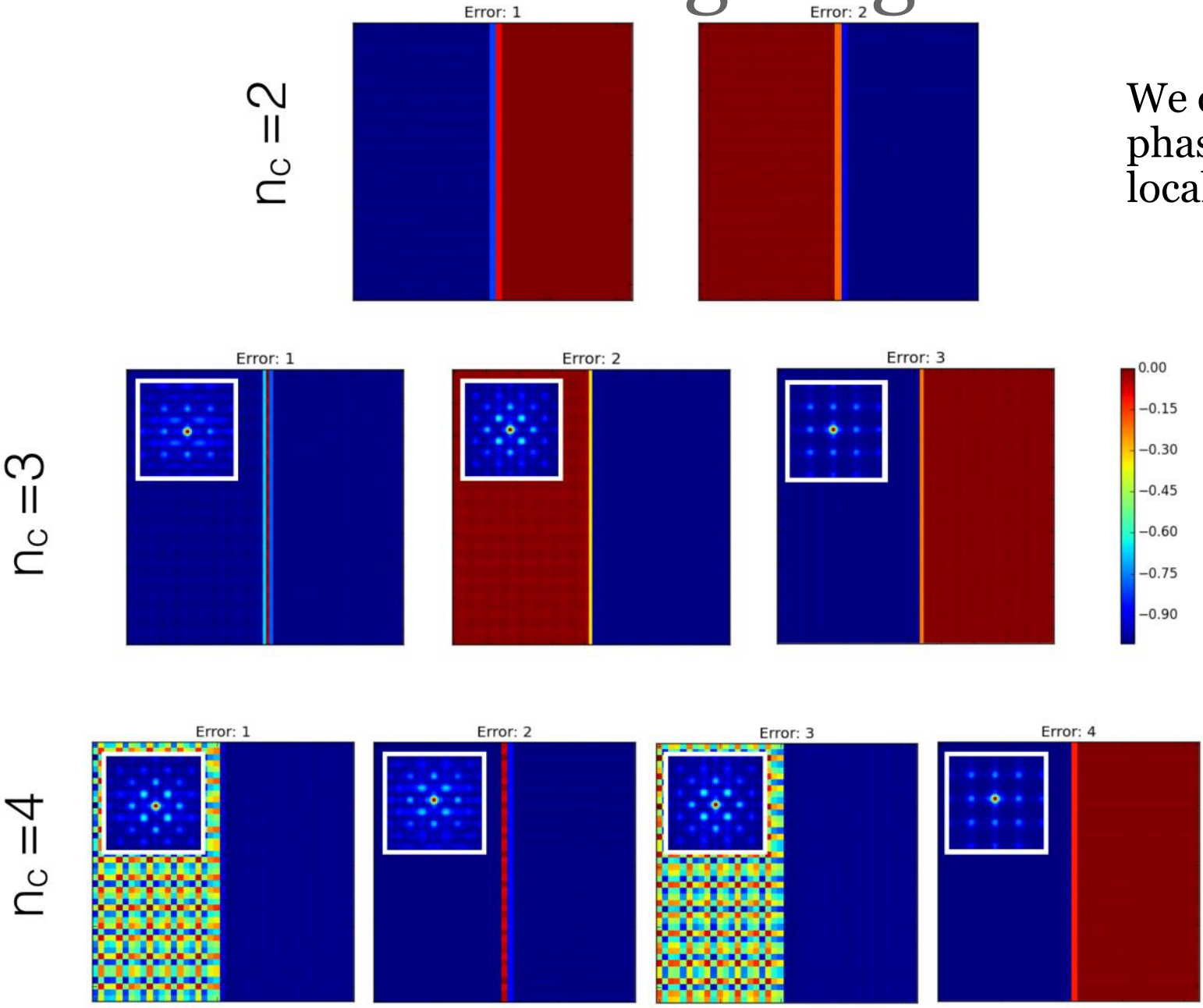
Figure by R. Vasudevan



Main idea:

- FFT amplitudes are non-negative;
- FFT removes translation

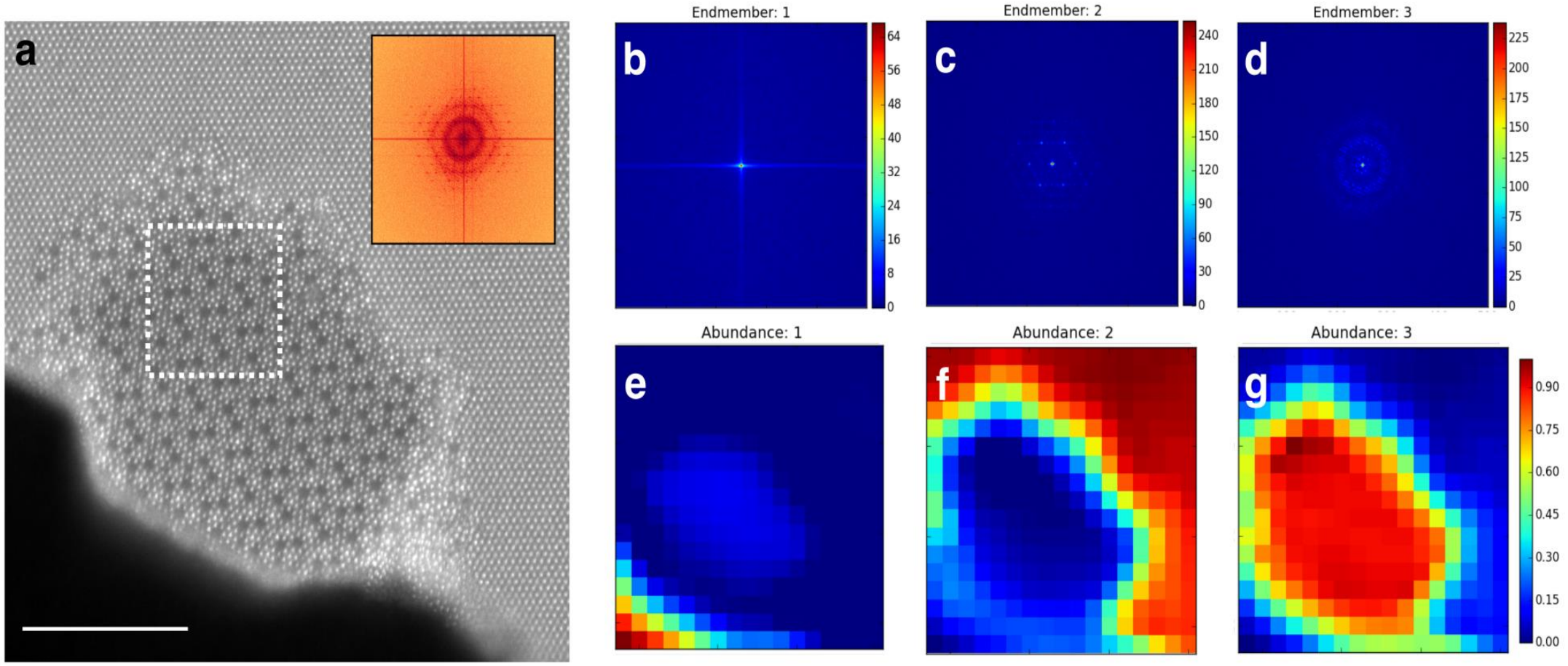
N-FINDR for image segmentation



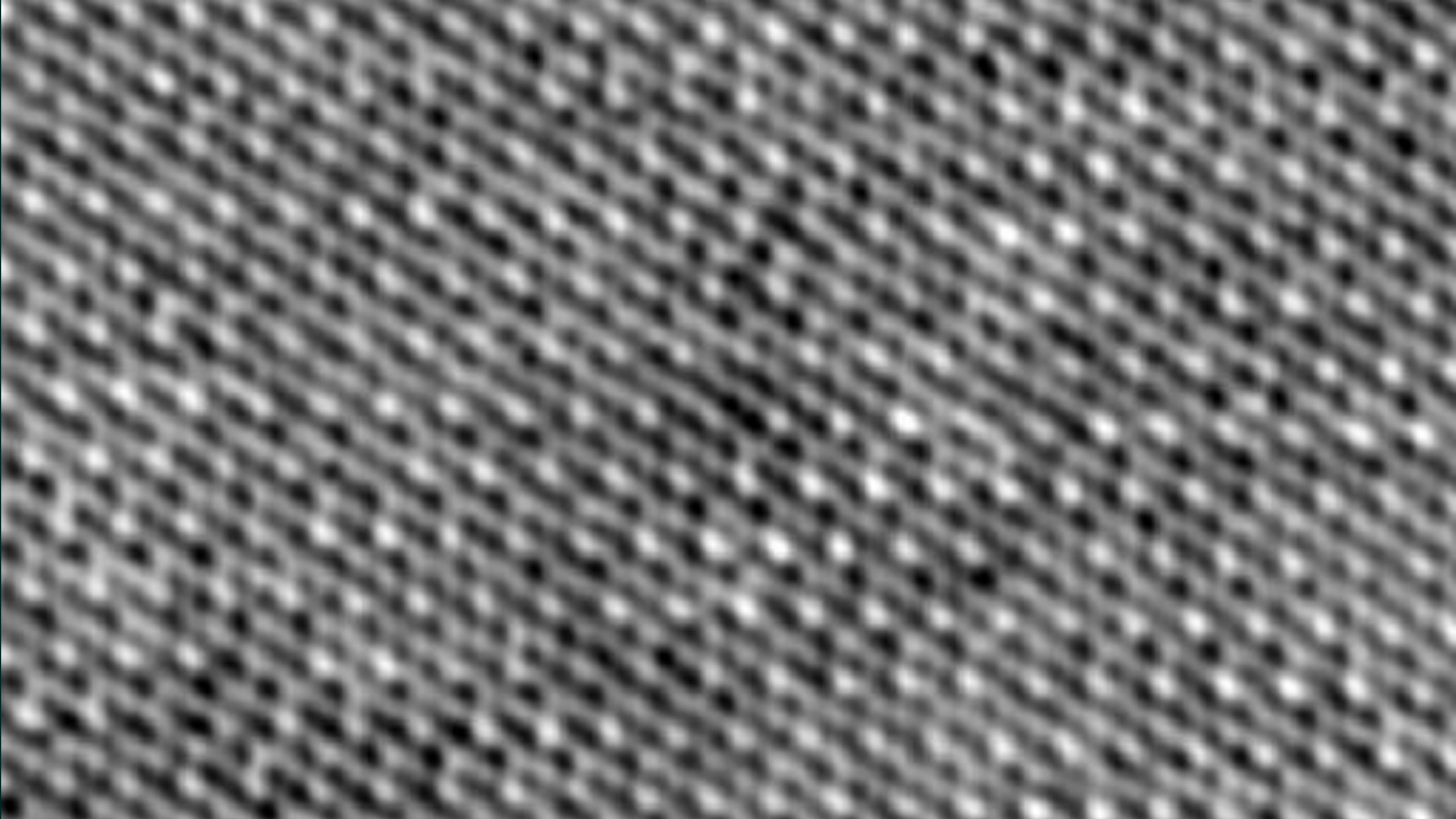
We can determine number of phases based on spatial localization and geometry

Figure by R. Vasudevan

N-FINDR for chemically separated images

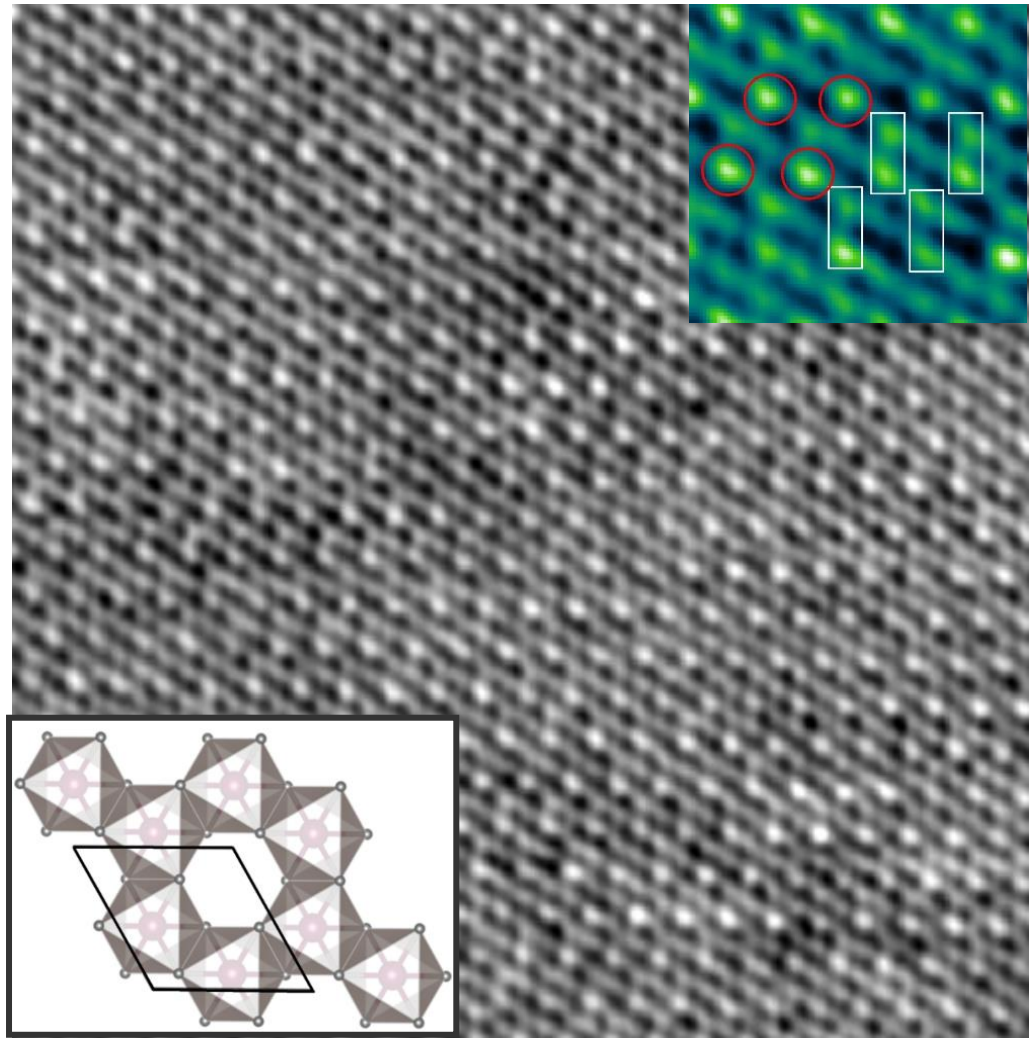


Q. He, J. Woo, A. Belianinov, V.V. Guliants, A.Y. Borisevich, *Better catalysts through microscopy: mesoscale M1/M2 intergrowth in Molybdenum–Vanadium based complex oxide catalysts for propane ammoxidation*, ACS Nano 9, 3470-3478

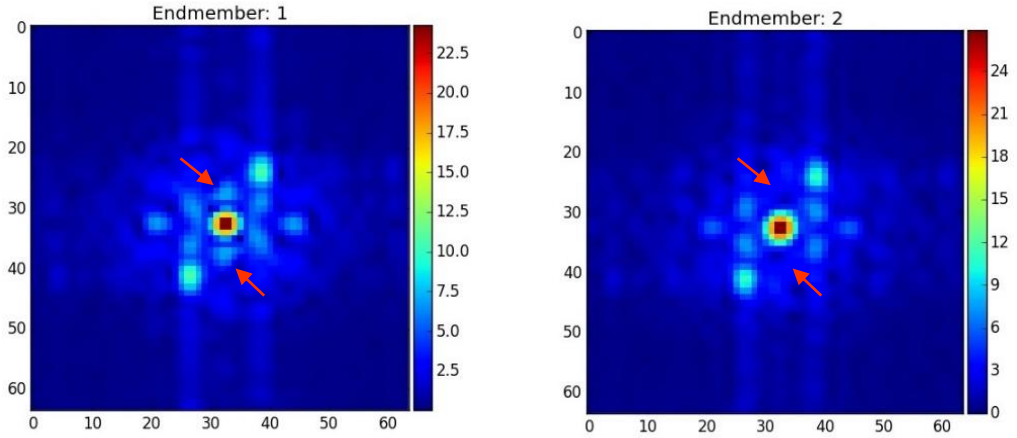


NFINDR for coexisting order parameters

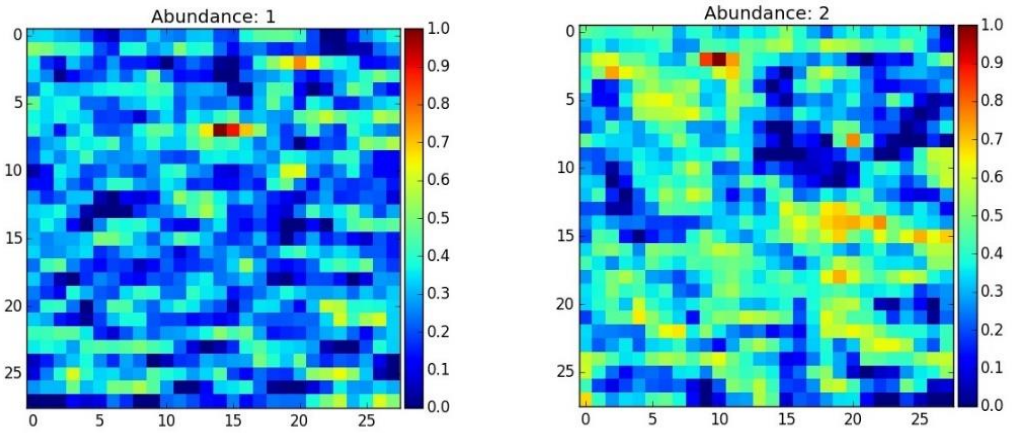
Input experimental image



FFT endmembers



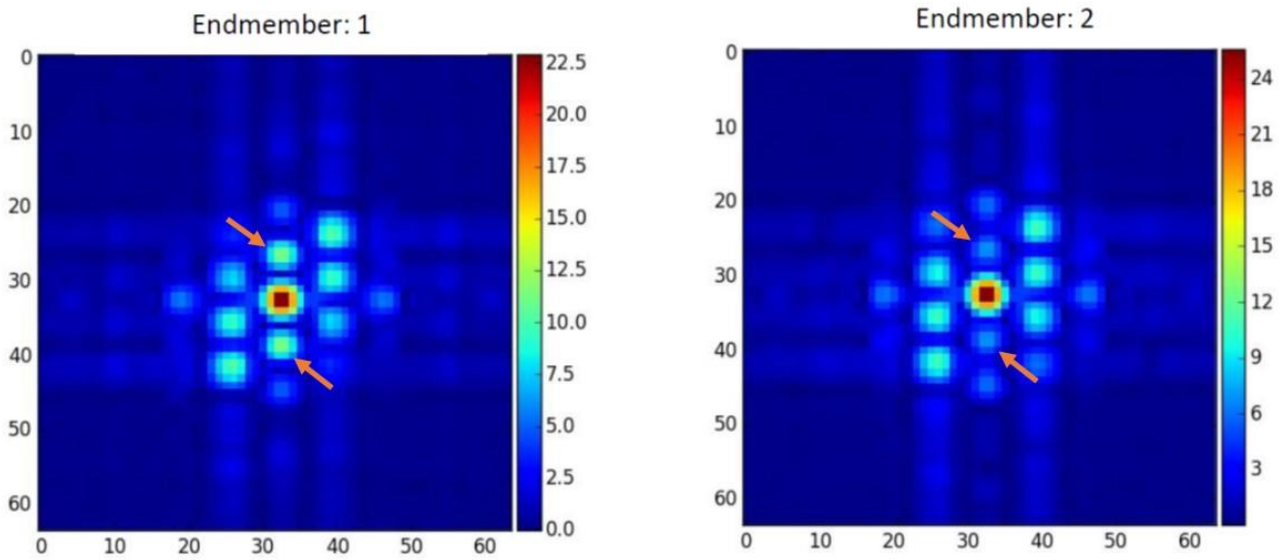
Real-space abundance maps



In a good agreement with test case, 2 spots in the “inner hexagon” are strongly suppressed in the 2nd component reflecting a fine structure of charge ordered pattern

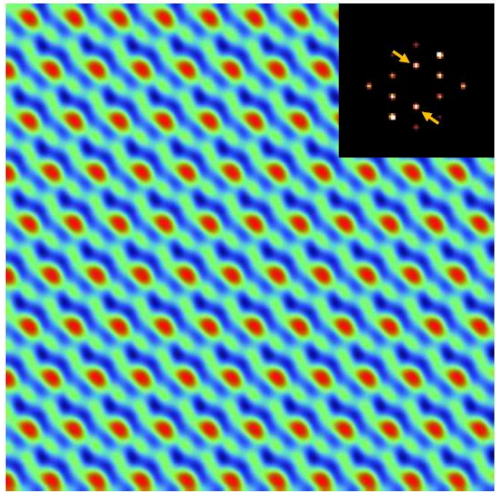
NFIND-R for coexisting order parameters

FFT endmembers

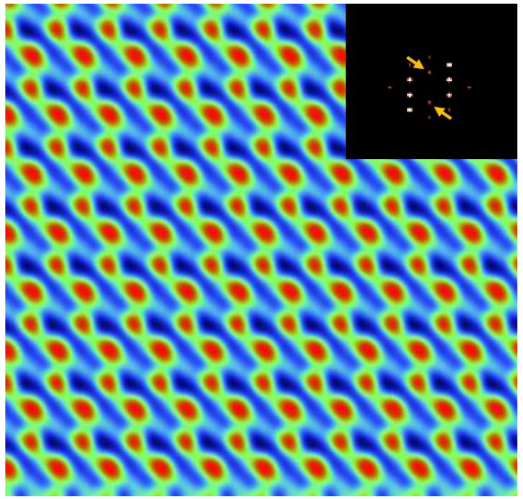


In the 2nd component, 2 spots in the "inner hexagon" are strongly suppressed reflecting a fine structure of charge ordered pattern

Real-space images of corresponding phases



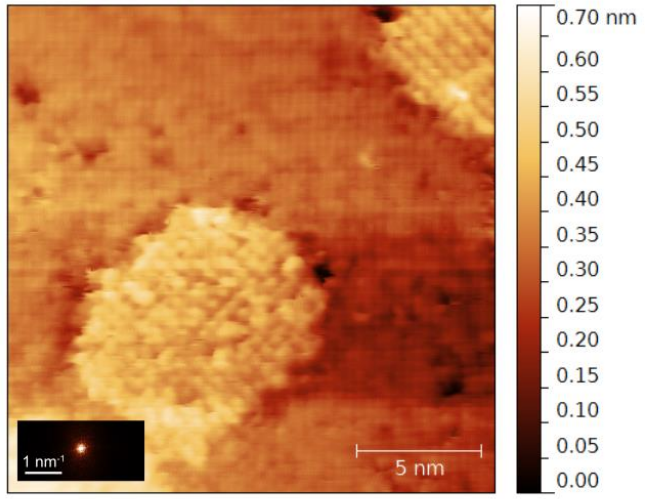
Hexagonal superlattice

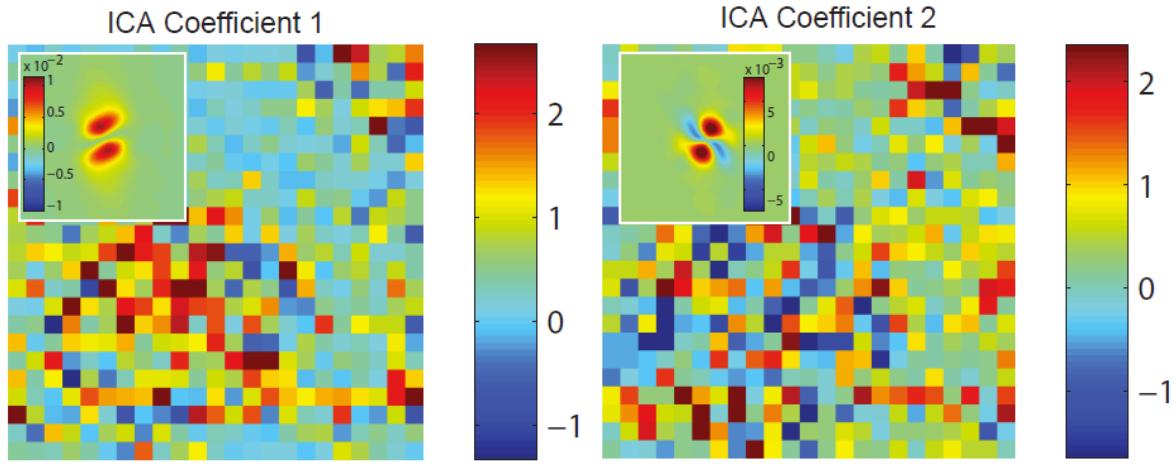


Dimer superlattice

M. Ziatdinov, A. Banerjee, A. Maksov, T. Berlijn, W. Zhou, H.B. Cao, J.Q. Yan, C.A. Bridges, D.G. Mandrus, S.E. Nagler, A.P. Baddorf, and S.V. Kalinin, *Atomic-scale observation of structural and electronic orders in the layered compound* α-$RuCl_3$, Nature Comm. 7, 13774 (2016).

ICA on Sliding FFT





- Appears to separate into pairs of components
- Unsuitable to the physics of the problem

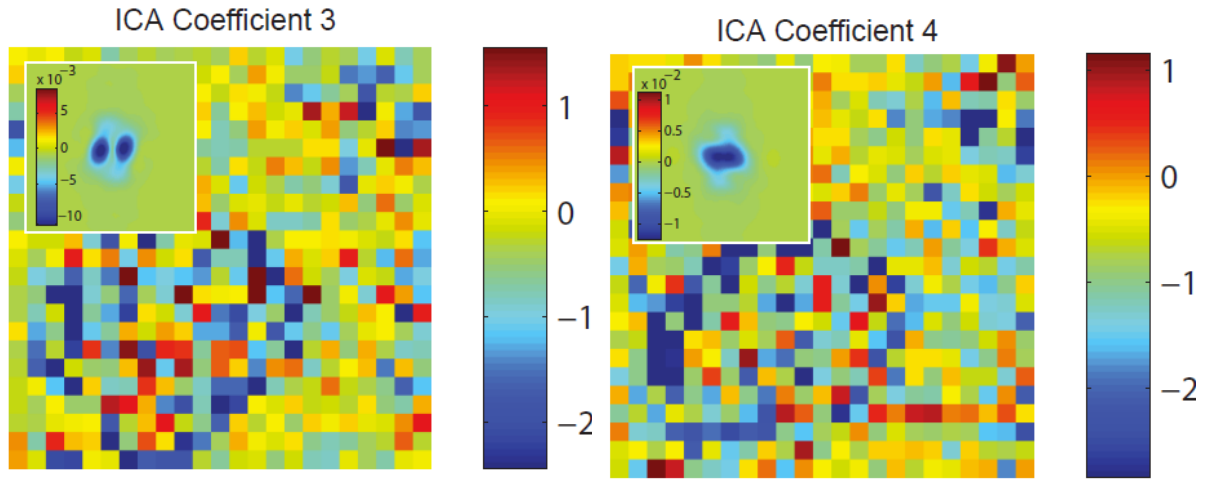
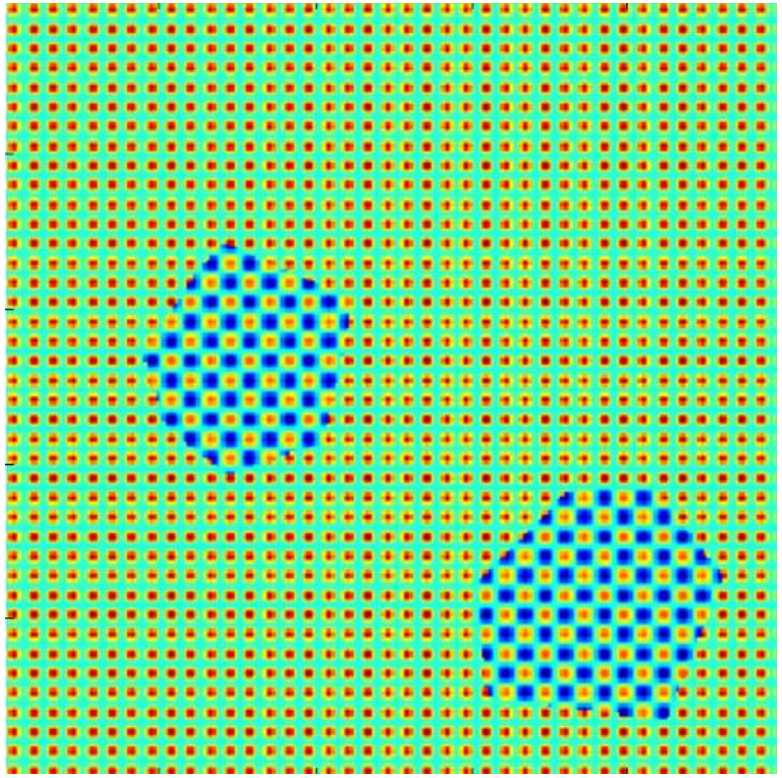
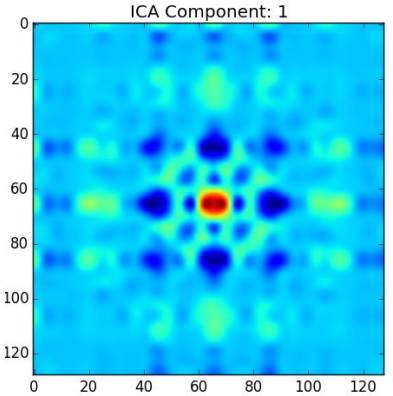
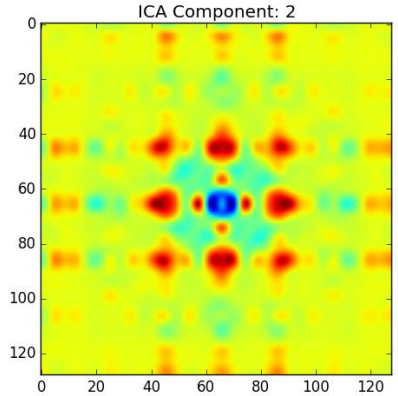


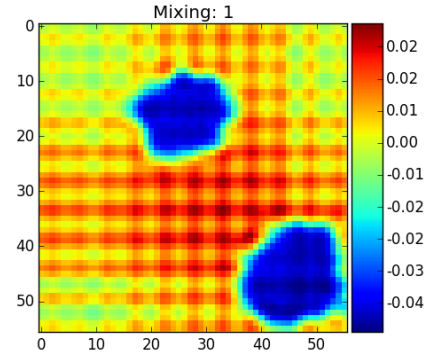
Figure by R. Vasudevan

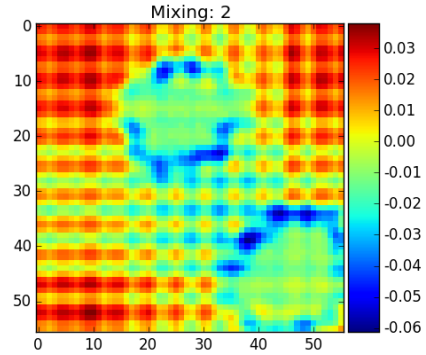
NFIND-R vs. ICA

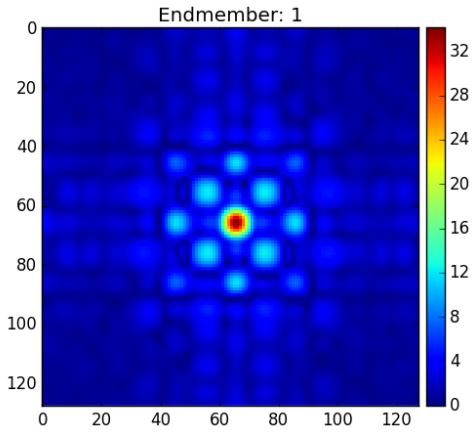


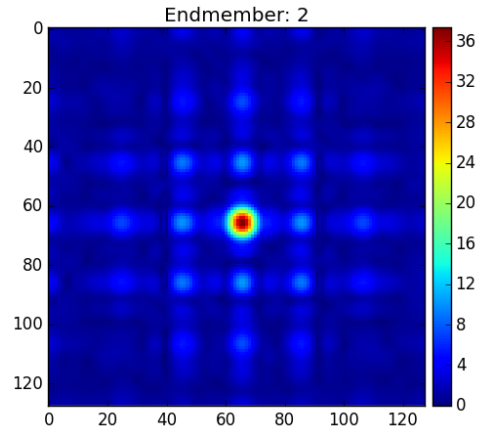


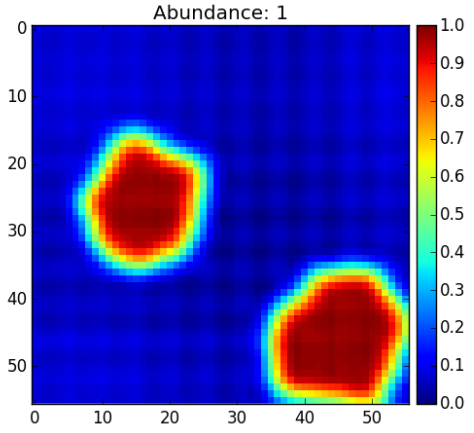












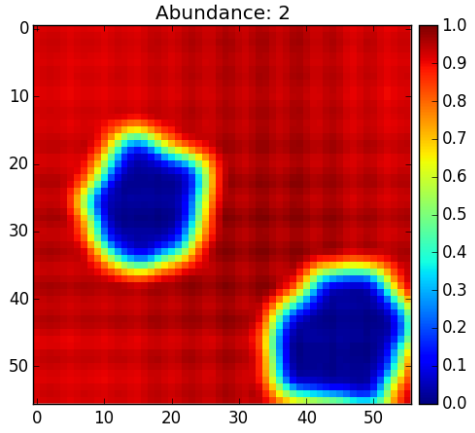


Figure by
R. Vasudevan

Sliding FFT:

- We always have a problem of window size:
  - too large – loose spatial resolution,
  - too small – FFT behaves poorly due to edge effects
- Interpretation of FFT data is complicated (too much data if fit each peak, unclear meaning of the unmixing components)
- Natural descriptor for atomically resolved images – atomic coordinates!

Local crystallography

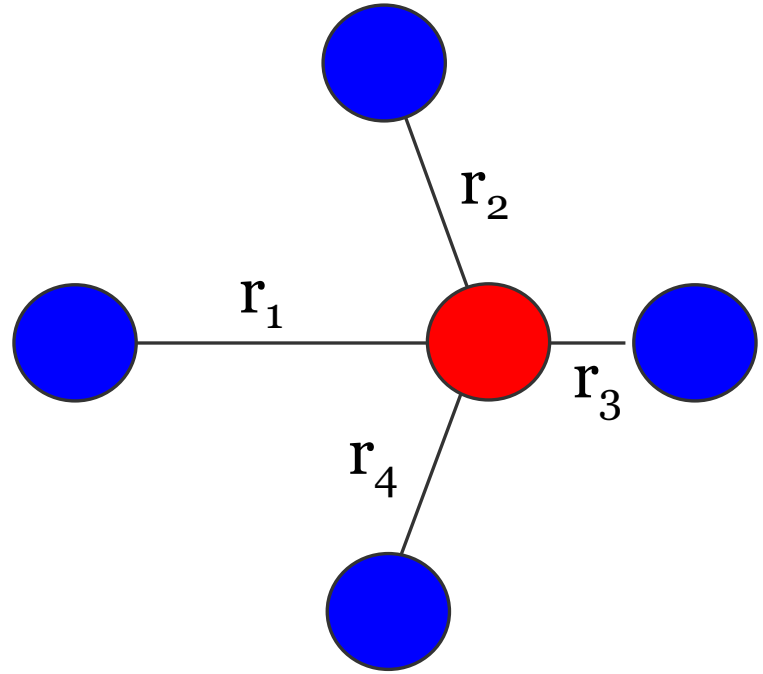
For each atom, define nearest neighbors and generate array of the corresponding radius-vectors of the form

$NA_{ij} = (rx_1, ry_1, rx_2, ry_2, rx_3, ry_3, rx_4, ry_4)_{ij}$

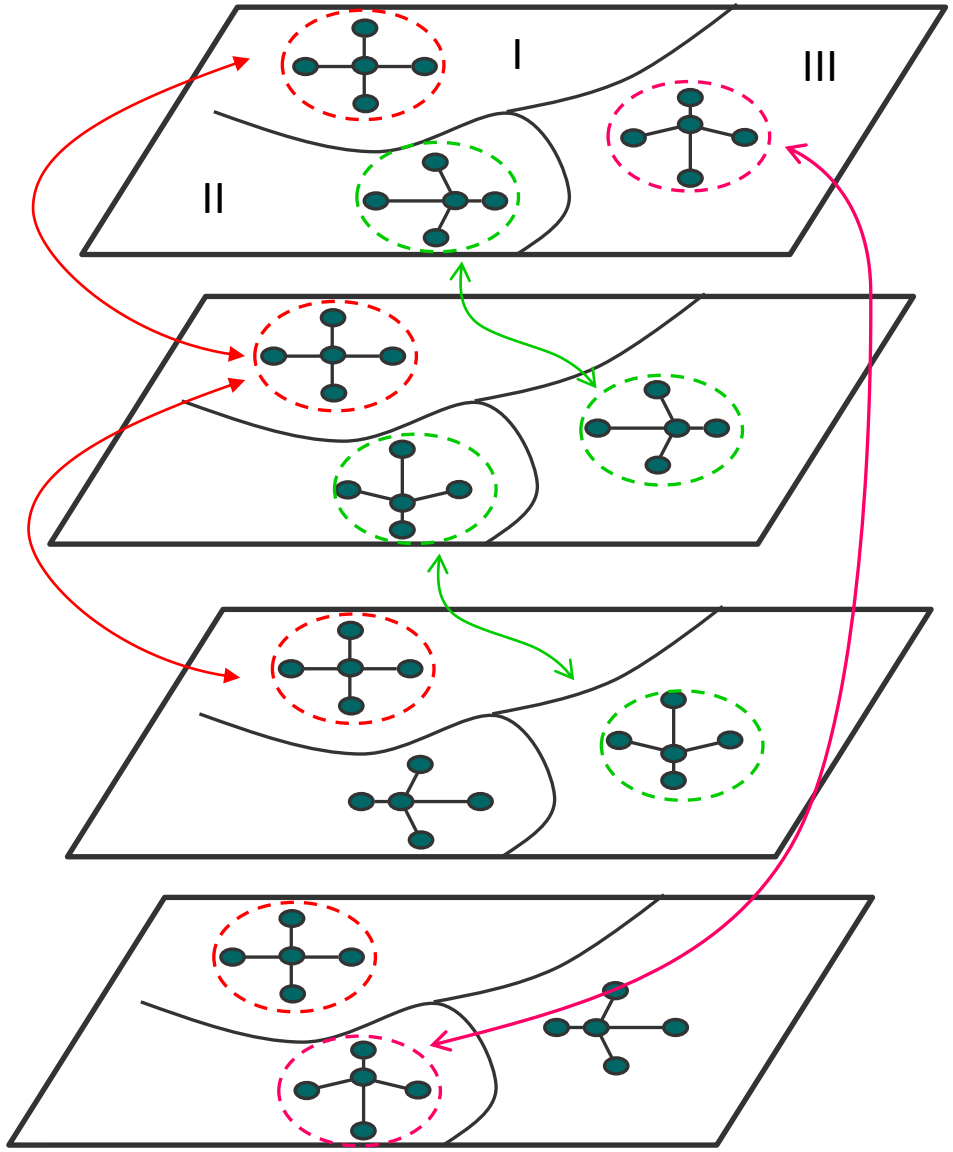
Indexes 1,2,3,4 are chosen in the same sense for all atoms (generalization for different lattice and/or next coordination sphere obvious)

Then, phase/ferroic variant identification problem can be reduced to finding equivalent (in statistical sense) groups of nearest neighbors

We can also use group theory to make hypotheses, e.g. add translation symmetry operations, i.e. i -> i+1 and j -> j+1 for lattice doubling)



Local crystallography



A

A’

A’’

Same cluster in all replicas:
Non-ferroic phase

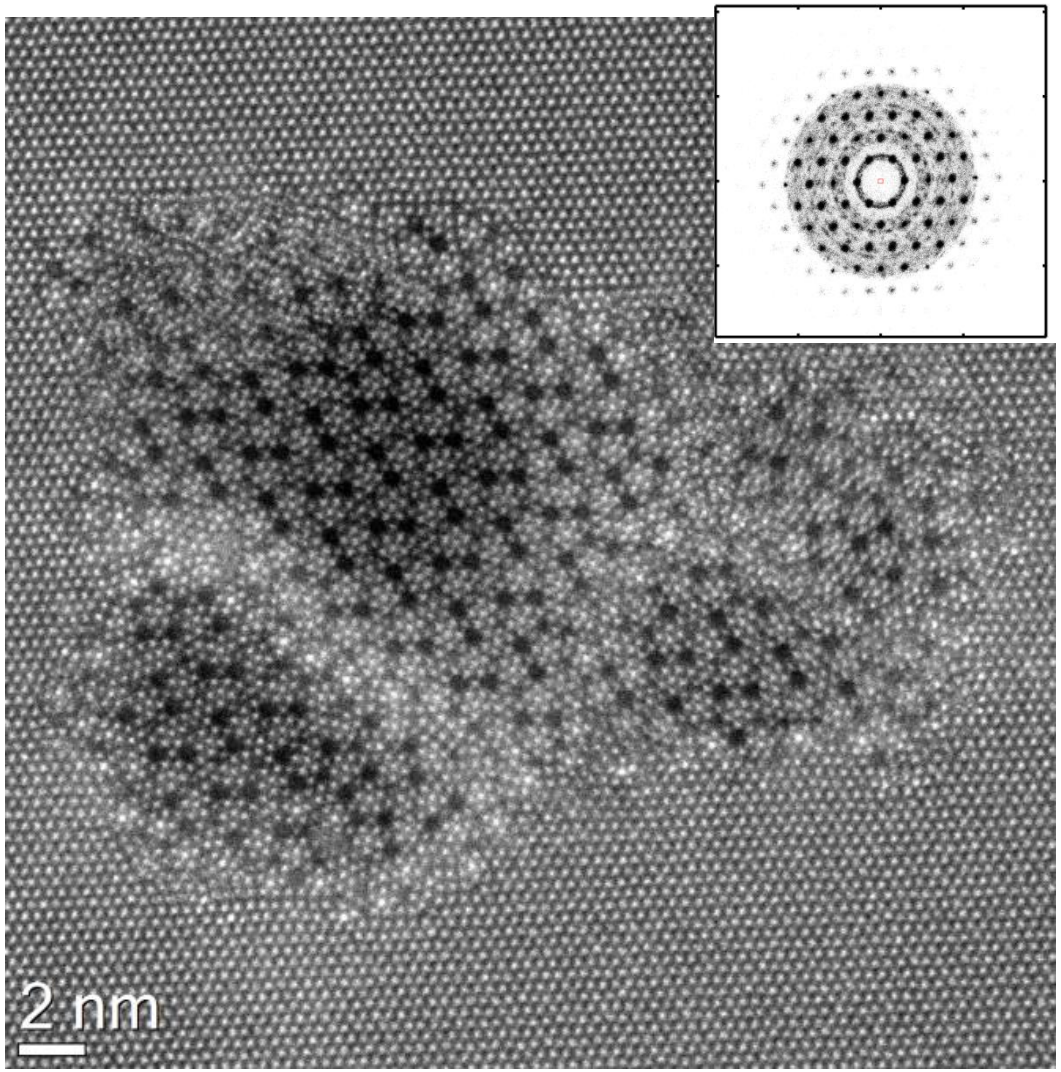
Form cluster with different regions in different replicas:
Ferroic phase

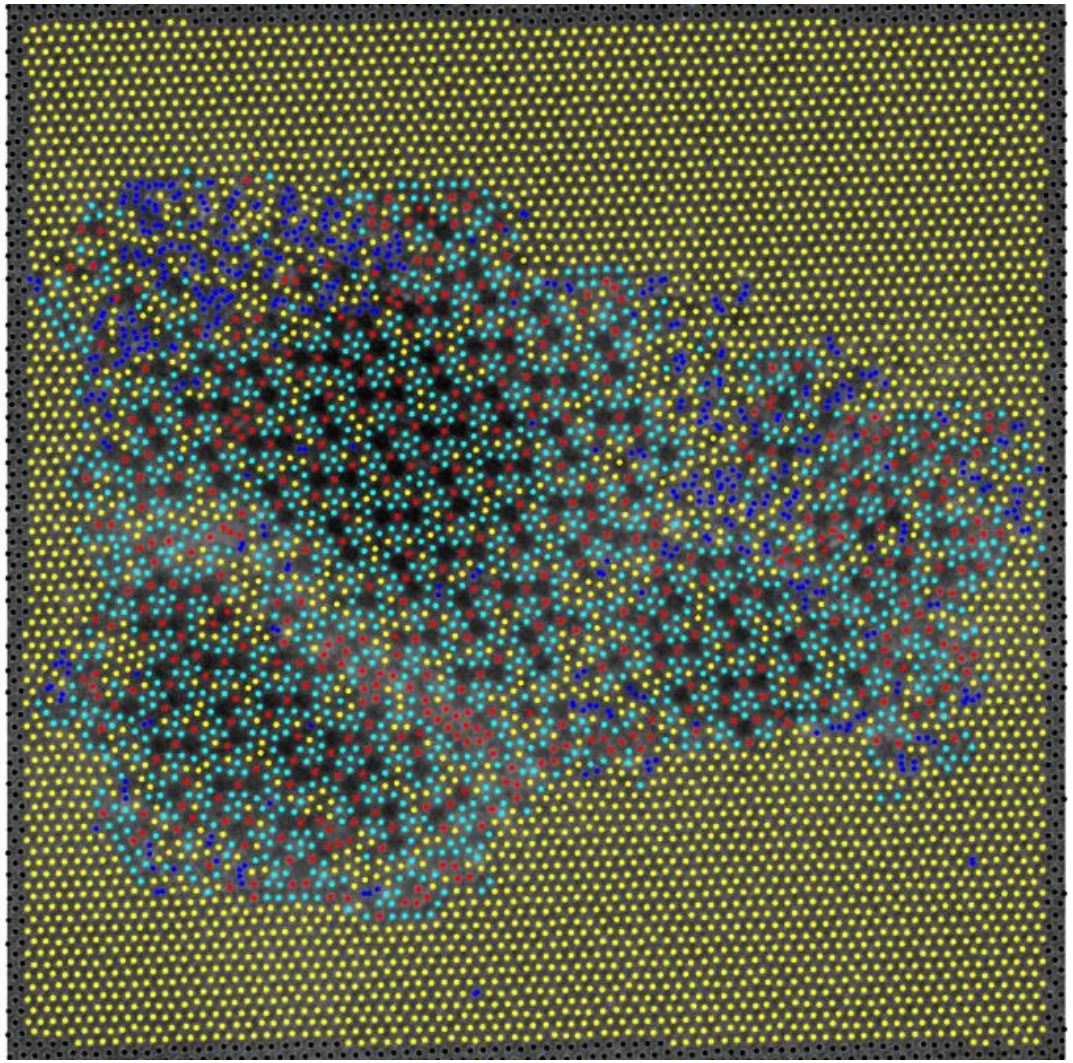
Only some of the correspondences are shown (but these are obvious)

A. Belianinov, Q. He, M. Kravchenko, S. Jesse, A. Borisevich, and S.V. Kalinin, *Identification of phases, symmetries, and defects through local crystallography*, Nat. Comm. **6**, 7801 (2015).

W. Lin, Q. Li, A. Belianinov, B.C. Sales, A. Sefat, Z. Gai, A.P. Baddorf, M. Pan, S. Jesse, and S.V. Kalinin, *Local crystallography analysis for atomically resolved scanning tunneling microscopy images*, Nanotechnology **24**, 415707 (2013).

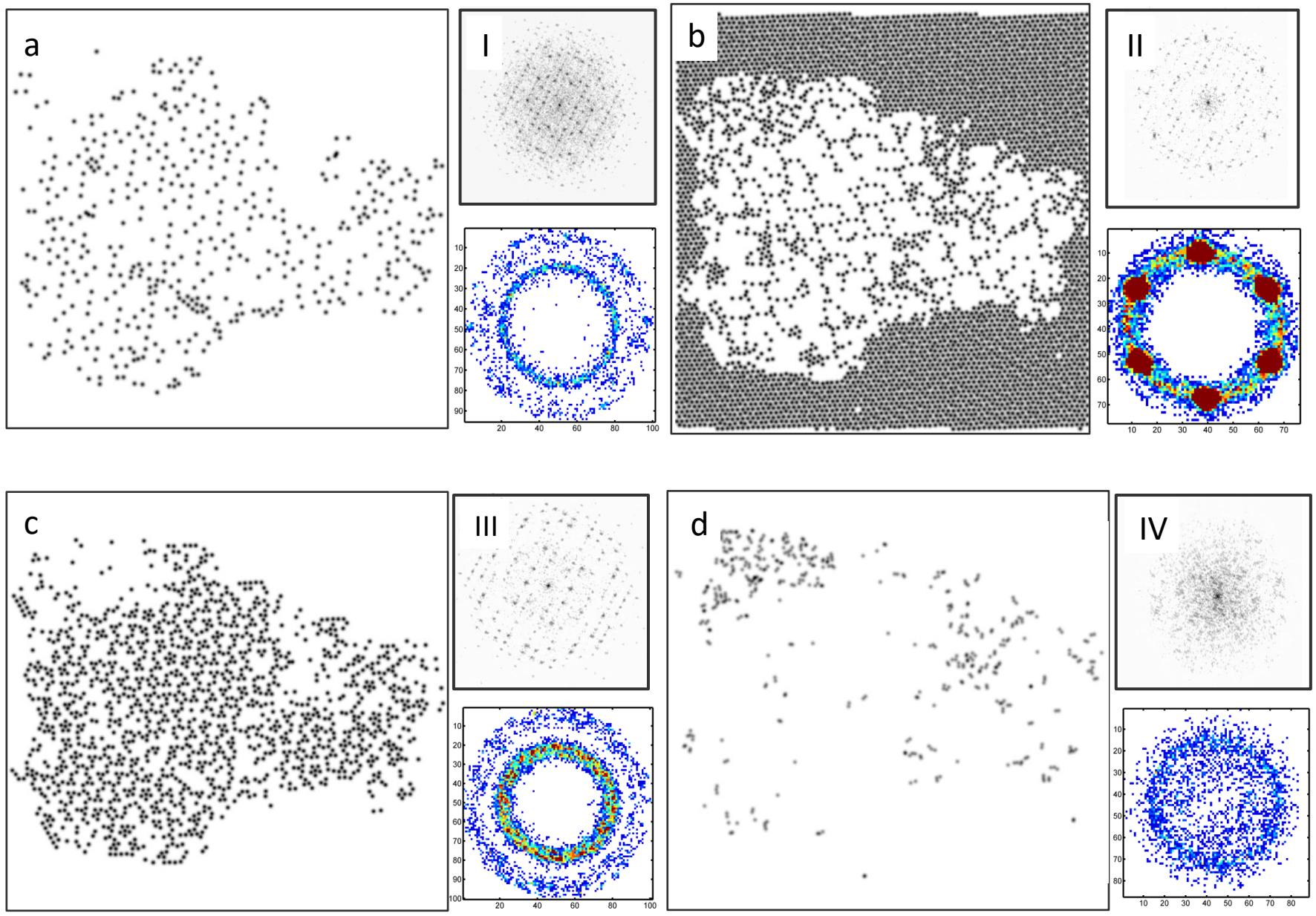
Local crystallography: k-means

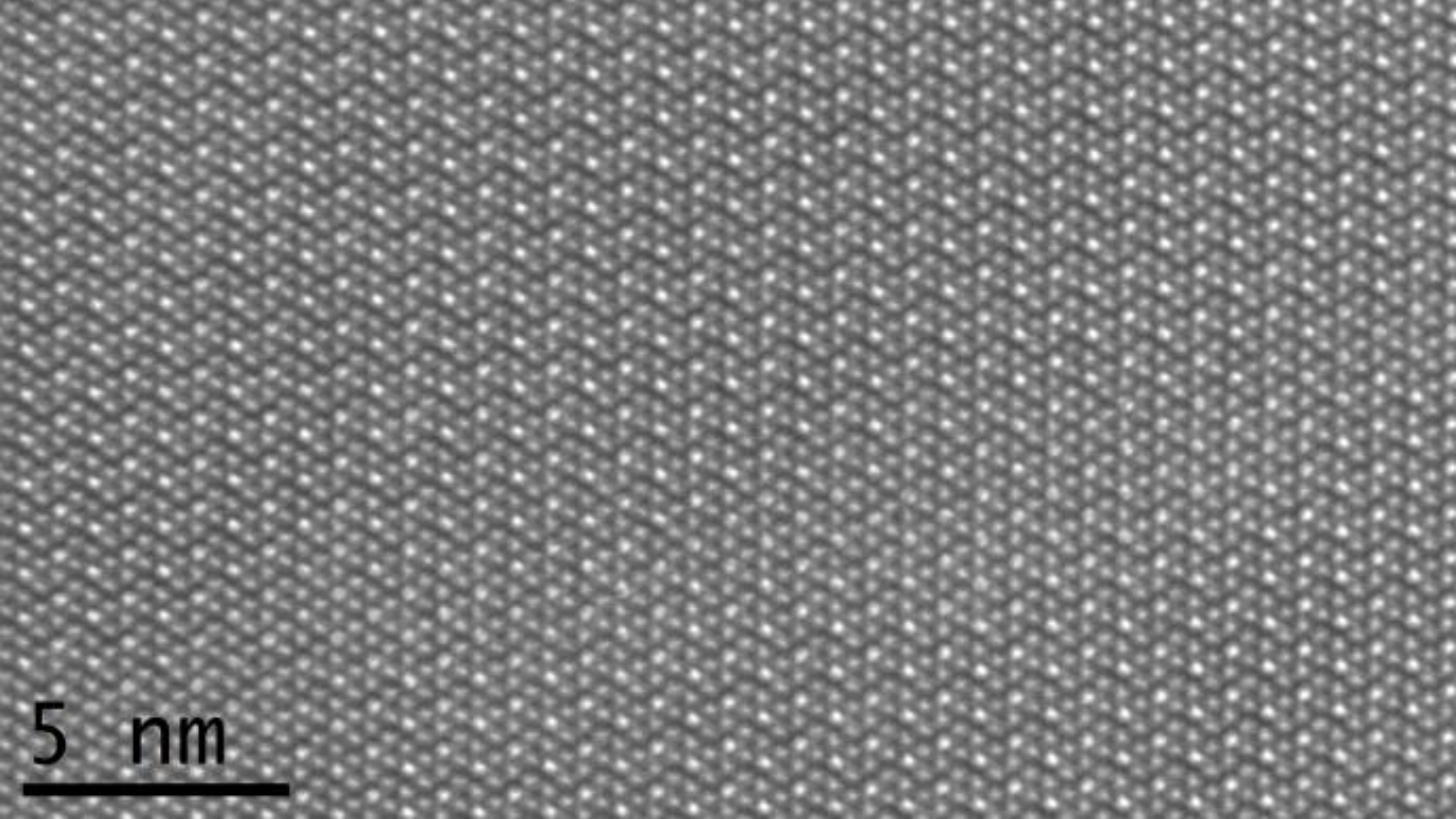




A. BELIANINOV, Q. HE, M. KRAVCHENKO, S. JESSE, A. BORISEVICH, and S.V. KALININ, *Identification of phases, symmetries, and defects through local crystallography*, Nat. Comm. **6**, 7801 (2015).

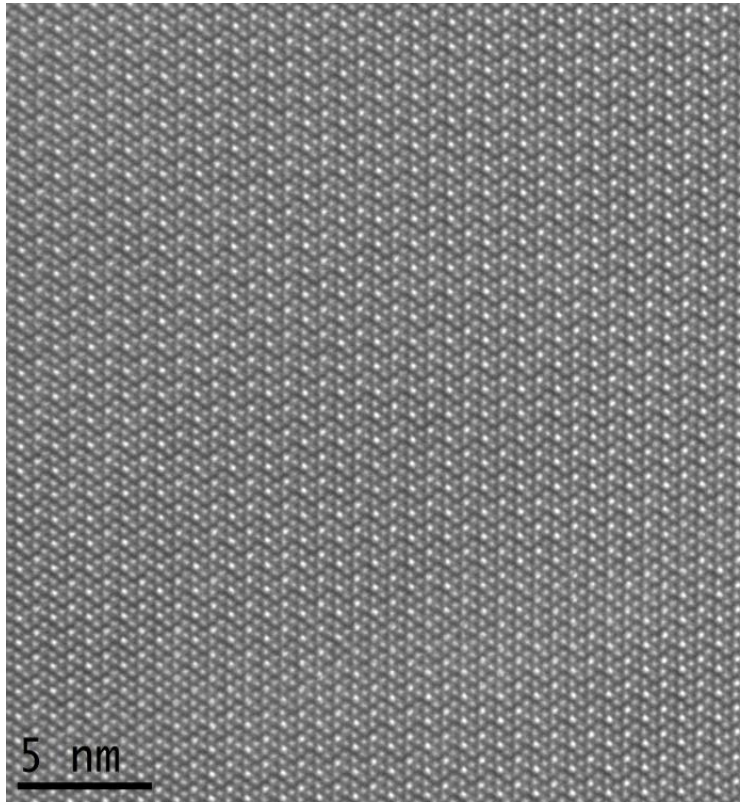
Local crystallography



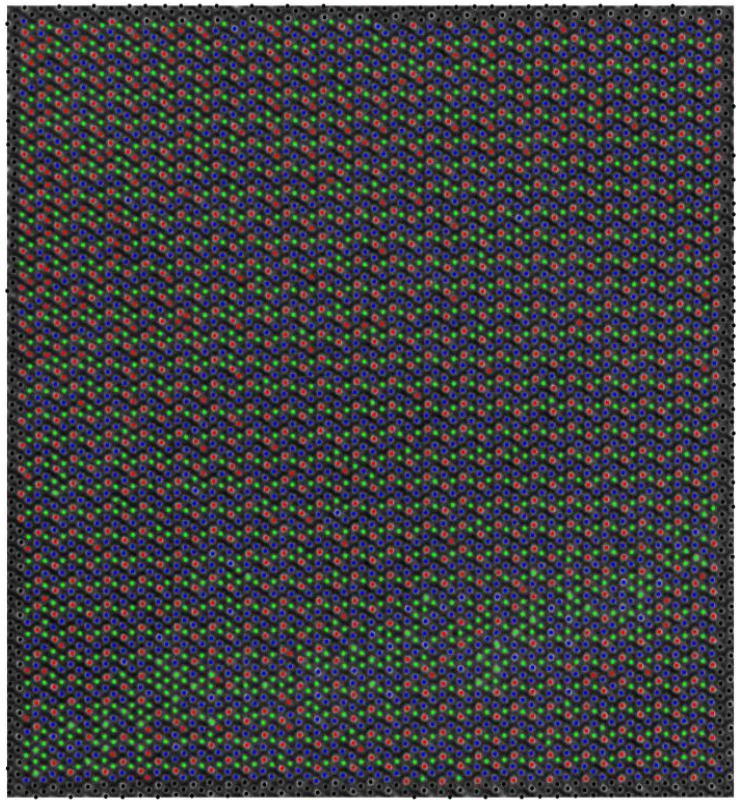

5 nm

Local crystallography

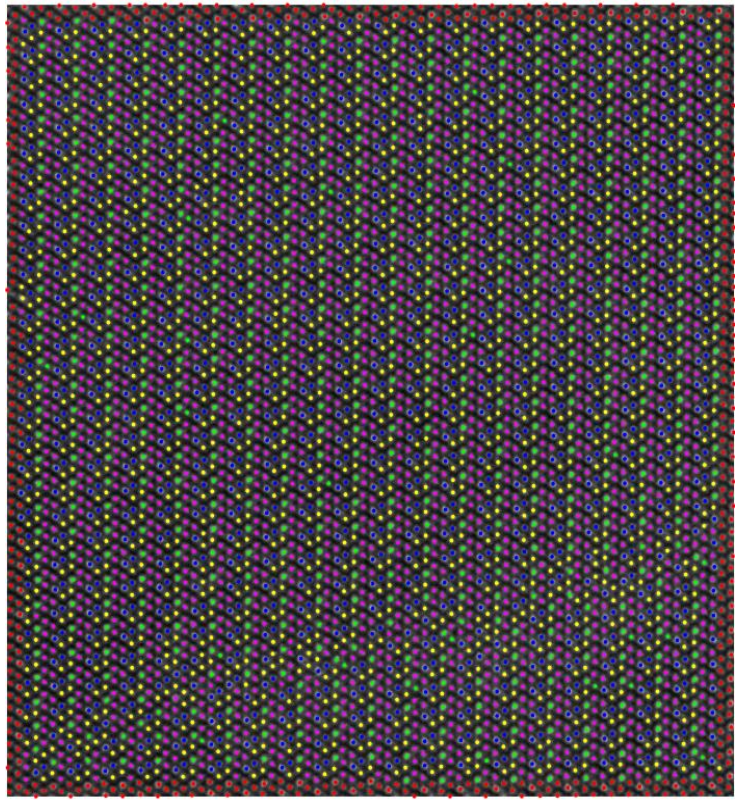
Image



K-means full vector



K-means angles



Normal modes: group theory

- **Group Representations** map group elements onto matrices, ensuring matrix multiplication aligns with the group operation.
- Molecules have **symmetries** defined by point groups linked to symmetry operations, such as rotations.
- Molecules have various **vibrational modes** with specific symmetries. Using the molecule's point group, one can deduce which modes are spectroscopically active.
- **Selection Rules:** Group representations dictate which vibrational modes appear in techniques like IR or Raman. Some modes may be IR-active but not Raman-active based on symmetry.
- **Vibrational frequencies** indicate energy differences between vibrational levels that are connected to group representations

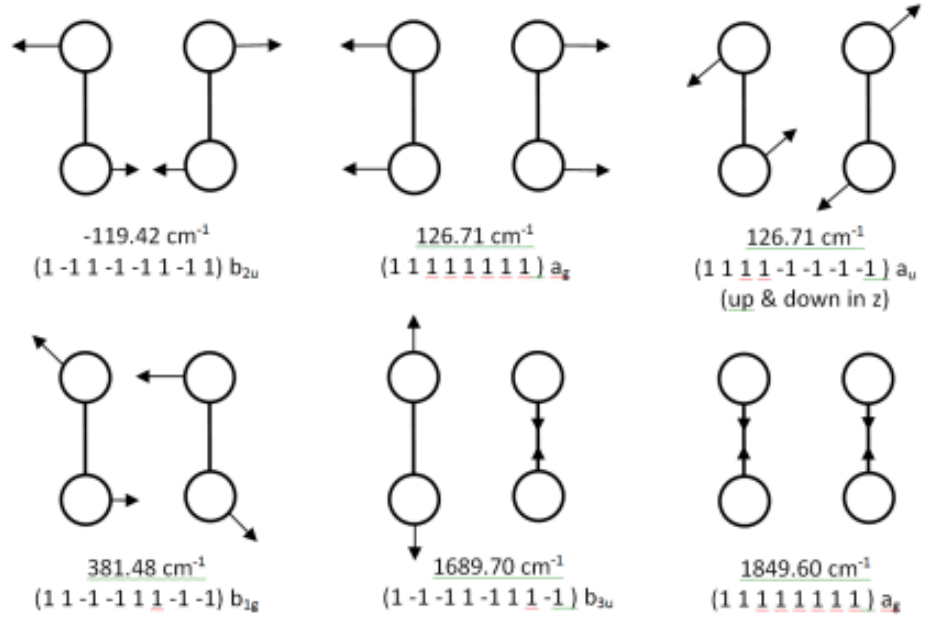


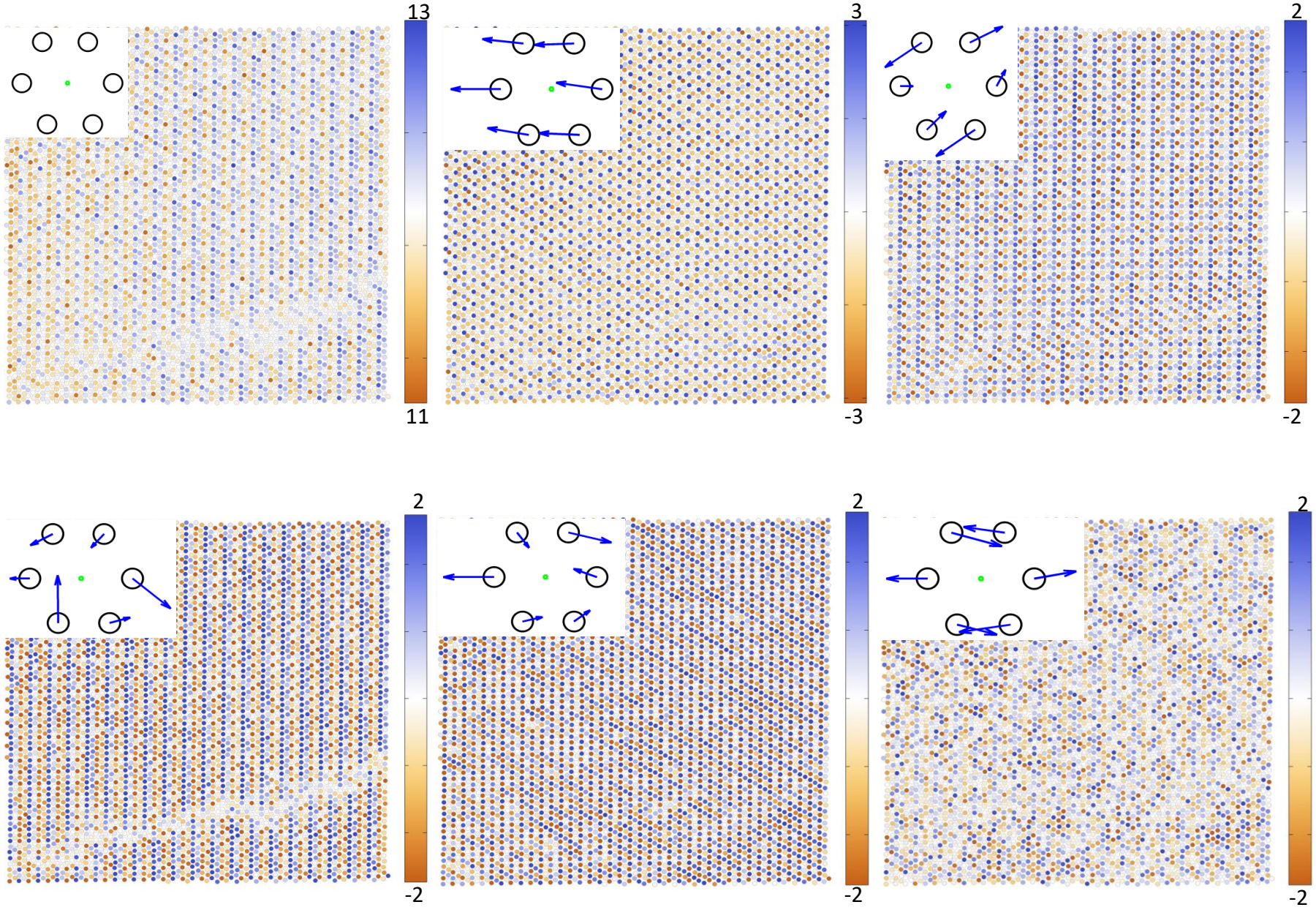
Figure 2: Sketches of Normal Modes of O_4^+

Table 2: Character Table for Point Group D_{2h}

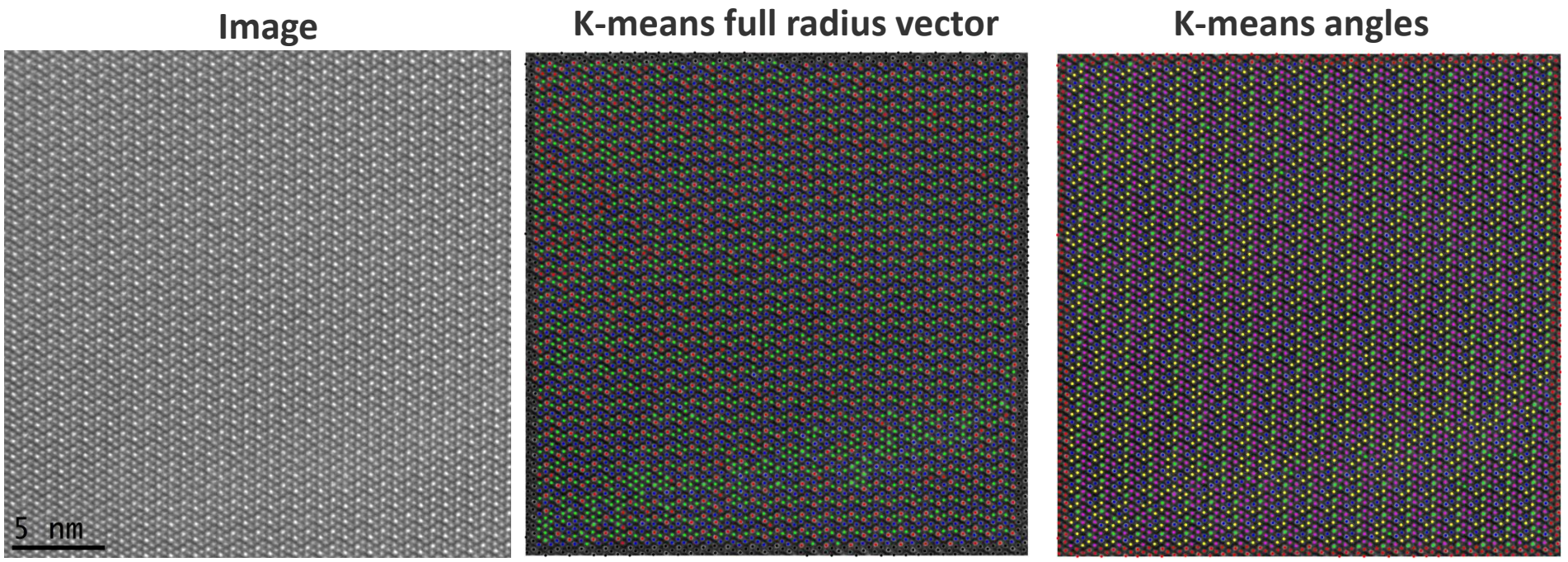
D_{2h}	E	$C_2(z)$	$C_2(y)$	$C_2(x)$	i	$\sigma(xy)$	$\sigma(xz)$	$\sigma(yz)$		
A_g	1	1	1	1	1	1	1	1		x^2, y^2, z^2
B_{1g}	1	1	-1	-1	1	1	-1	-1	R_z	xy
B_{2g}	1	-1	1	-1	1	-1	1	-1	R_y	xz
B_{3g}	1	-1	-1	1	1	-1	-1	1	R_x	yz
A_u	1	1	1	1	-1	-1	-1	-1		
B_{1u}	1	1	-1	-1	-1	-1	1	1	z	
B_{2u}	1	-1	1	-1	-1	1	-1	1	y	
B_{3u}	1	-1	-1	1	-1	1	1	-1	x	

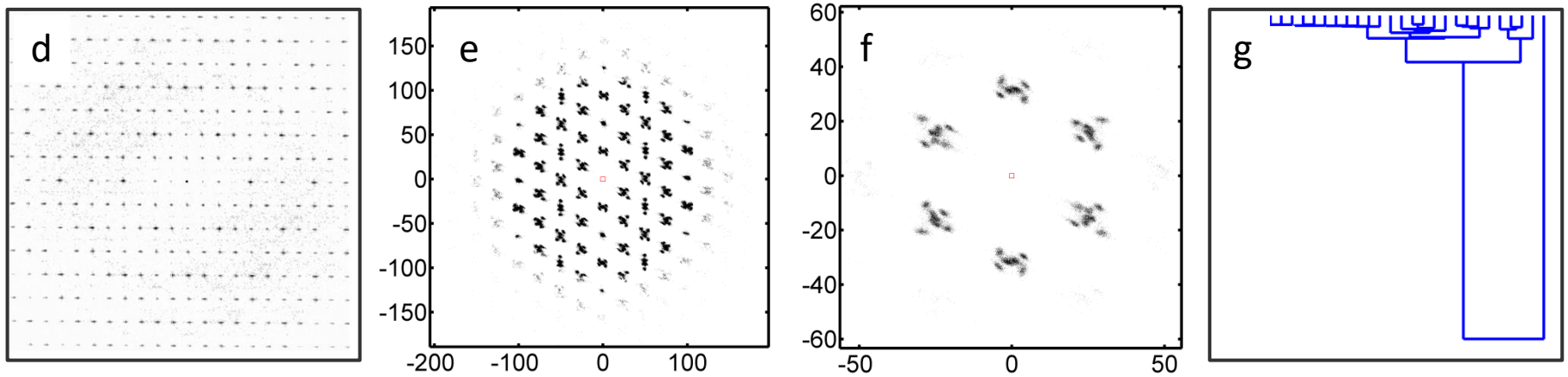
http://vergil.chemistry.gatech.edu/notes/grpthy-vib/grpthy-vib.pdf

Local crystallography: PCA



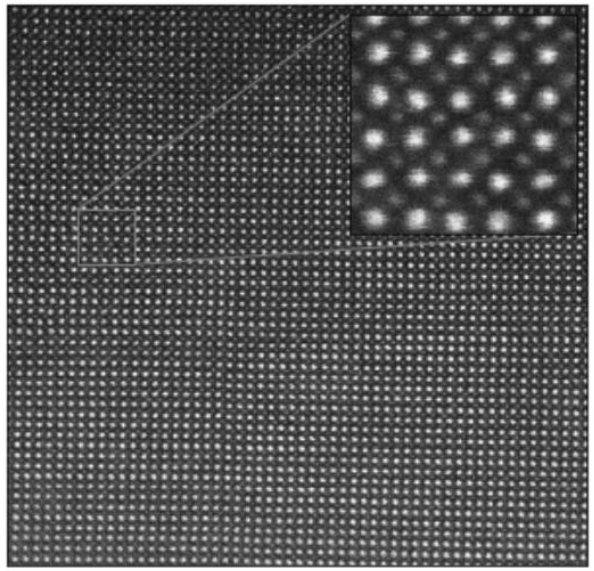
Local crystallography



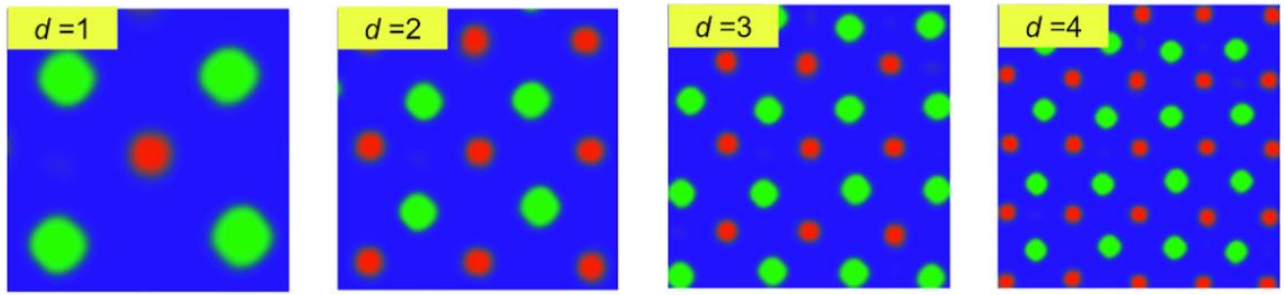


Local crystallography: FerroNET

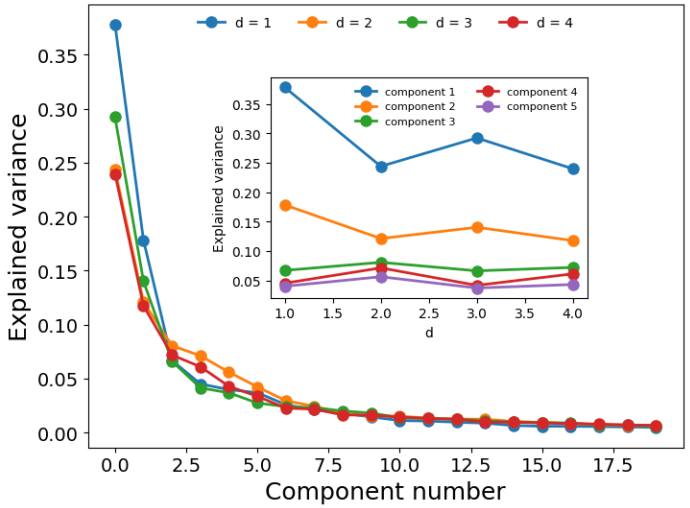
Experimental (LBFO film)



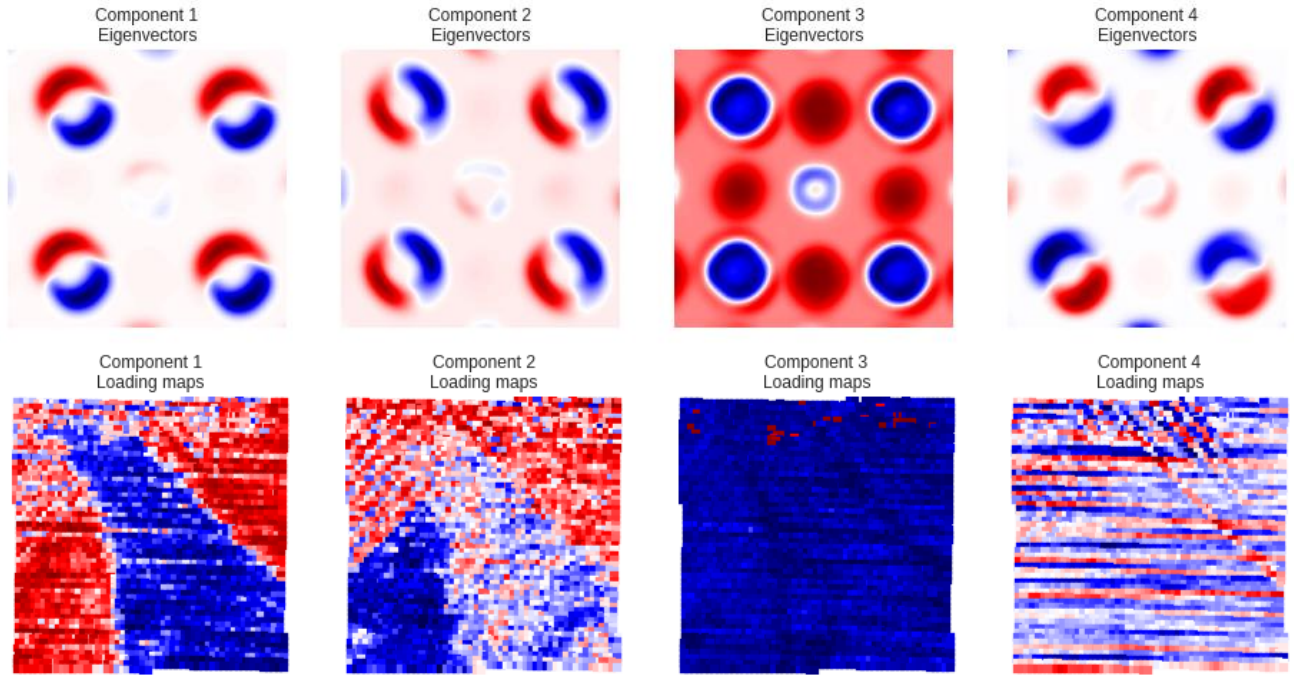
Building blocks (from neural network output)



Information Content

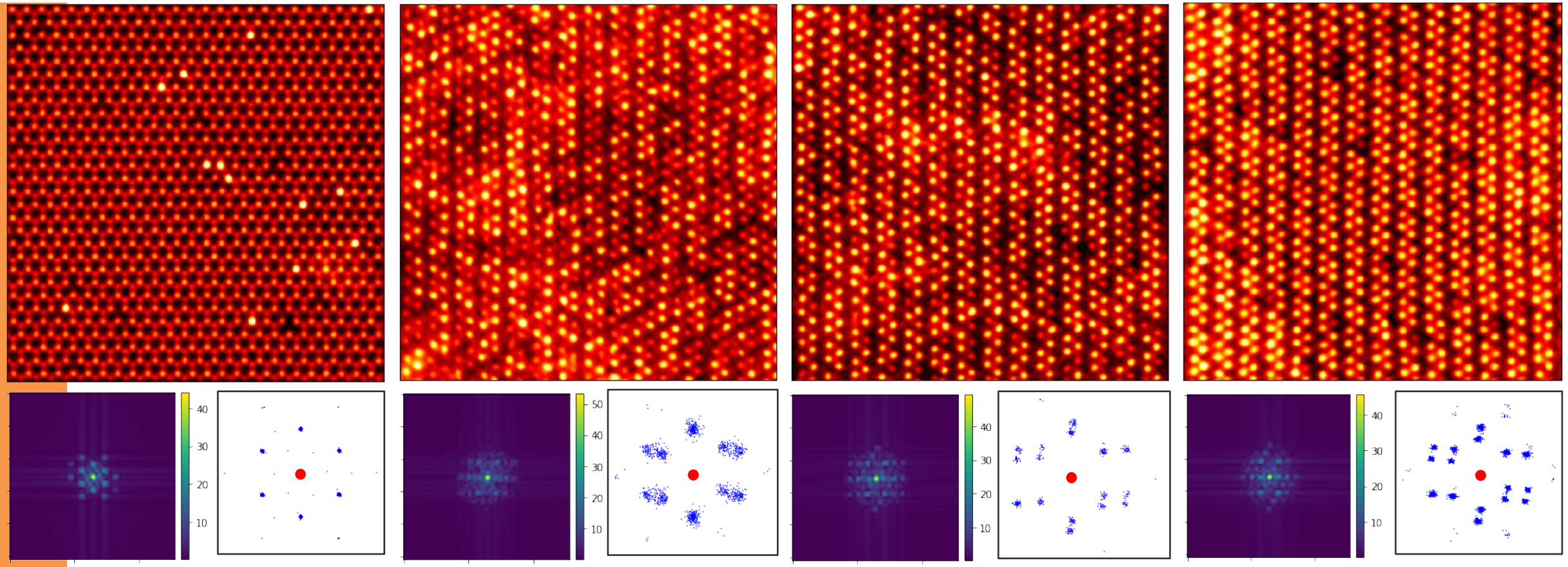


PCA eigenvectors and loading maps



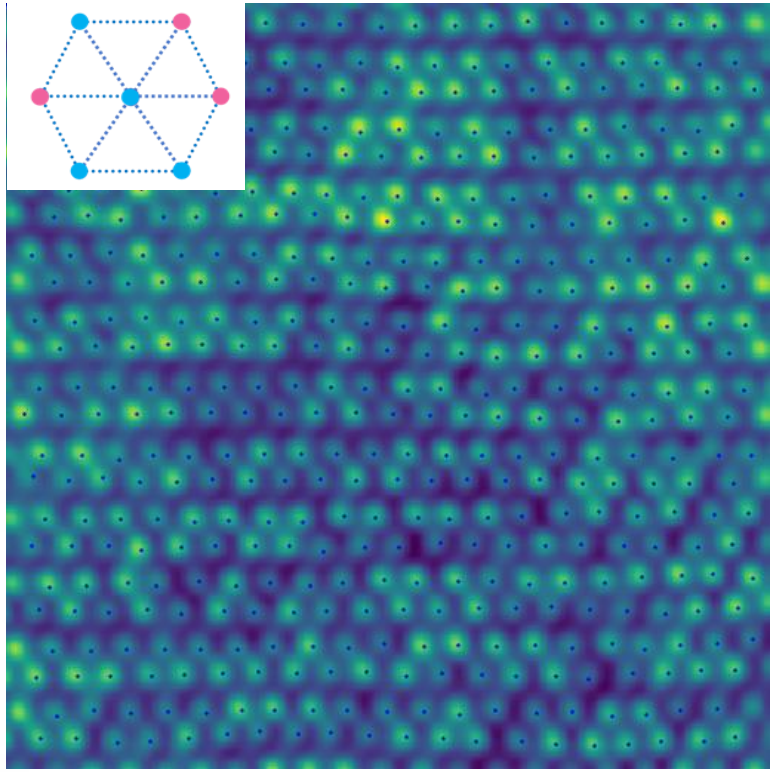
MoS_2-ReS_2

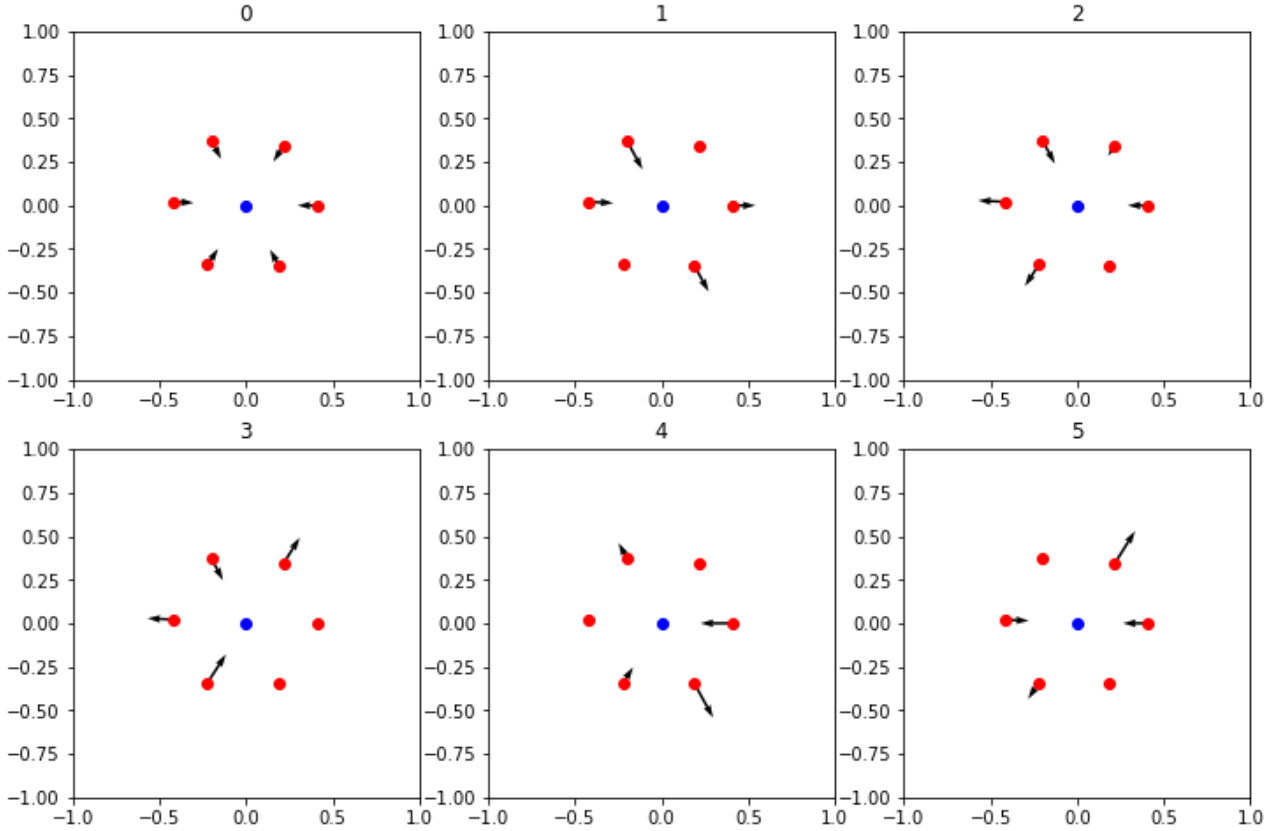
5% ReS_2 | 55% ReS_2 | 78% ReS_2 | 95% ReS_2

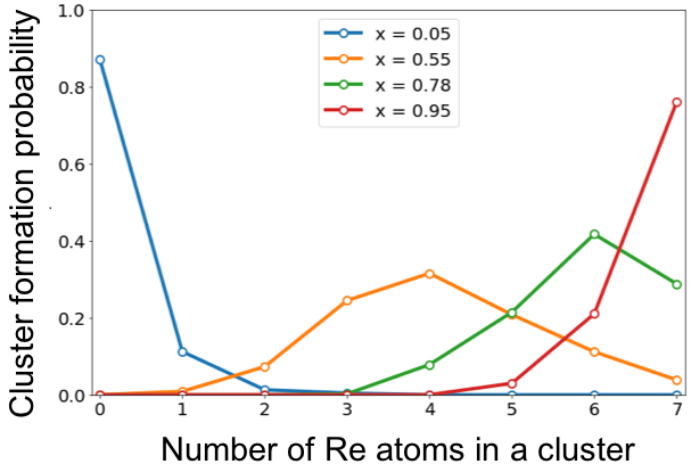


Data by Shize Yang and Matt Chisholm

Local crystallography



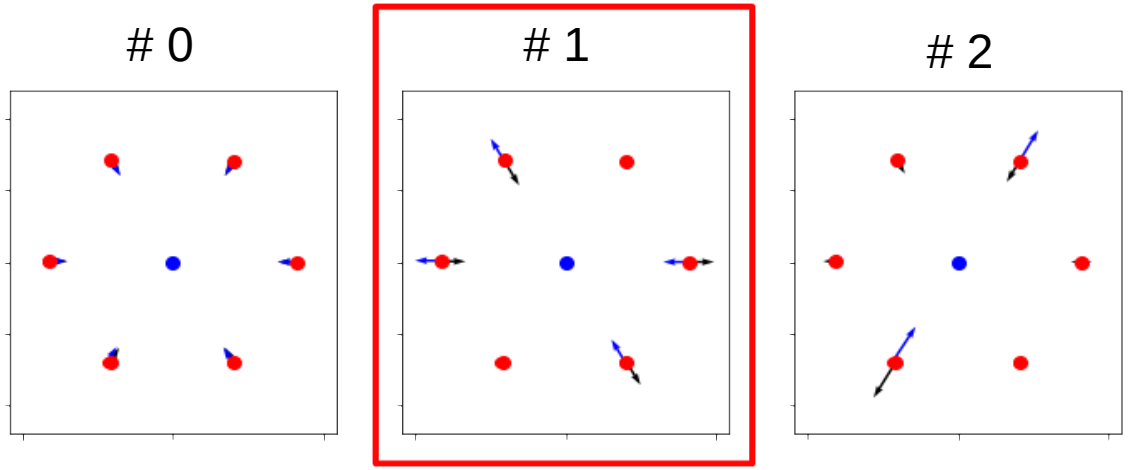




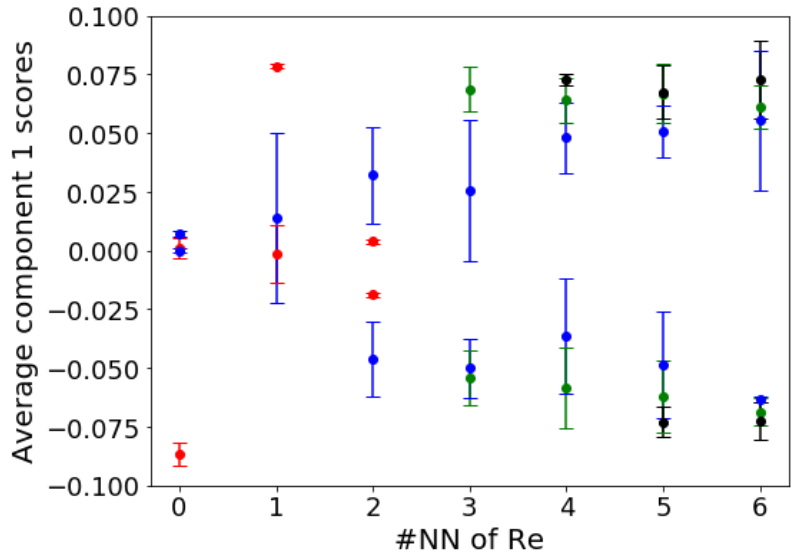
- Traditionally, the order parameter is defined based on symmetry and atomistic representation is established in the *ad hoc* manner
- But what if we define order parameter from the bottom up – based on the statistics of atomic distortions?
- And further correlate it to local chemical composition?

Describing the phase transition

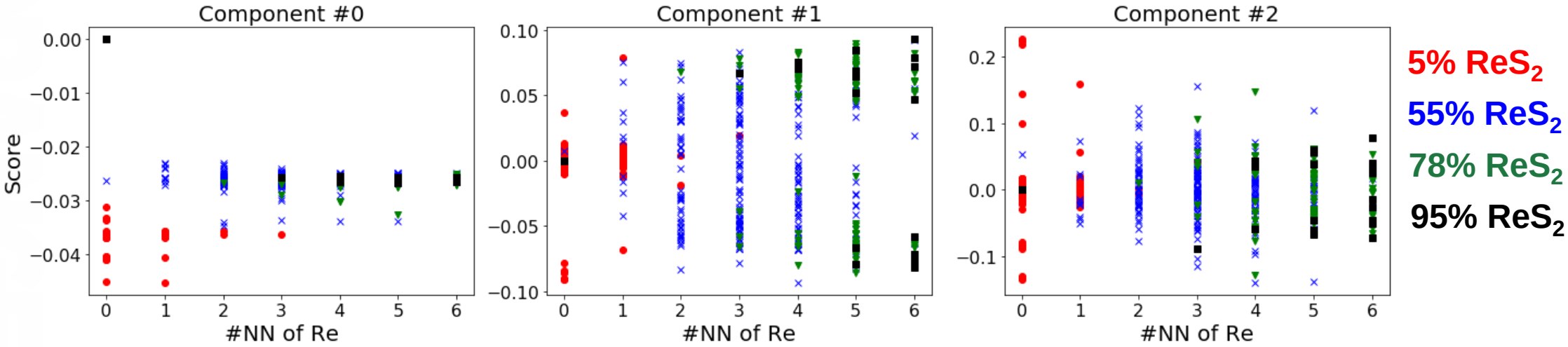
Three dominant distortion modes



Local symmetry breaking!

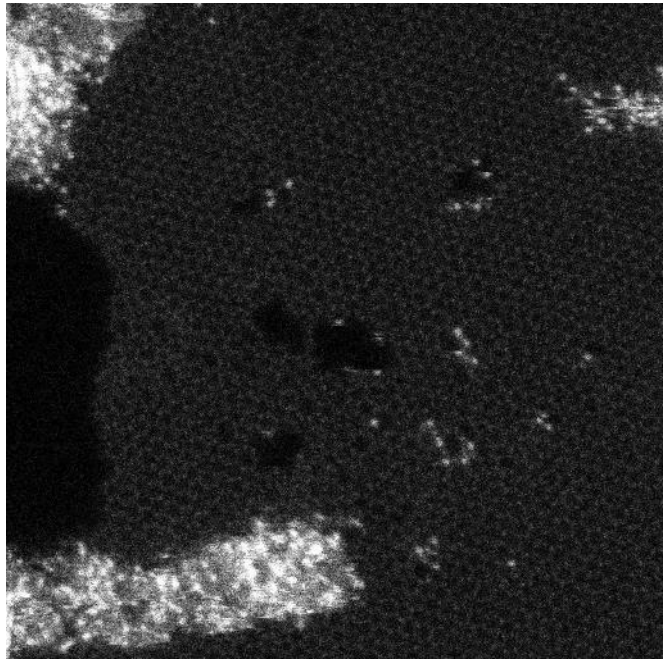


Mode distributions vs. global and local composition



What about chemical dynamics?

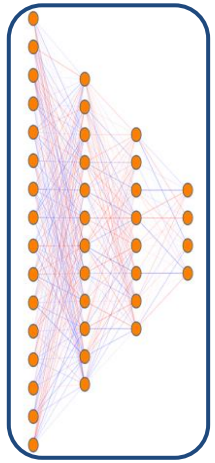
Experimental STEM movie
Graphene+Si under e-beam



Data collected by O. Dyck (ORNL)

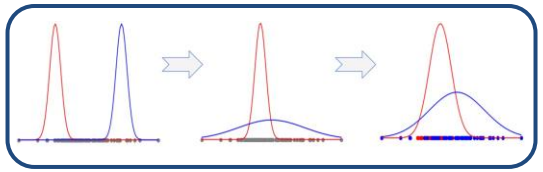
Deep & machine learning

Deep learning

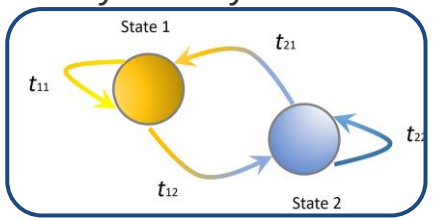


Position and type
of all atomic species

Clustering/unmixing:
Library of structural descriptors



Markov model:
System dynamics



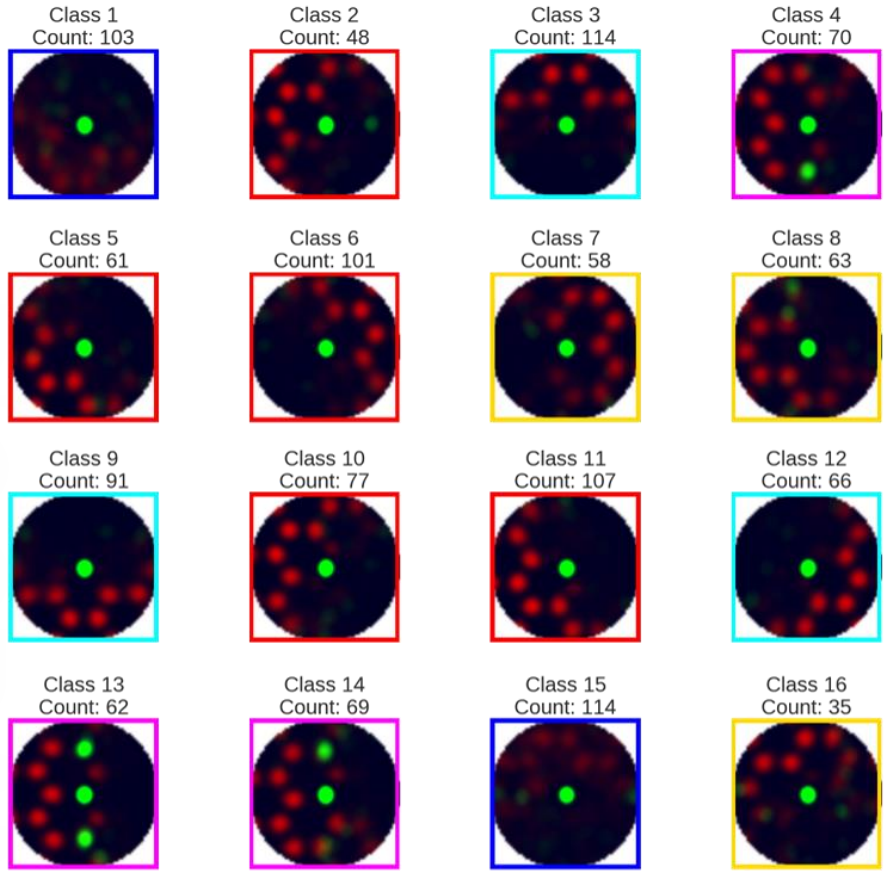
1. Convert noisy experimental data into atomic positions/trajectories → Deep convolutional neural networks
2. Create libraries of structural descriptors → Clustering/unmixing applied to the output of neural networks
3. Analysis of dynamics and transition probabilities → Markov modelling on the constructed classes

Ziatdinov et al., ACS nano 11, 12742 (2017)
Ziatdinov et al., Appl. Phys. Lett. 115, 052902 (2019)
Ziatdinov et al., npj Computational Materials 3, 31 (2017)
Maksov et al., npj Computational Materials 5, 12 (2019)

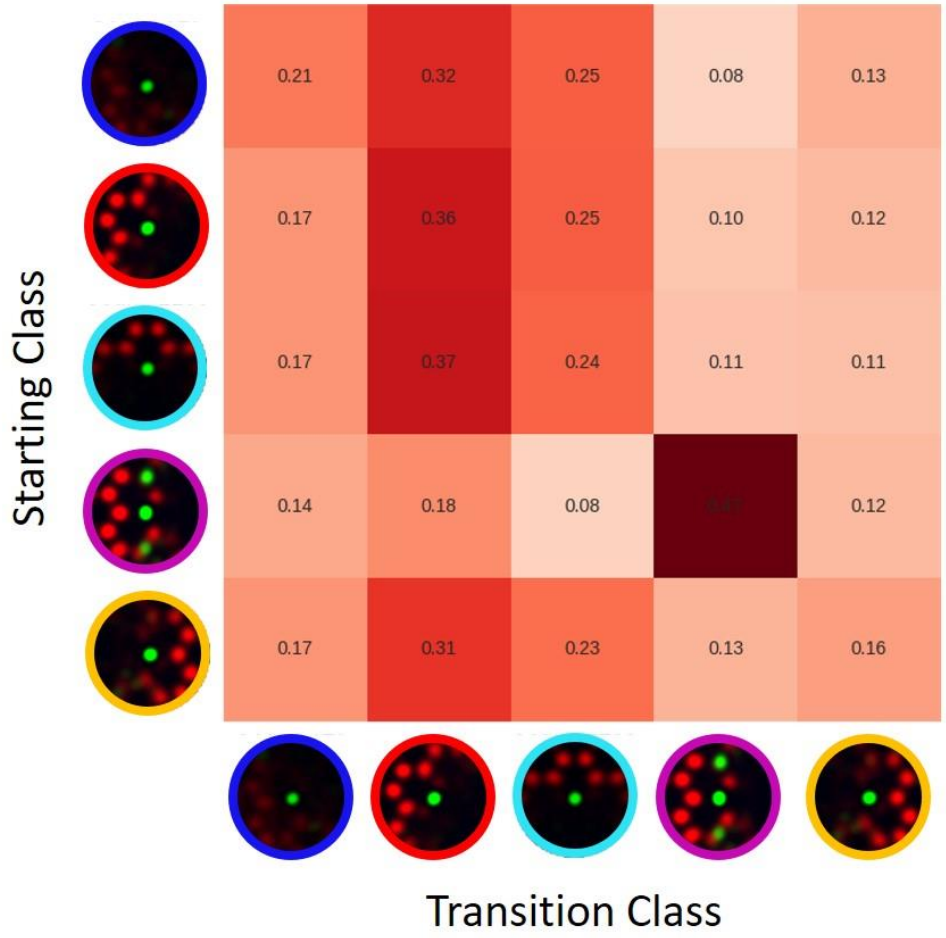
Local crystallography

THE UNIVERSITY of TENNESSEE KNOXVILLE

Derived classes of Si-C edge configurations



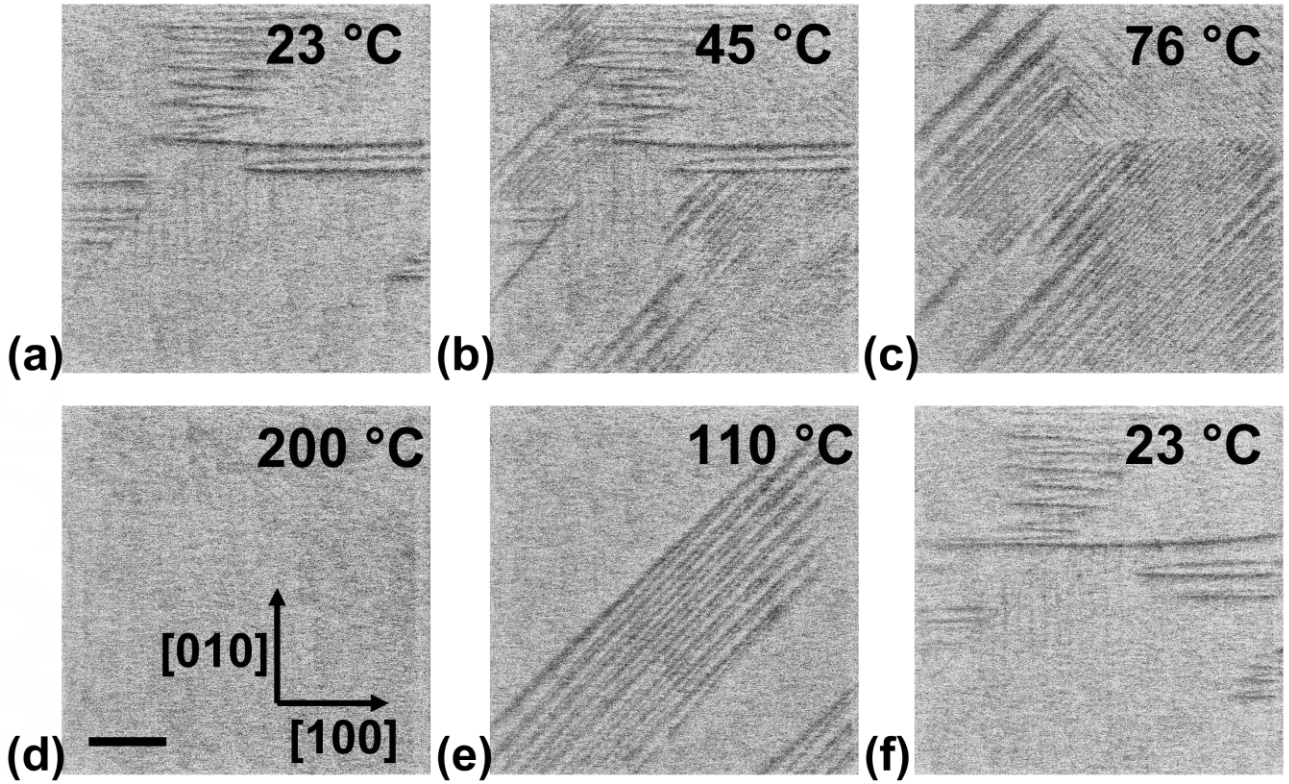
Transition probabilities matrix



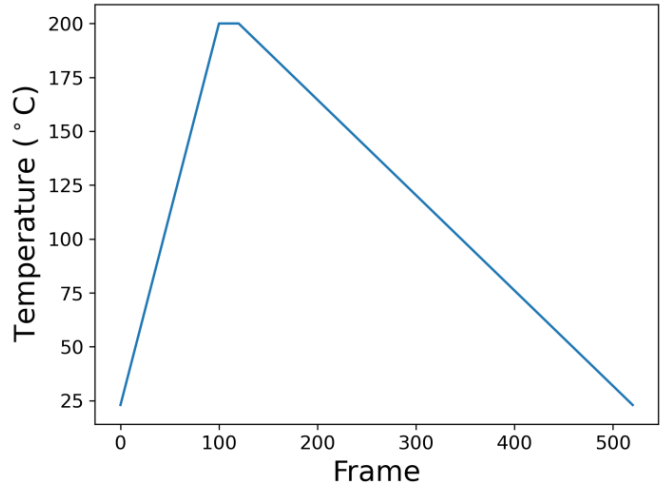
- Gaussian mixture model
- Discrete rotation symmetry + structural similarity algorithm
- Markov state analysis

Ziatdinov *et al.*, ArXiv:1901.09322 (2019)

Mesoscopic STEM data



Scale: 500 nm

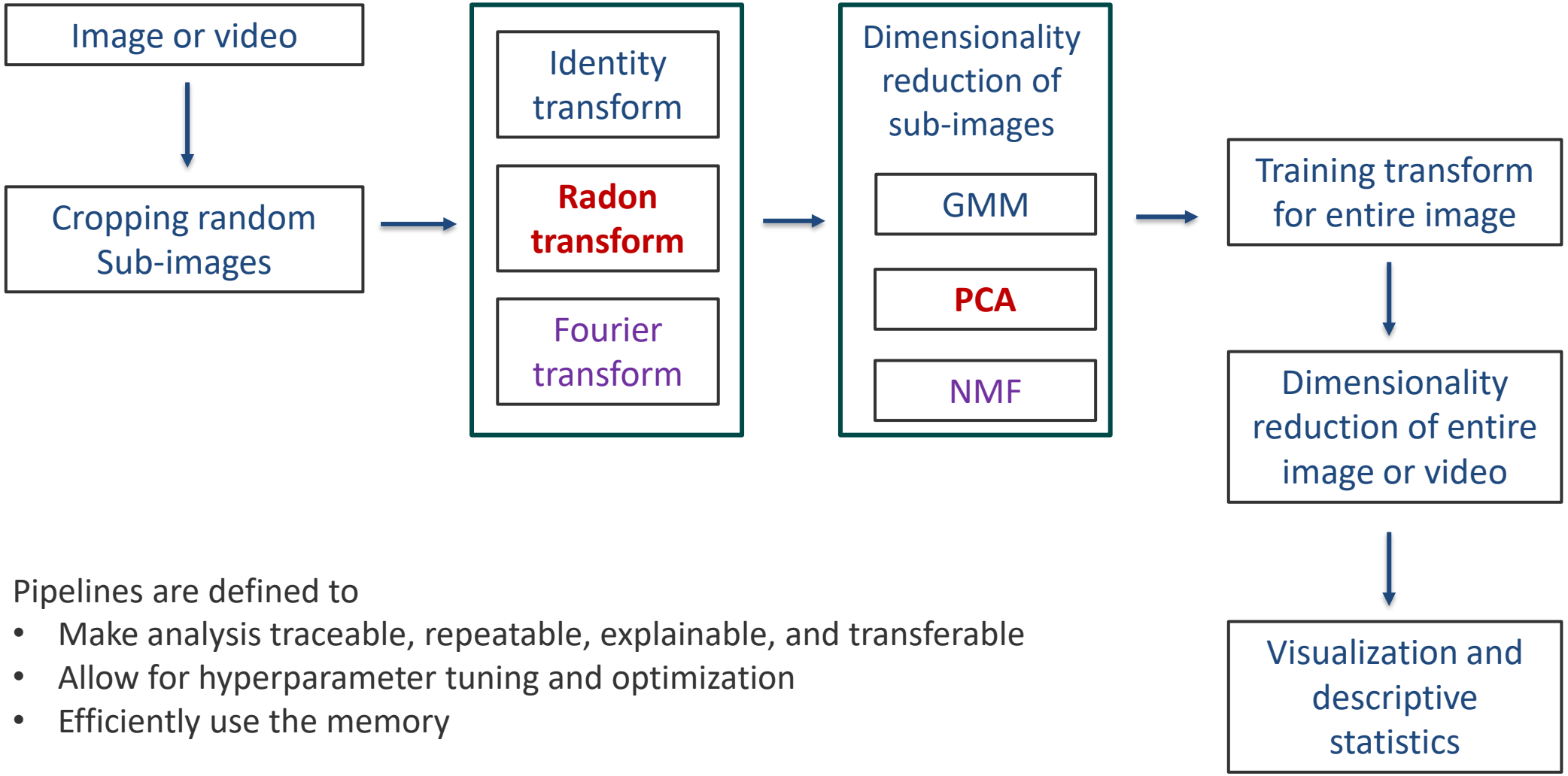


Key Observations

- We start with only 180° domain walls at RT (tetragonal phase)
- 90° domain walls at around 50°C (orthorhombic phase)
- Curie Temperature (T_c = 137°C) (Cubic phase)
- 90° - 180° transformation at around 30°C while cooling

Dataset: Dr. Vasiliki Tileli, EPFL

Extended analysis pipeline



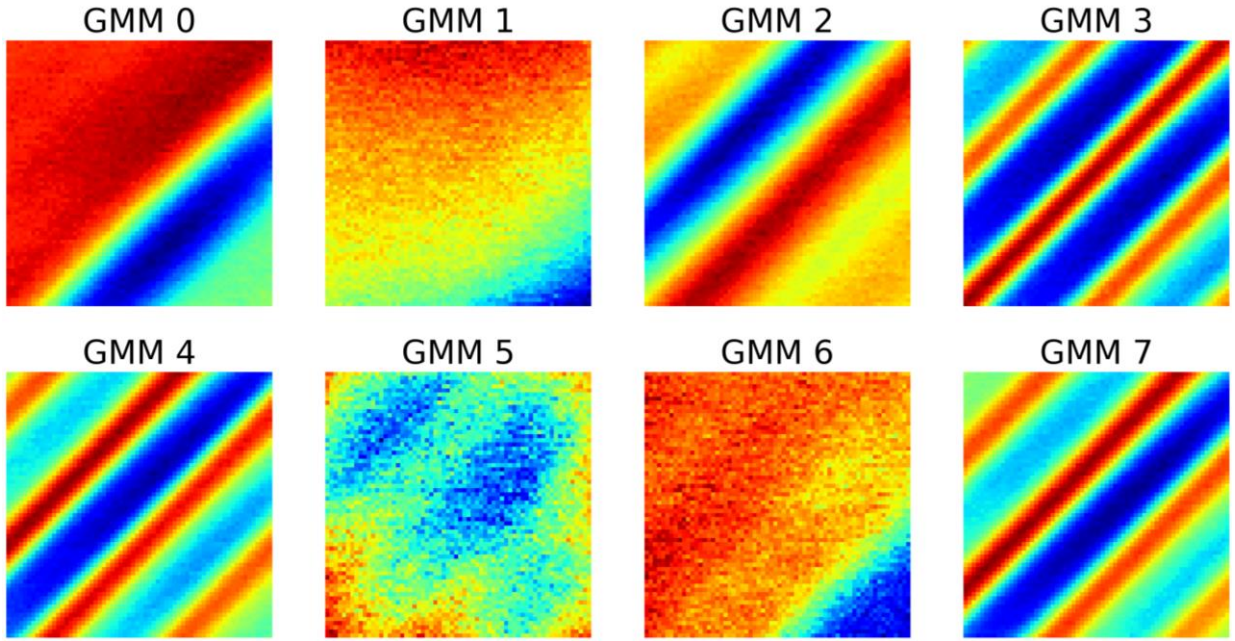
Pipelines are defined to

- Make analysis traceable, repeatable, explainable, and transferable
- Allow for hyperparameter tuning and optimization
- Efficiently use the memory

Gaussian Mixture Model

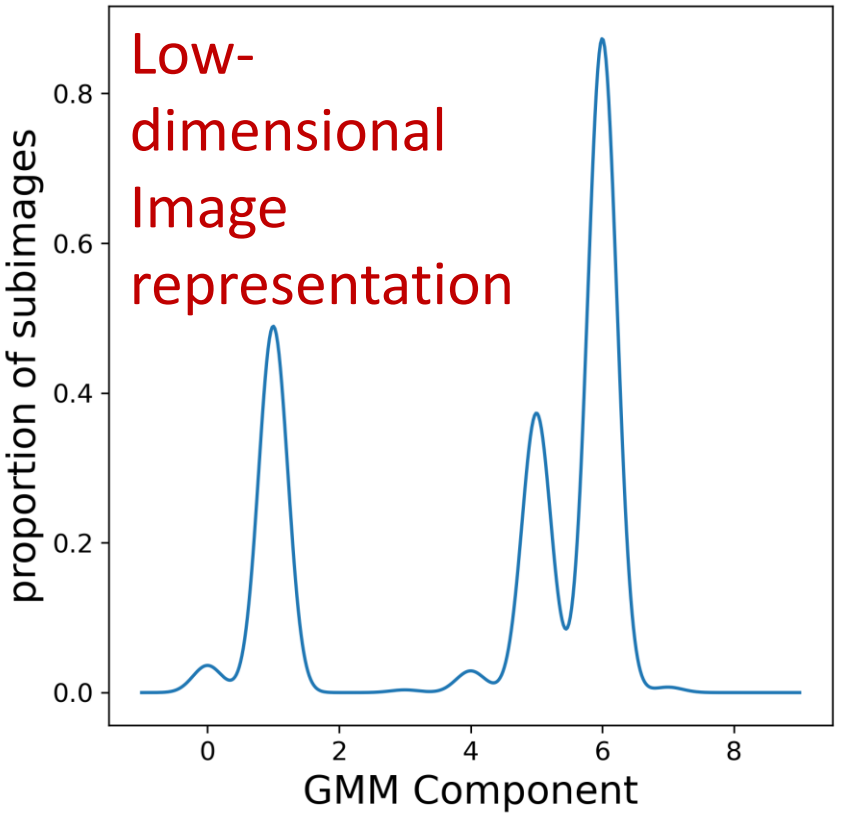
Workflow of the Gaussian Mixture Model (GMM) analysis

- Feature data: Create random sub-images (500 per frame) from each dataset of a given window size (64*64)
- Perform the GMM on the flattened feature data.
- And the results class centers of resulting classes:
- 90° domain are prominent, while increasing the number of classes will have classes associated with 180° domain walls
- Notice the shifts in the domain wall positions in similar looking classes

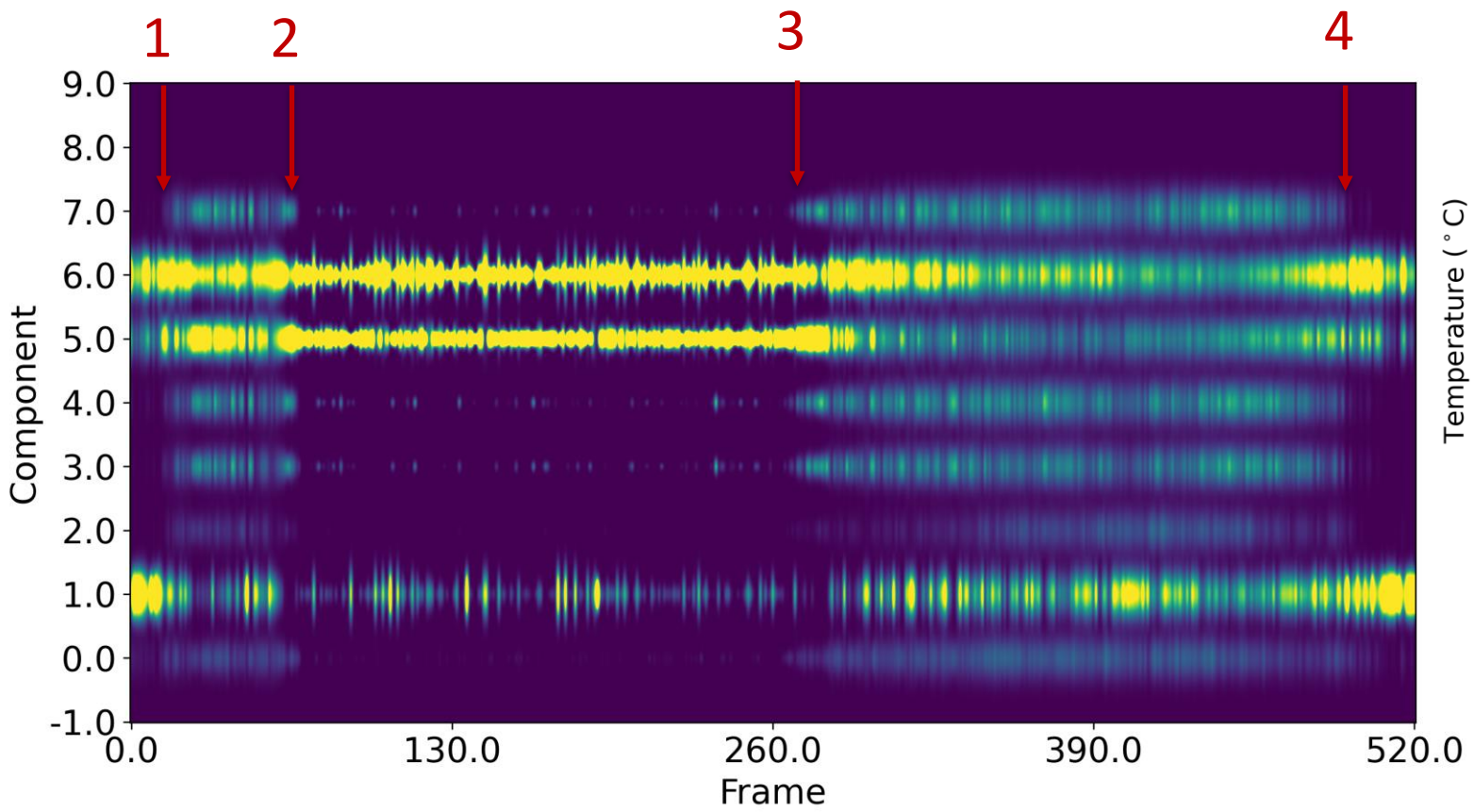


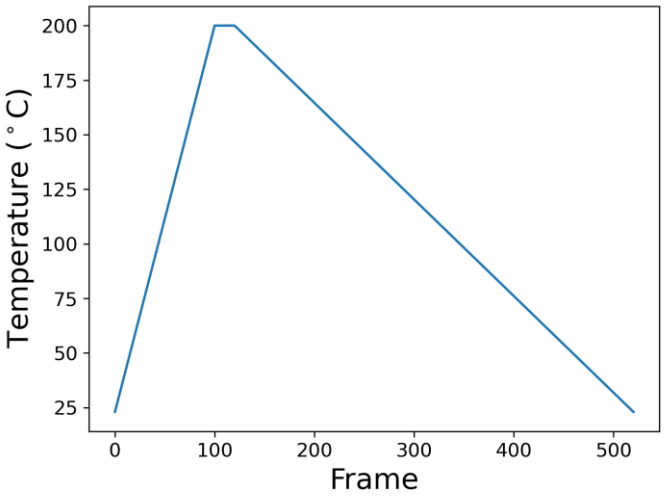
… Continued

- **Workflow of the Gaussian Mixture Model (GMM) analysis**
  - Now each image in our dataset can be represented as a histogram of 8 bins, where the height of each bin corresponds to number of sub-images that fall into the category.
  - Or a continuous probability distribution that approximates the above-mentioned histogram.
  - We chose the latter, and an example of this representation:
  - We will call this the which is a low-dimensional representation “feature-vector” of an image.



Et voila!

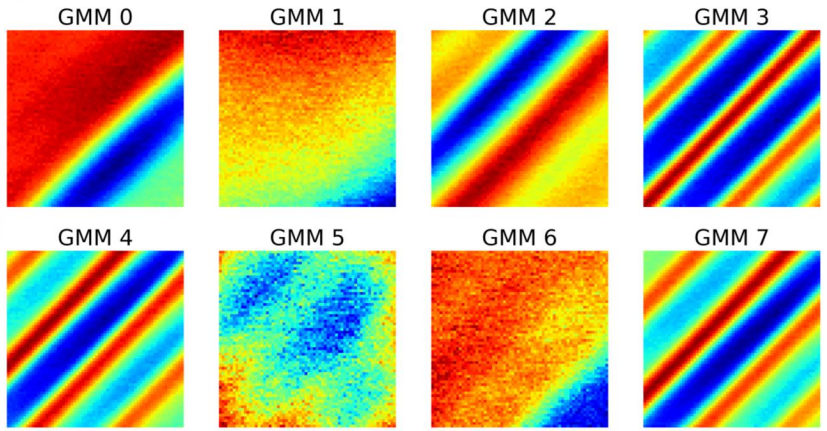


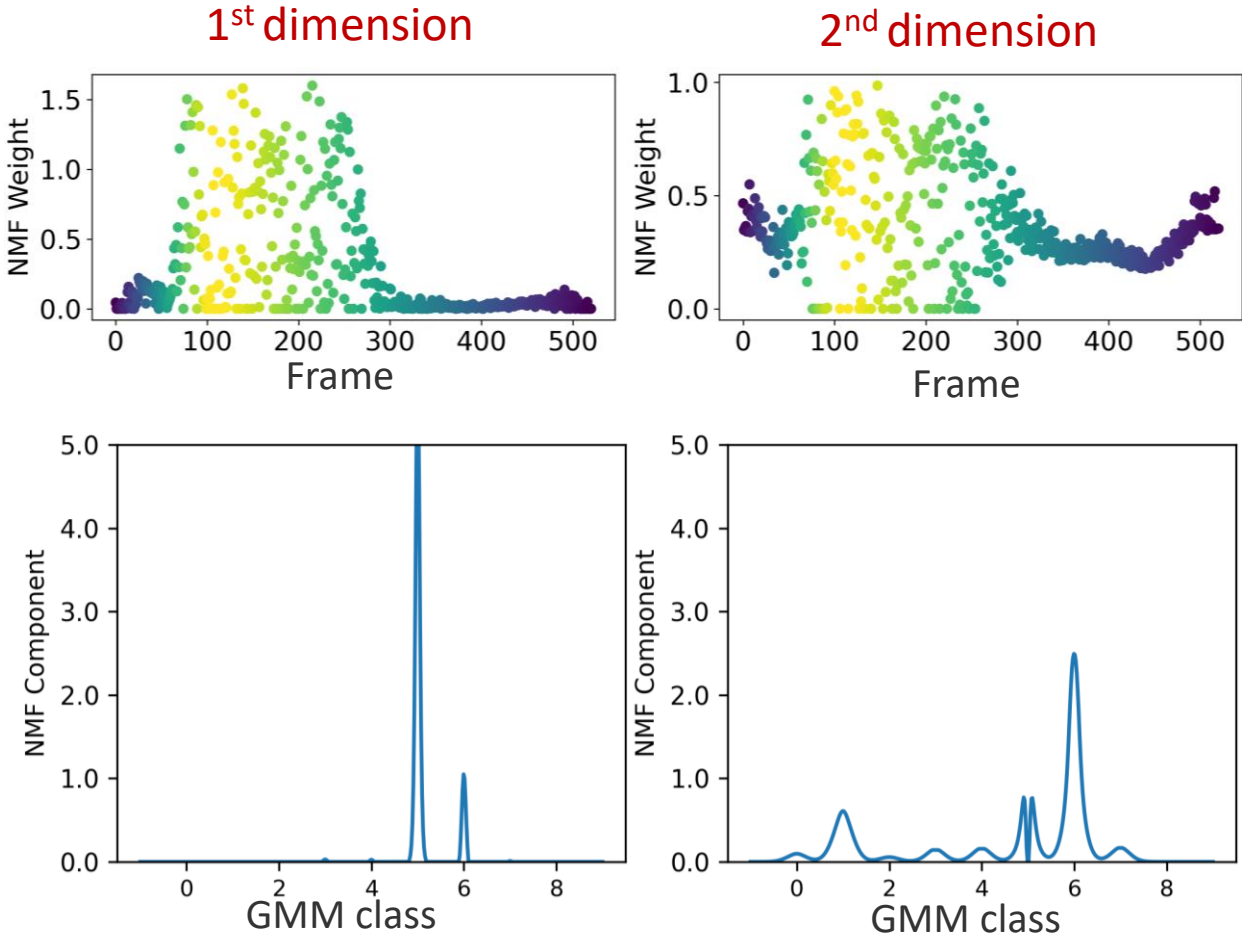


1) 180° domain walls -90°domain walls transition
2) Ferro to Paraelectric transition (Curie temperature)
3) Appearance of 90° domain walls
4) 90° domain - 180°domain walls transition

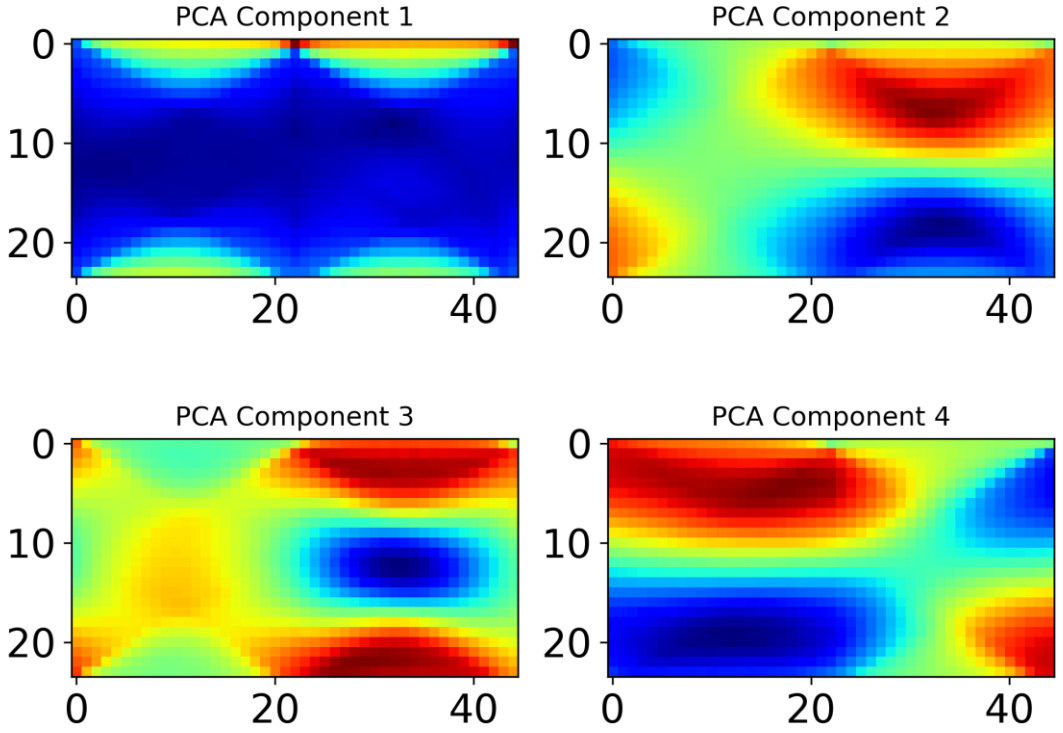
Describing phase transition

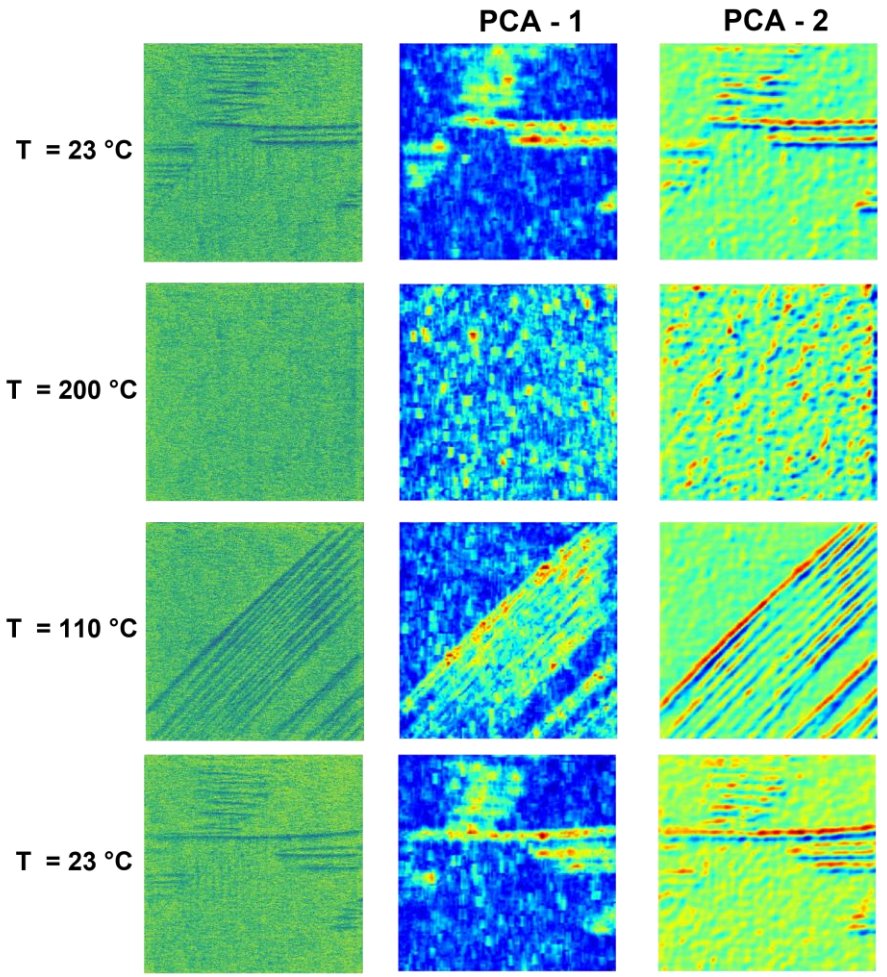
- The data representation can be furthe compressed using dimensionality reduction techniques on the feature vectors
- Applying NMF on the feature vectors matrix:



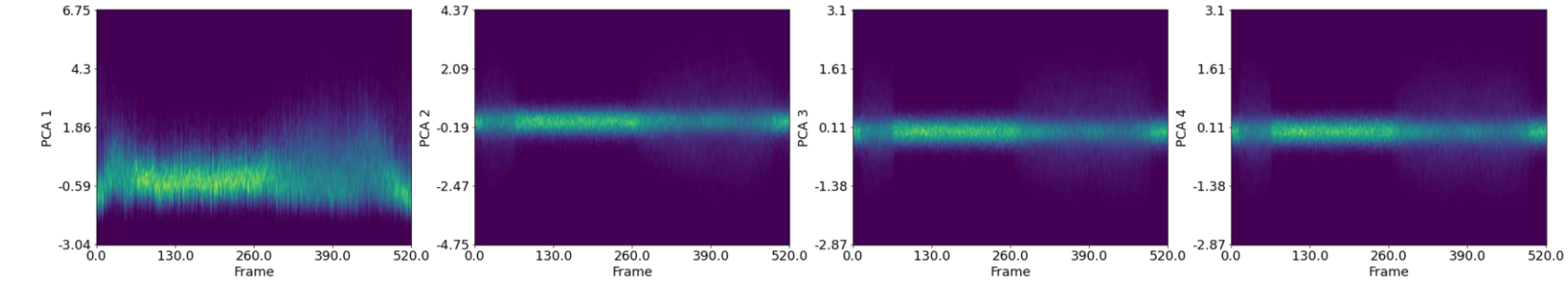


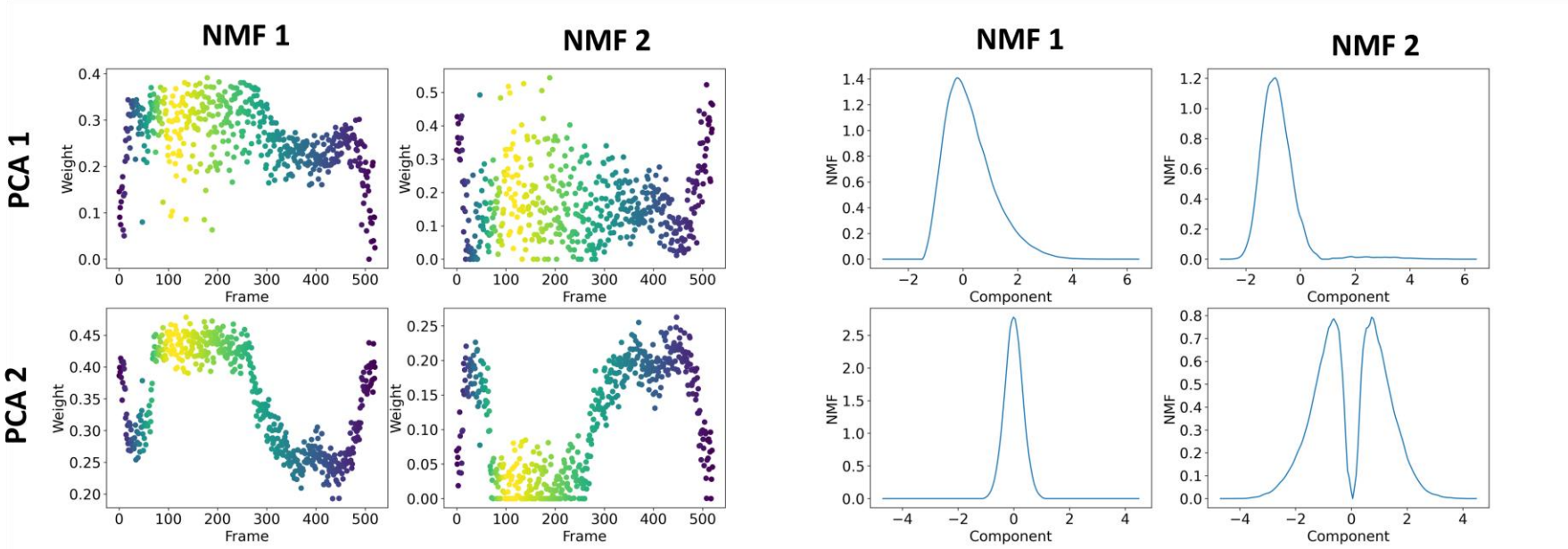
Radon Transform - PCA





Radon Transform - PCA

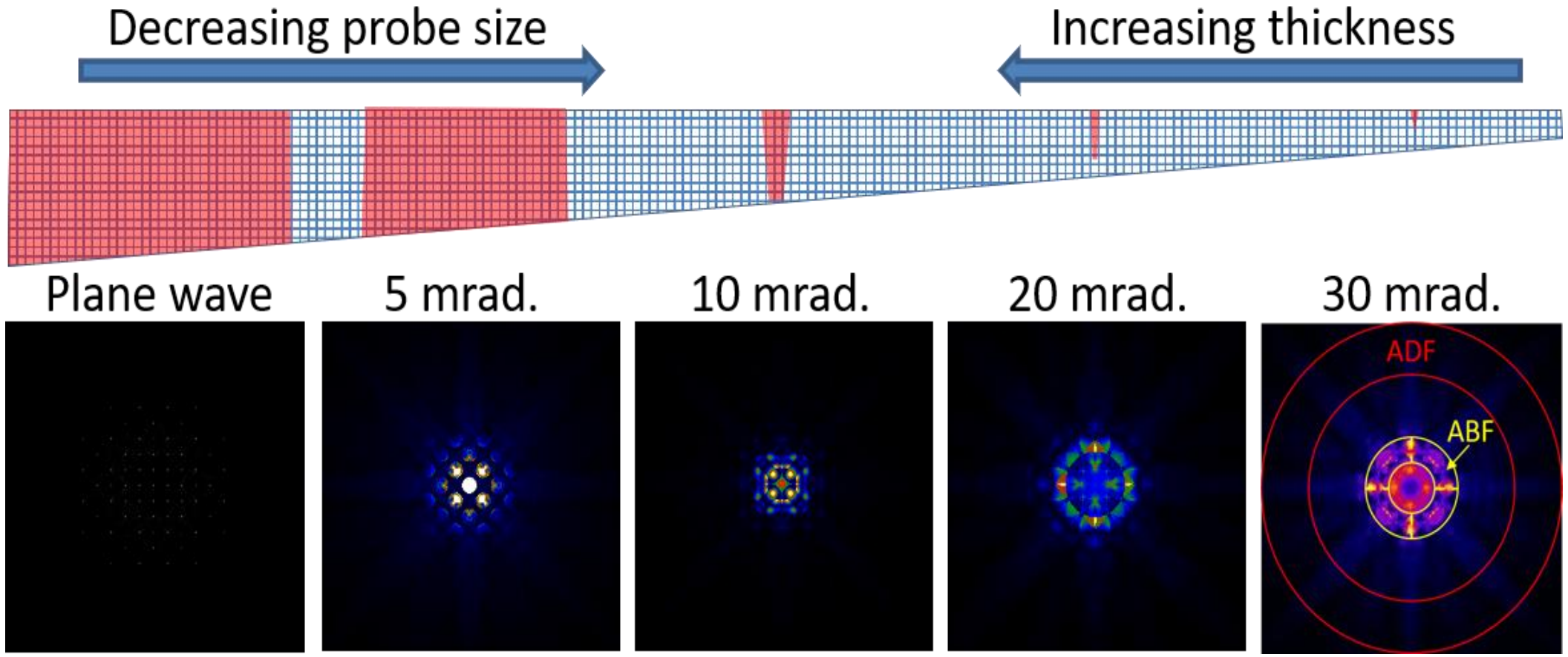




- We have mined the dataset at hand to produce low-dimensional representations for the images.
- The trends in this low-dimensional representations allowed us to study the physics of the system, in this case: phase transitions.

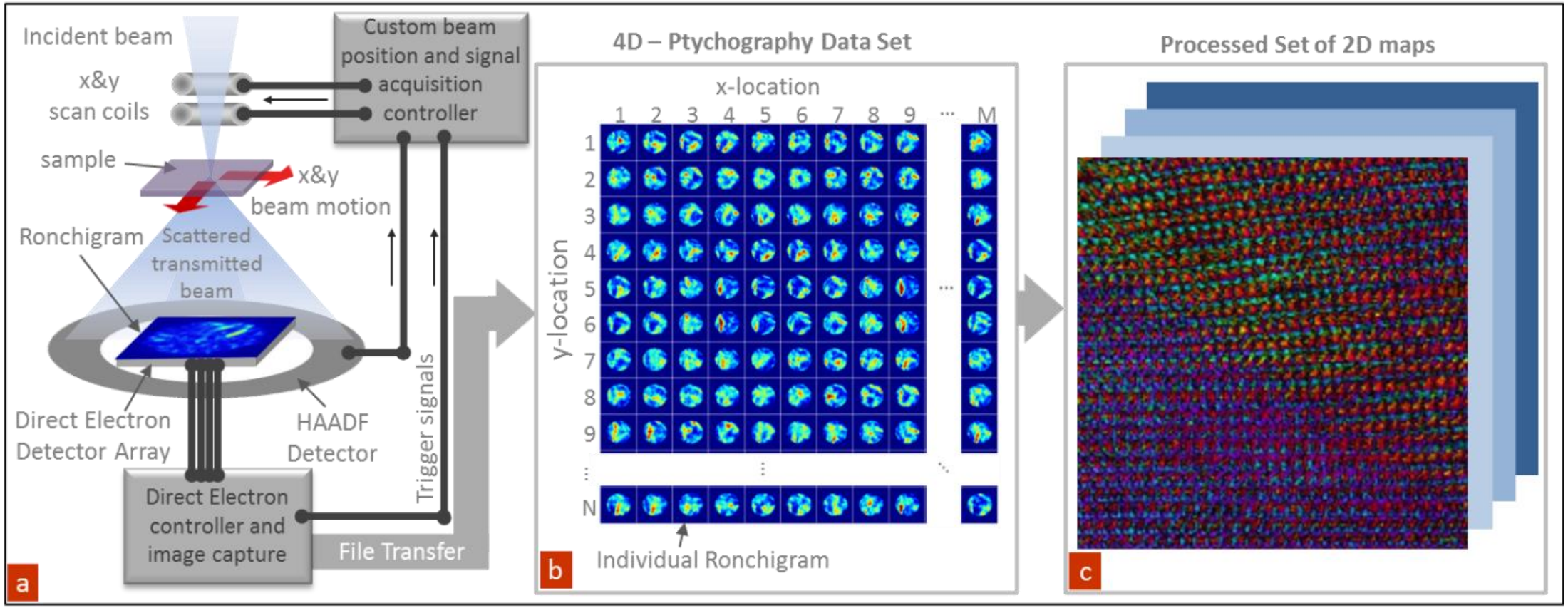
Valleti, S. M. P.; Ignatans, R.; Kalinin, S. V.; Tileli, V., Decoding the Mechanisms of Phase Transitions from In Situ Microscopy Observations. *Small* ***2022, 18 (40), 2104318.***

PCA on 4D STEM



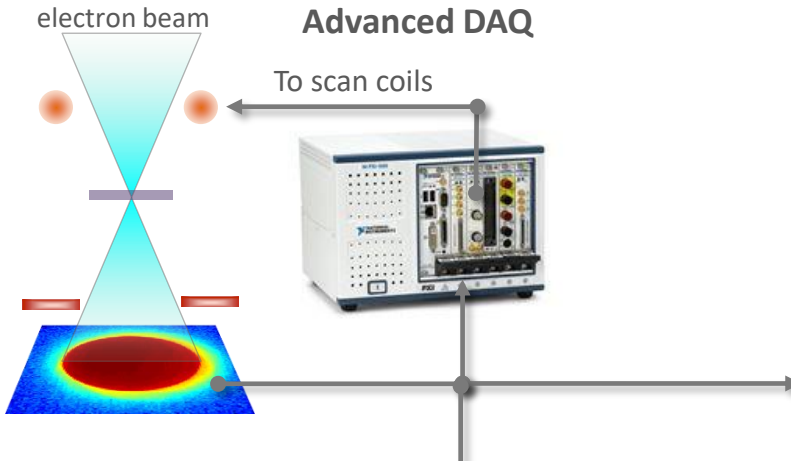
A.R. Lupini, M,P. Oxley, S,V. Kalinin, Pushing the limits of electron ptychography, Science 362, 399 (2018)

4D STEM Data

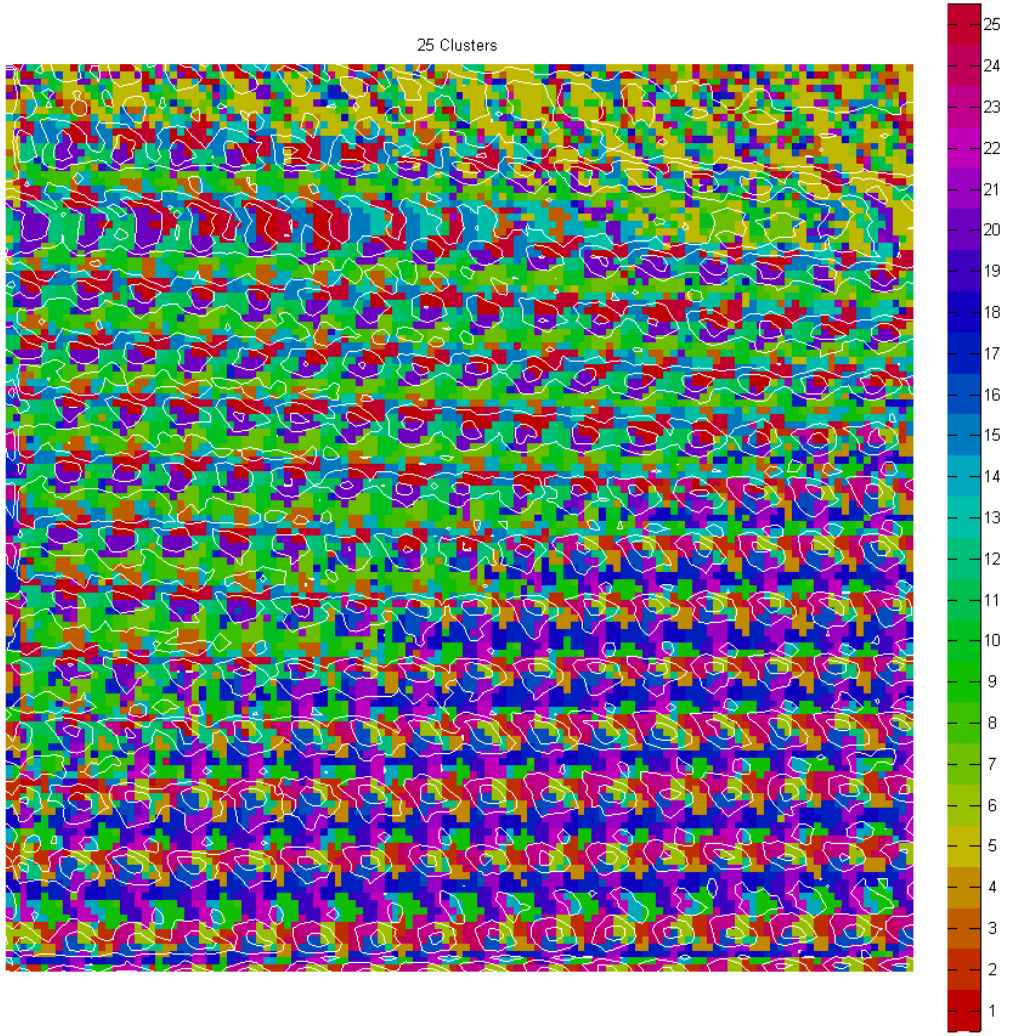


S. Jesse, M. Chi, A. Belianinov, C. Beekman, S.V. Kalinin, A.Y. Borisevich, and A.R. Lupini, *Big Data Analytics for Scanning Transmission Electron Microscopy Ptychography*, Sci. Rep. **6**, 26348 (2016).

Multivariate analysis of 4D STEM

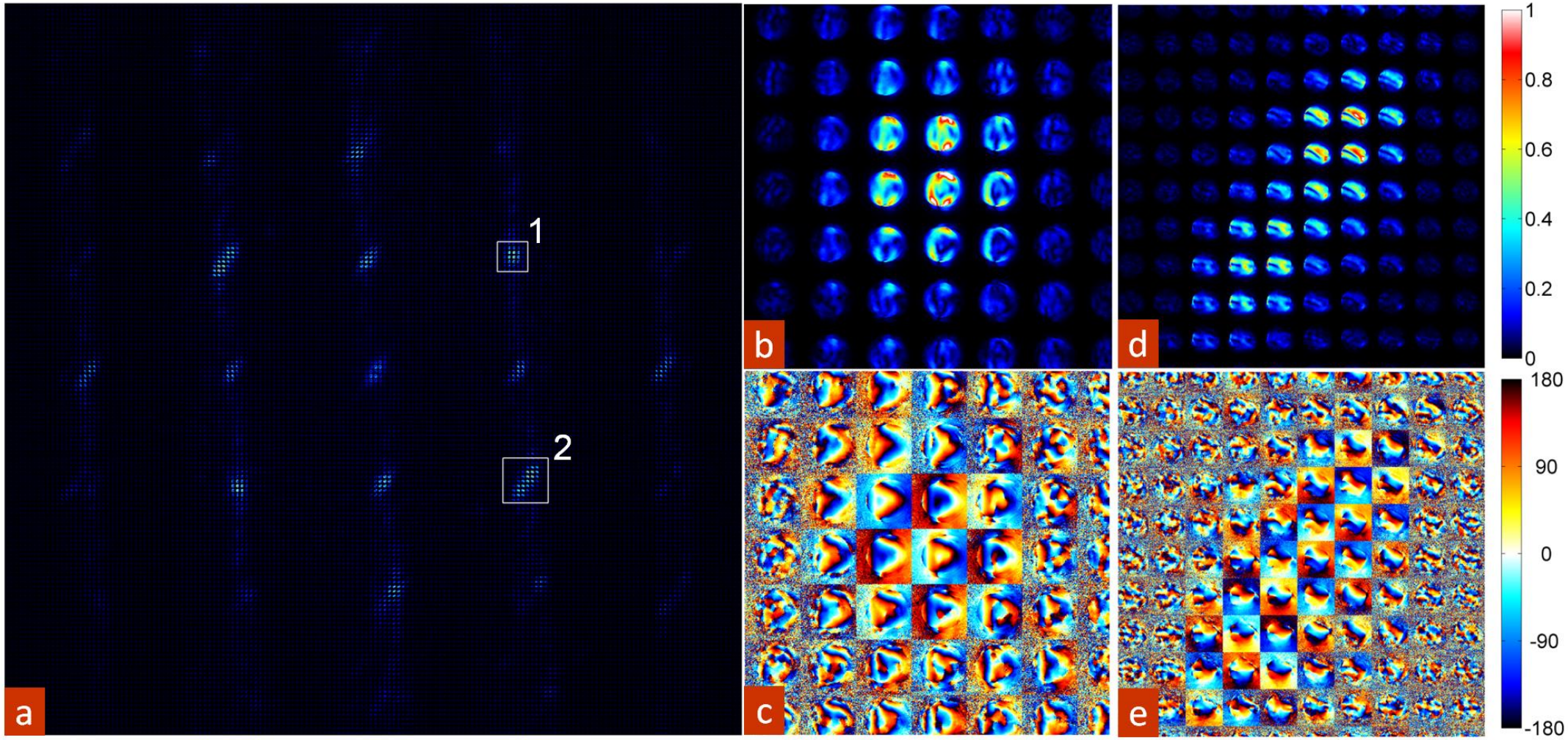


Bismuth Ferrite Domain Boundary



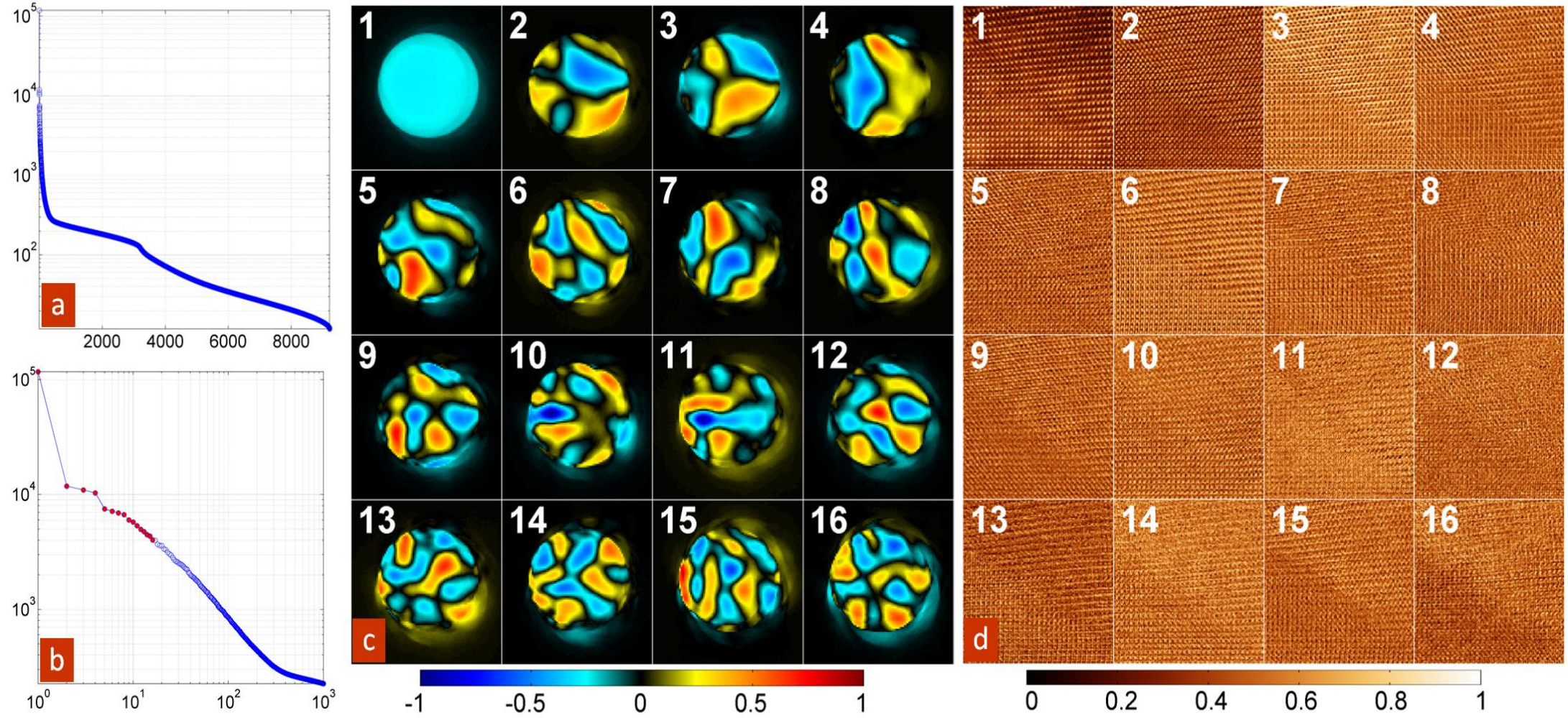
- Sub-Å imaging + Direct Electron Detection + Big data analysis enables effective STEM-ptychography.
- Next step: recover physical Information including strain, internal fields, orientation etc.

FFT of 4D STEM



S. JESSE, M. CHI, A. BELIANINOV, C. BEEKMAN, S.V. KALININ, A.Y. BORISEVICH, and A.R. LUPINI, *Big Data Analytics for Scanning Transmission Electron Microscopy Ptychography*, Sci. Rep. **6**, 26348 (2016).

PCA on 4D STEM Data

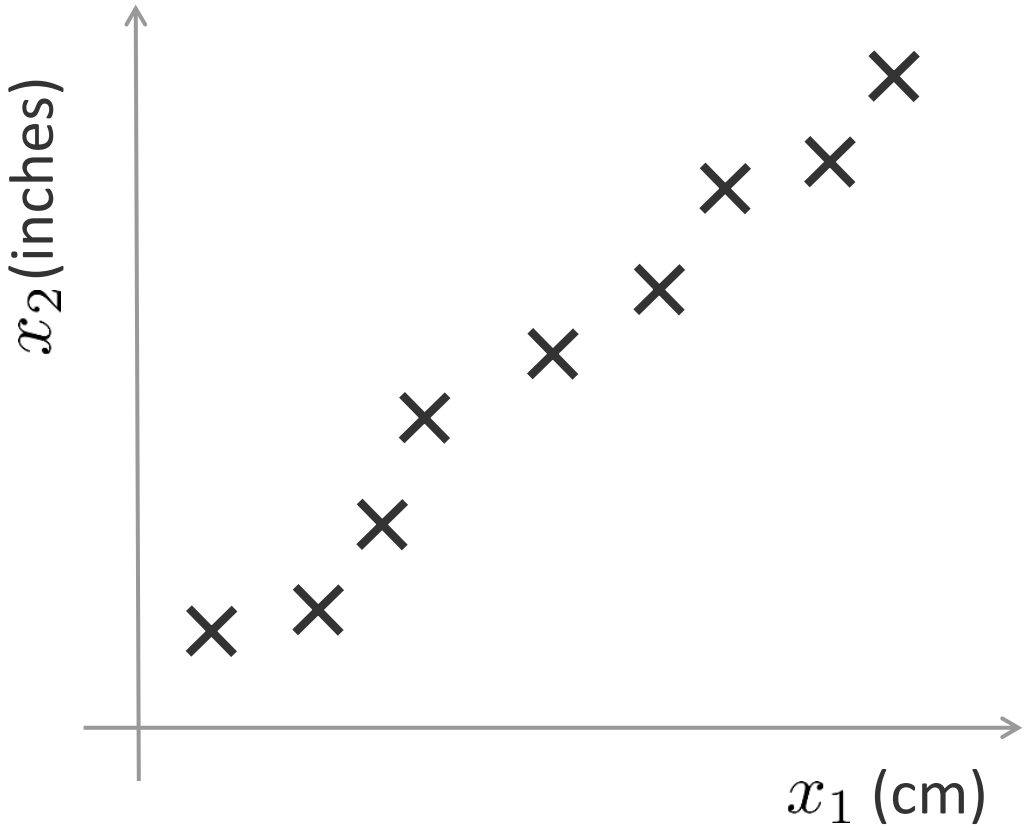


S. JESSE, M. CHI, A. BELIANINOV, C. BEEKMAN, S.V. KALININ, A.Y. BORISEVICH, and A.R. LUPINI, *Big Data Analytics for Scanning Transmission Electron Microscopy Ptychography*, Sci. Rep. **6**, 26348 (2016).

14_FerroicBlocks_mockup_paper_v3a.ipynb

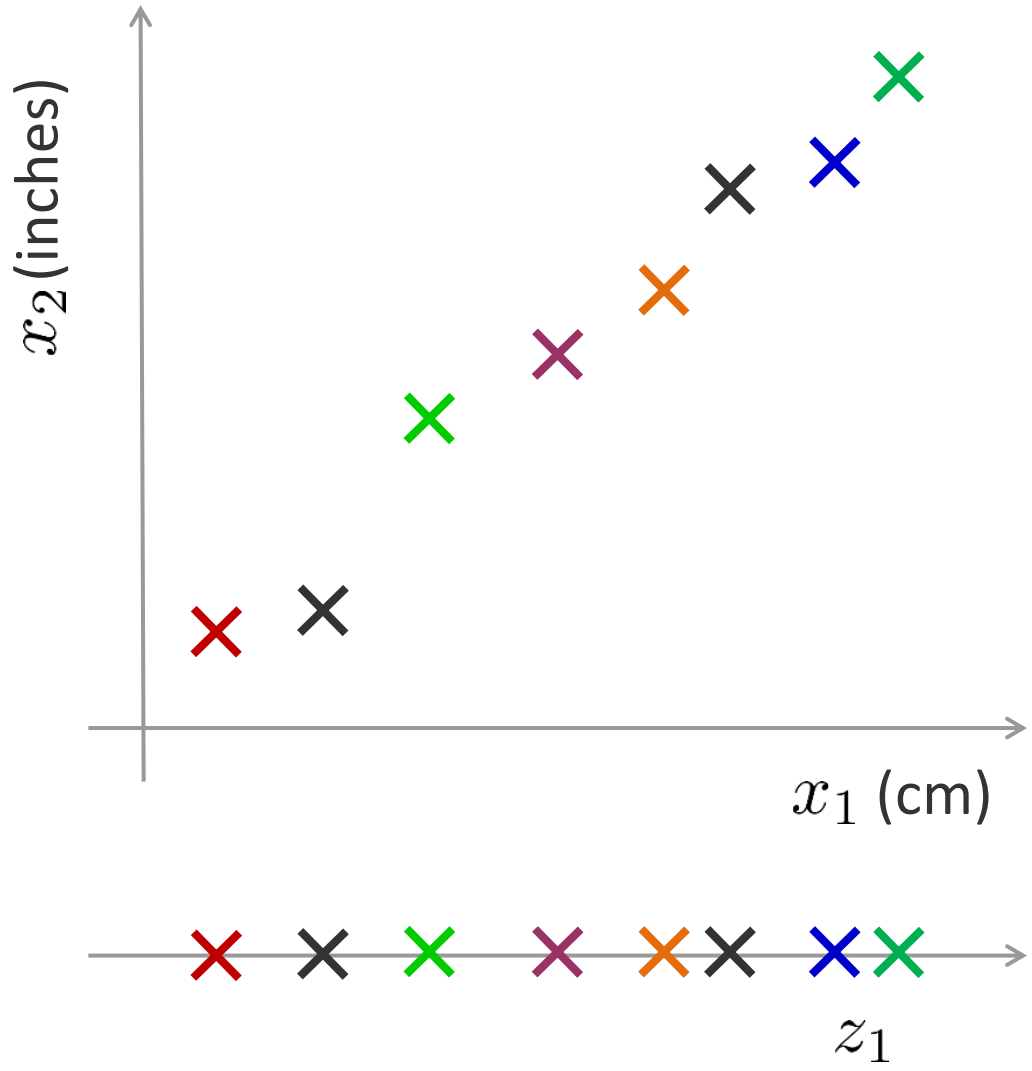
Manifold Hypothesis!

Simple Example



Reduce data from 2D to 1D

Simple Example



Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

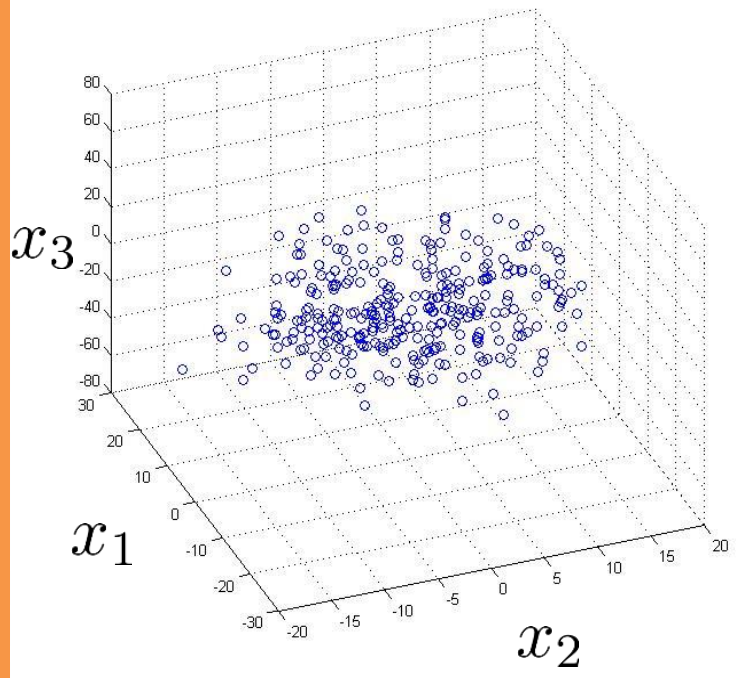
$$x^{(2)} \rightarrow z^{(2)}$$

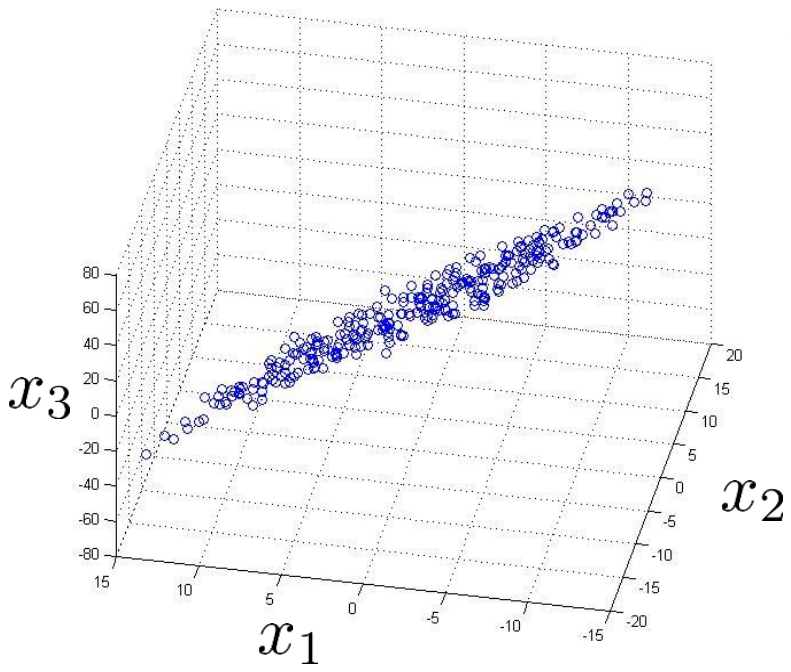
$$\vdots$$

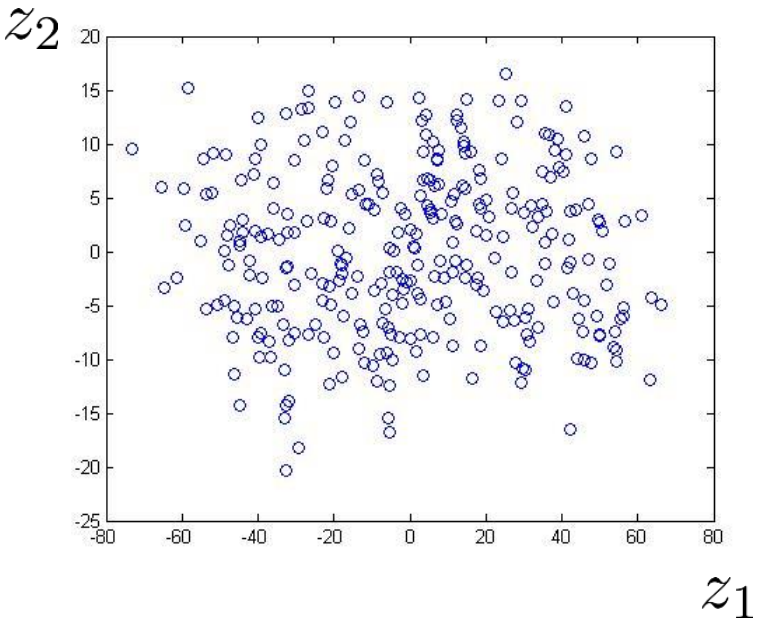
$$x^{(m)} \rightarrow z^{(m)}$$

Another Simple Example

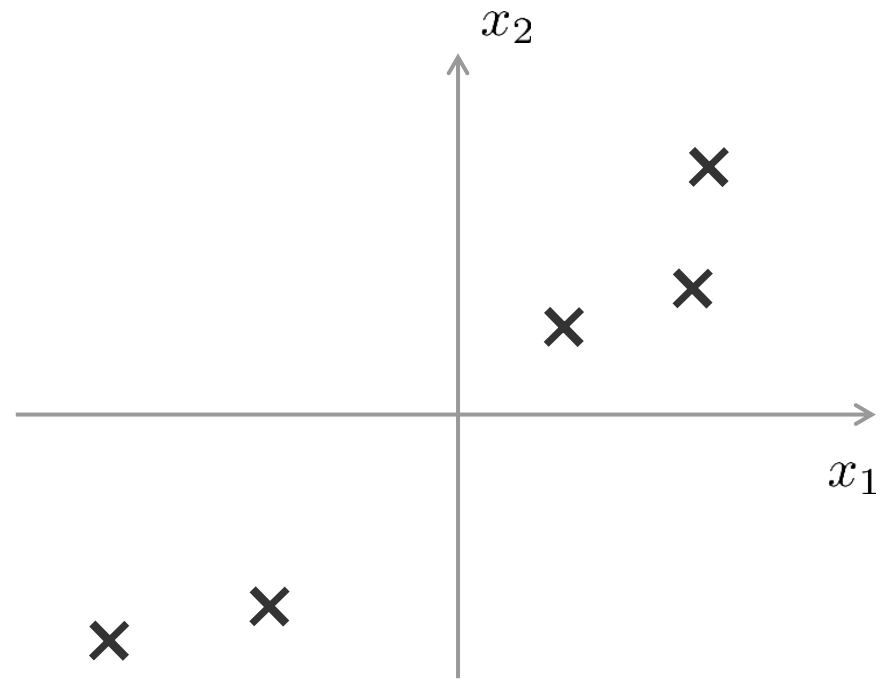
Reduce data from 3D to 2D



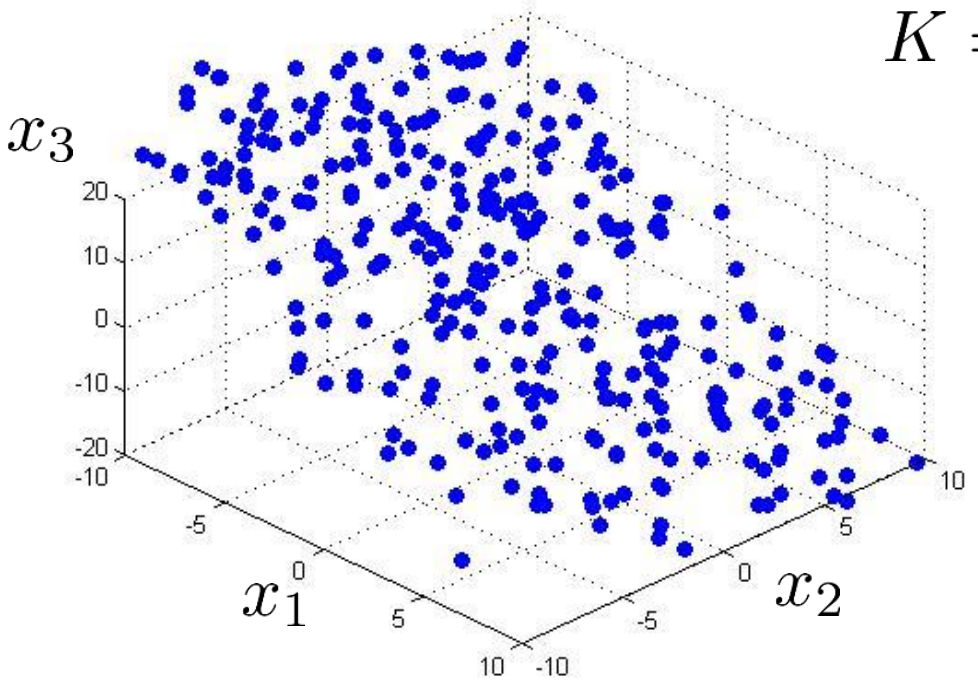




Generalize the problem



$3D \rightarrow 2D$
$K = 2$



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Variance and covariance

1D: Variance=(Standard deviation)^2

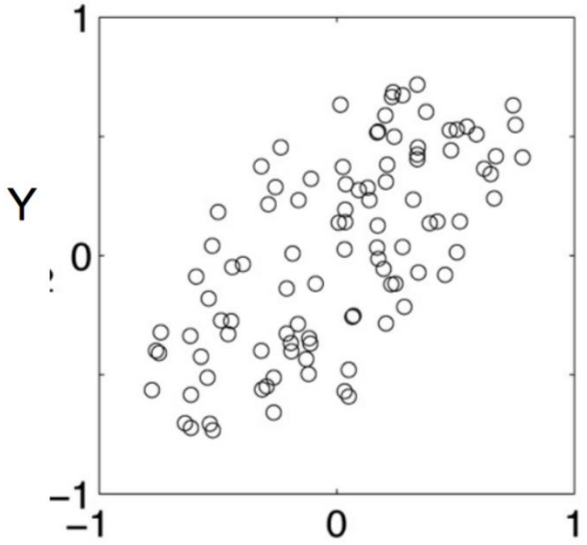
$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

2D: Covariance: measures the correlation between X and Y

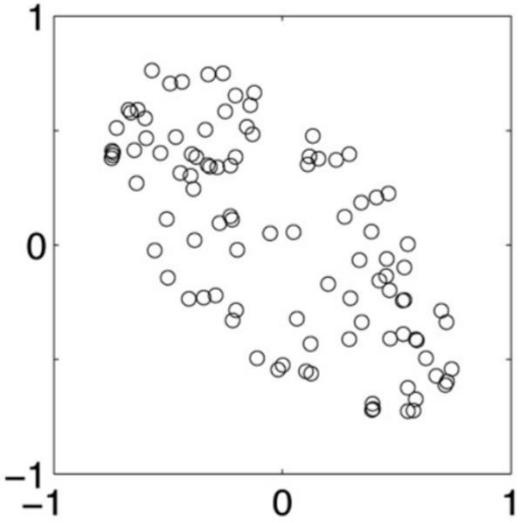
- Cov(X,Y)=0: independent
- Cov(X,Y)>0: move in the same direction
- Cov(X,Y)<0: move in opposite direction

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

positive covariance



negative covariance



Multidimensional data: covariance matrix

- Contains covariance values between all possible dimensions (=attributes):

$$C^{nxn} = (c_{ij} \mid c_{ij} = \mathrm{cov}(Dim_i, Dim_j))$$

- Example for three attributes (x,y,z):

$$C = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix}$$

- Eigenvalues of covariance matrix contain information on the independent factors of variability
- Eigenvectors of covariance matrix provide the information on direcitons

Principal Component Analysis

- Center the data (subtract the mean $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$ from each data point)
- Compute the $D \times D$ covariance matrix $\mathbf{S}$ using the centered data matrix $\mathbf{X}$ as

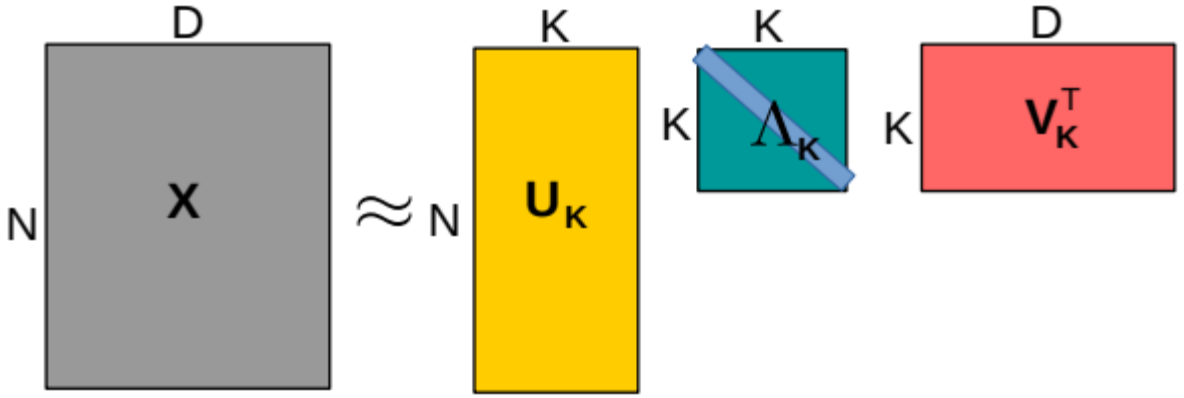
$$\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$$ (Assuming $\mathbf{X}$ is arranged as $N \times D$)

- Do an eigen decomposition of the covariance matrix $\mathbf{S}$ (many methods exist)
- Take top $K < D$ leading eigvectors $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K\}$ with eigen values $\{\lambda_1, \lambda_2, \ldots, \lambda_K\}$
- The K-dimensional projection/embedding of each input is $\boldsymbol{z}_n \approx \mathbf{W}_K^\top \boldsymbol{x}_n$
- Where $\mathbf{W}_\mathrm{K} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$ is projection matrix of size $D \times K$

Adapted from Introduction to Machine Learning by Nisheeth

Singular Value Decomposition

- If we just use the top $K < \min\{N, D\}$ singular values, we get a rank-K SVD



$$\mathbf{X} \approx \hat{\mathbf{X}} = \sum_{k=1}^{K} \lambda_k \boldsymbol{u}_k \boldsymbol{v}_k^\top = \mathbf{U}_K \Lambda_K \mathbf{V}_K^\top$$

reconstruction error $\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|$'

- Fact: SVD gives the best rank-K approximation of a matrix

- PCA is done by doing SVD on the covariance matrix **S** (left and right singular vectors are the sar and become eigenvectors, singular values become eigenvalues)

Adapted from Introduction to Machine Learning by Nisheeth

Principal Component Analysis

PCA: orthogonal transformation converting possibly correlated variables into linearly uncorrelated *principal components*

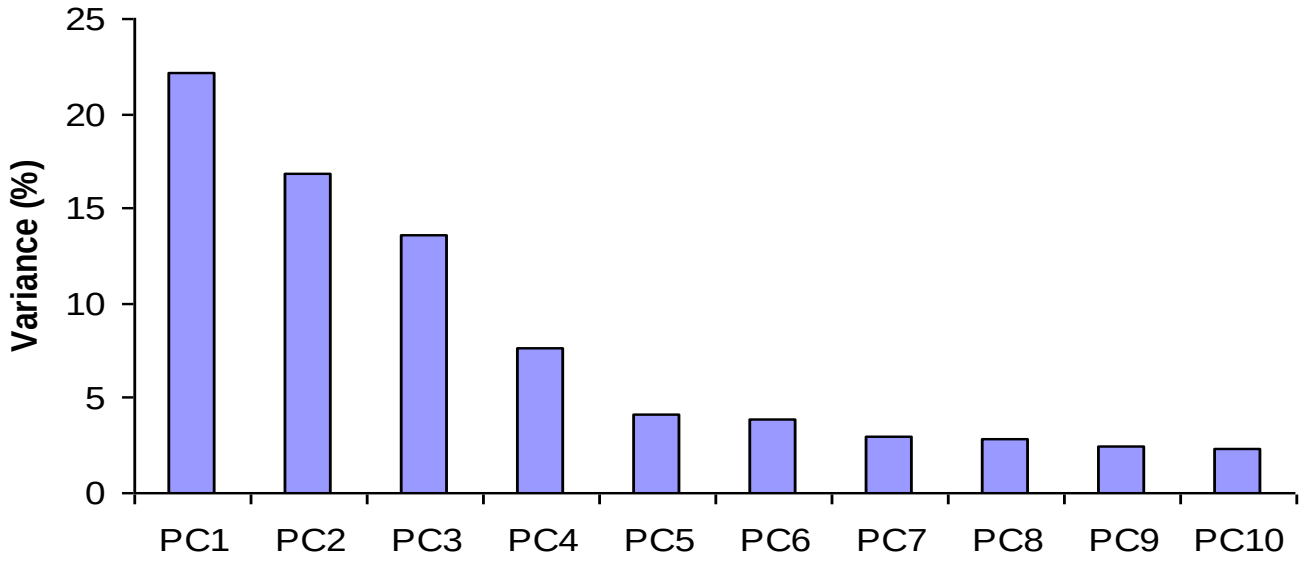
- PCA was invented by Karl Pearson in 1901, however the Singular Value Decomposition was independently derived some half a century earlier in Italy, Germany and France
- PCA transforms the data such that the greatest variance by any projection lies on the first coordinate
- Reveals internal structure of the data that best explains variance in the data set
- Since data often moves in clusters, PCA reveals those variables that drive the variance

Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points is Space". *Philosophical Magazine Series 6* **2** (11): 559–572

PCA: Eigenvalues

Eigenvalues λ_j are used for calculation of [% of total variance] (V_j) for each component j:

$$V_j = 100 \cdot \frac{\lambda_j}{\sum_{x=1}^{n} \lambda_x} \qquad \sum_{x=1}^{n} \lambda_x = n$$



PCA: Components

- The first PC retains the greatest amount of variation in the sample
- The k^{th} PC retains the k^{th} greatest fraction of the variation in the sample
- The k^{th} largest eigenvalue of the correlation matrix C is the variance in the sample along the k^{th} PC
- The least-squares view: PCs are a series of linear least squares fits to a sample, each orthogonal to all previous ones

PCA Components and Loadings

- Technique useful for compression and classification of data
- Find new descriptors smaller than original variables
- Retain most of sample's information - correlation between original variables
- New descriptors are principal components (PCs)
- Loadings represent the “fraction” of PCs in initial data
- Uncorrelated, and ordered by fraction of total information retained in each PC

PCA in scikit-learn

sklearn.decomposition.PCA

class sklearn.decomposition.**PCA**(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None*) [source]

Methods

fit(X[, y])	Fit the model with X.
fit_transform(X[, y])	Fit the model with X and apply the dimensionality reduction on X.
get_covariance()	Compute data covariance with the generative model.
get_feature_names_out([input_features])	Get output feature names for transformation.
get_metadata_routing()	Get metadata routing of this object.
get_params([deep])	Get parameters for this estimator.
get_precision()	Compute data precision matrix with the generative model.
inverse_transform(X)	Transform data back to its original space.
score(X[, y])	Return the average log-likelihood of all samples.
score_samples(X)	Return the log-likelihood of each sample.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Apply dimensionality reduction to X.

Examples: Eigenfaces

- When viewed as vectors of pixel values, face images are extremely high dimensional. Image of 100x100 pixels has 10,000 dimensions.
- However, very few of 100x100 vectors are valid face images
- We want to effectively represent the subspace of face images

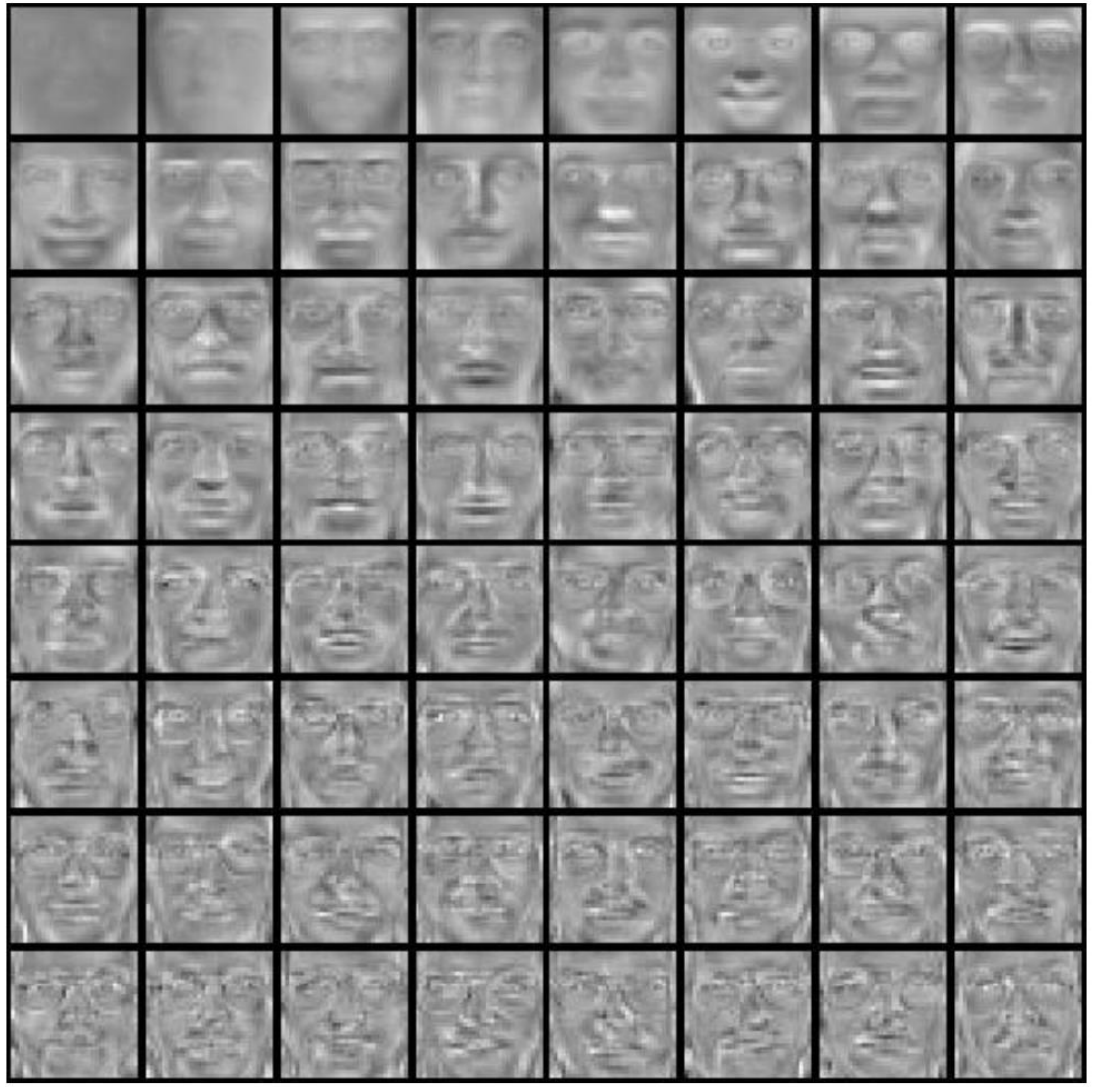
Adapted form Fereshteh Sadeghi

slide by Derek Hoiem

Examples: Eigenfaces

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$



Adapted from Fereshteh Sadeghi slide by Derek Hoiem

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$

$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

$$\hat{x} = \mu + w_1 u_1 + w_2 u_2 + w_3 u_3 + w_4 u_4 + \ldots$$

slide by Derek Hoiem

Adapted from Fereshteh Sadeghi

Reconstruction

P = 4

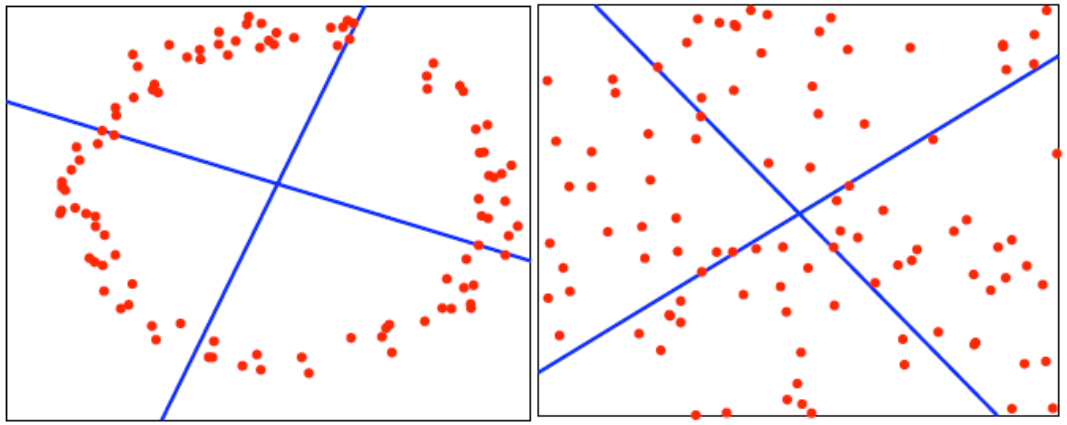
P = 200

P = 400

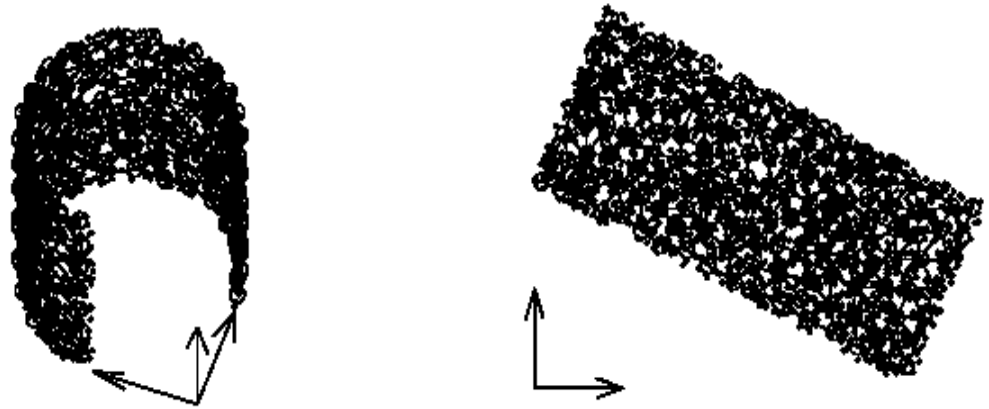
We can represent faces well by 400 components – rather then 10,000!

Adapted from Fereshteh Sadeghi

Limitations of PCA



PCA will make no difference between these examples



Non linear projection of a horseshoe

Non-linear PCA

- Suppose that instead of using the points $\mathbf{x}_i$ as is, we wanted to go to some different *feature space* $\phi(\mathbf{x}_i) \in \Re^N$
- E.g. using polar coordinates instead of cartesian coordinates would help us deal with the circle
- In the higher dimensional space, we can then do PCA
- The result will be non-linear in the original data space!
- Similar idea to support vector machines

Kernel PCA

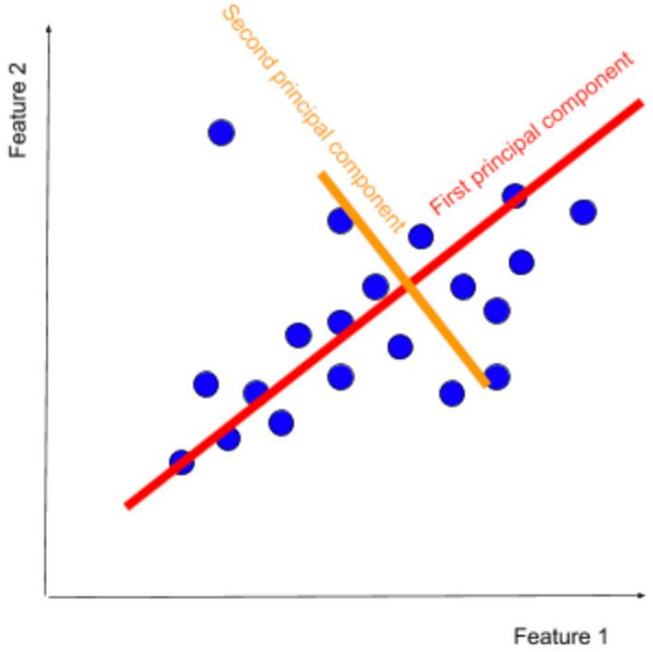
Kernel PCA is an unsupervised manifold learning technique that maps data points to a generally lower-dimensional space. It generalizes the Principal Components Analysis approach to non-linear transformations using the kernel trick (Schölkopf, Smola and Müller, 1996; Schölkopf, Smola and Müller, 1998; Schölkopf, Burges and Smola, 1999). The algorithm implicitly finds the leading eigenvectors and eigenvalues of the covariance of the projection $\phi(x)$ of the data in "feature space", where $\phi(x)$ is such that the kernel $K_n(x, y) = \phi(x) \cdot \phi(y)$ (i.e. K_n must not have negative eigenvalues). If the data is

(http://research.microsoft.com/users/Cambridge/nicolasl/papers/eigen_dimred.pdf)

Principal Component Analysis

PCA: orthogonal transformation converting possibly correlated variables into linearly uncorrelated *principal components*

- PCA transforms the data such that the greatest variance by any projection lies on the first coordinate
- The first PC retains the greatest amount of variation in the sample. The k^{th} PC retains the k^{th} greatest fraction of the variation in the sample.
- The k^{th} largest eigenvalue of the correlation matrix C is the variance in the sample along the k^{th} PC
- Reveals internal structure that best explains variance in the dataset



The least-squares view: PCs are a series of linear least squares fits to a sample, each orthogonal to all previous ones

https://medium.com/@kayalgen/principal-component-analysis-pca-for-machine-learning-89db6cb63e4

PCA in scikit-learn

sklearn.decomposition.PCA

class sklearn.decomposition.**PCA**(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None*) [source]

Methods

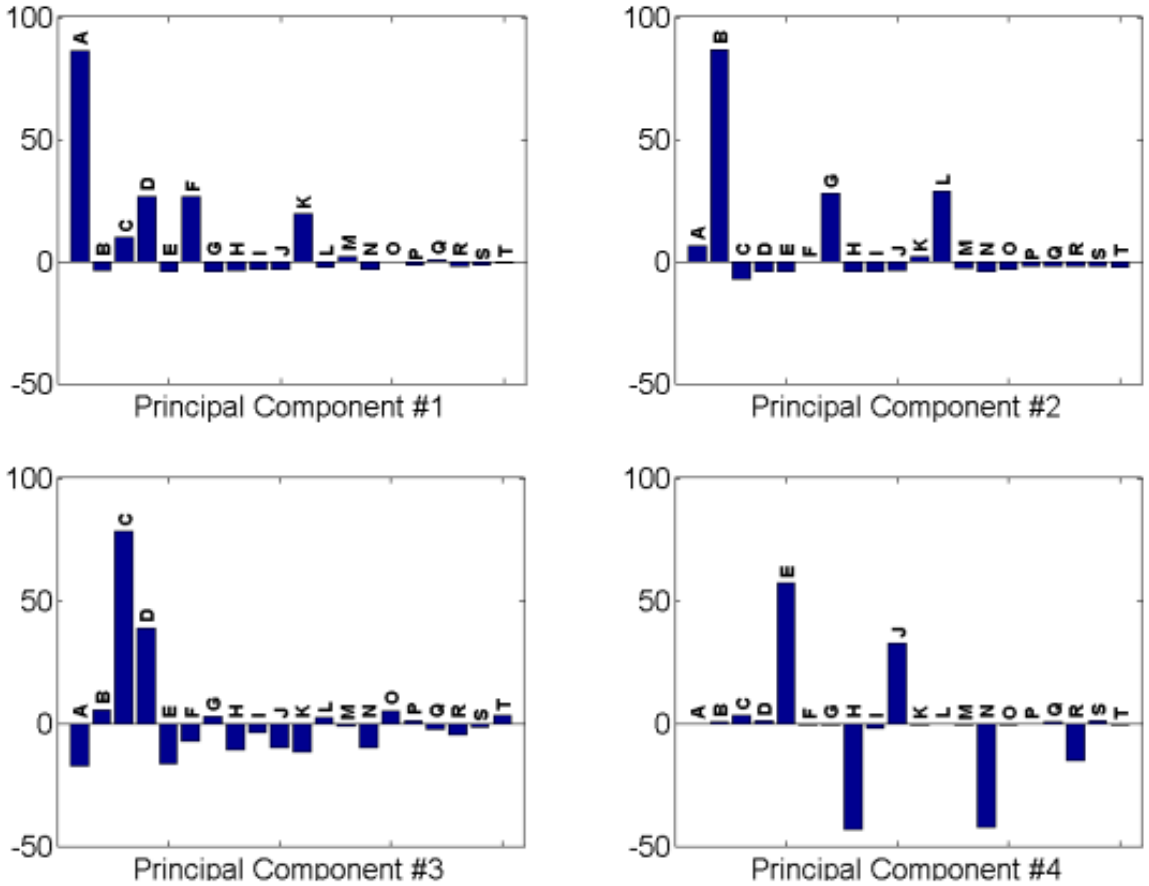
fit(X[, y])	Fit the model with X.
fit_transform(X[, y])	Fit the model with X and apply the dimensionality reduction on X.
get_covariance()	Compute data covariance with the generative model.
get_feature_names_out([input_features])	Get output feature names for transformation.
get_metadata_routing()	Get metadata routing of this object.
get_params([deep])	Get parameters for this estimator.
get_precision()	Compute data precision matrix with the generative model.
inverse_transform(X)	Transform data back to its original space.
score(X[, y])	Return the average log-likelihood of all samples.
score_samples(X)	Return the log-likelihood of each sample.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Apply dimensionality reduction to X.

- In scikit-learn, PCA is implemented as **probabilistic model**: can estimate the likelihood of data based on the amount of variance it explains.
- **Incremental PCA:** add partial_fit method
- **SparsePCA:** introduce sparsity in decomposition

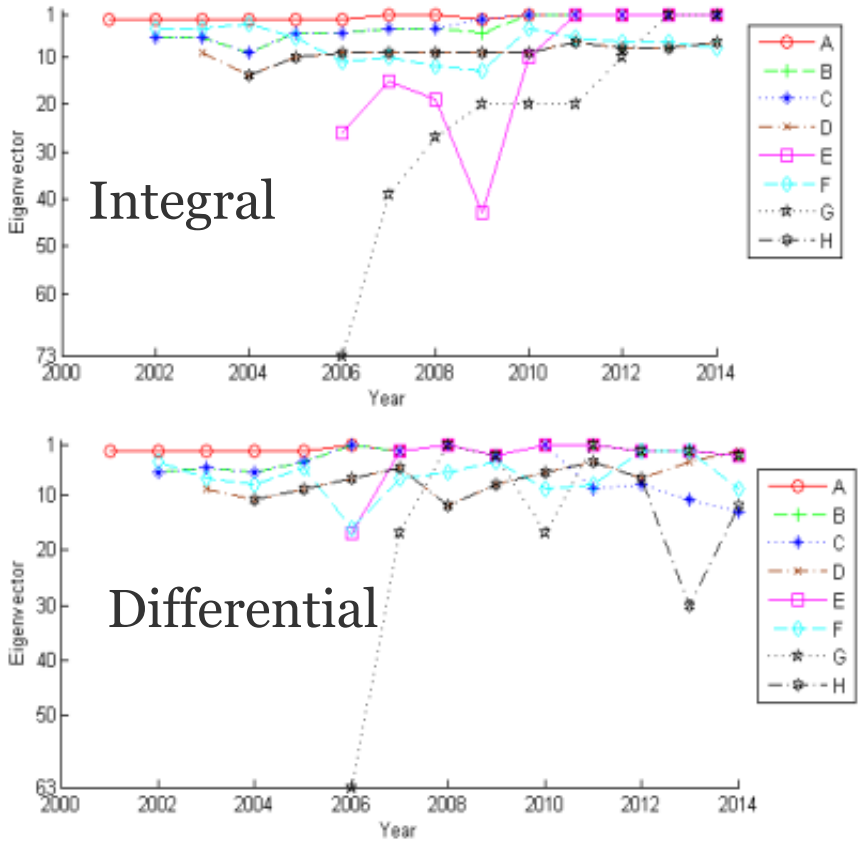
$$(U^*, V^*) = \underset{U,V}{\arg\min} \frac{1}{2}||X - UV||_{\text{Fro}}^2 + \alpha||V||_{1,1}$$
$$\text{subject to } ||U_k||_2 <= 1 \text{ for all } 0 \leq k < n_{components}$$

- … and there is always more (dictionary methods, etc)

PCA can be applied to everything



Publication activity



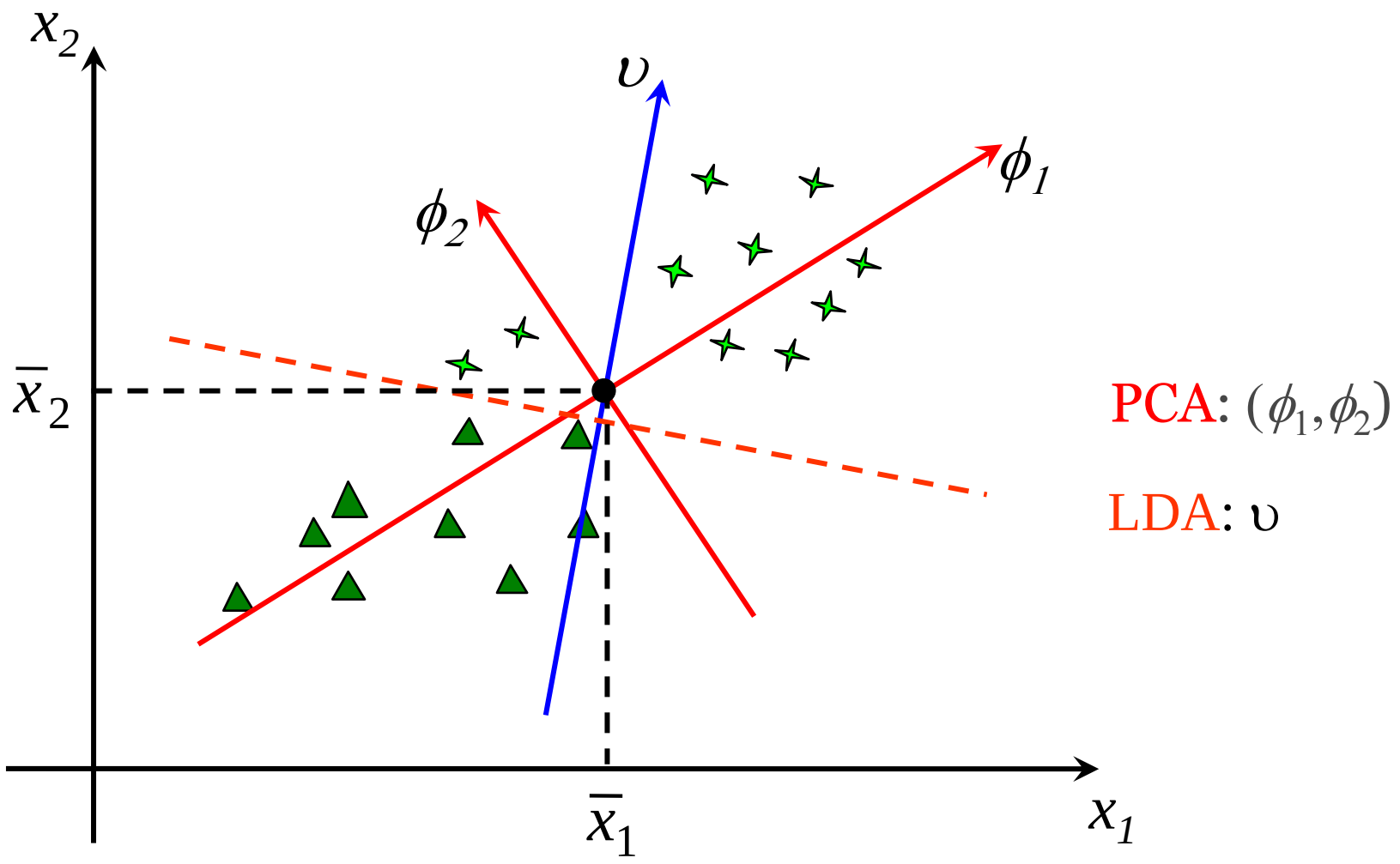
Publication analysis: The dimensionality of space, N, is defined by the total number of scientists. Each publication in this case then defines a single point in this space. For example, for the 28-dimensional space of authors of {A,B,C, … , Z} the publication authored by A and C will be represented as {1,0,1, …, 0}, and by A,B, and Z as {1,1,0, …, 0,1}, where all missing elements are zeroes.

https://arxiv.org/abs/1502.03439

Linear Discriminant Analysis

- Linear Discriminant Analysis, or simply LDA, is a feature extraction technique that has been used successfully in many statistical pattern recognition problems.
- LDA is often called Fisher Discriminant Analysis (FDA).
- The primary purpose of LDA is to separate samples of distinct groups by transforming then to a space which maximises their between-class separability while minimising their within-class variability.
- It assumes implicitly that the true covariance matrices of each class are equal because the same within-class scatter matrix is used for all the classes considered.
- If we remove this assumption, we get Quadratic Discriminant Analysis

Geometric Idea of LDA



From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Steps

1. Compute the d-dimensional mean vectors.
2. Compute the scatter matrices
3. Compute the eigenvectors and corresponding eigenvalues for the scatter matrices.
4. Sort the eigenvalues and choose those with the largest eigenvalues to form a $d \times k$ dimensional matrix
5. Transform the samples onto the new subspace.

LDA Method

- Let the between-class scatter matrix S_b be defined as

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

- and the within-class scatter matrix S_w be defined as

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

- where $x_{i,j}$ is an n-dimensional data point j from class p_i, N_i is the number of training examples from class p_i, and g is the total number of classes or groups.

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Method

- The sample mean, sample covariance, and grand mean vector are given respectively by:

$$\bar{x}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{i,j}$$

$$S_i = \frac{1}{(N_i - 1)}\sum_{j=1}^{N_i}(x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

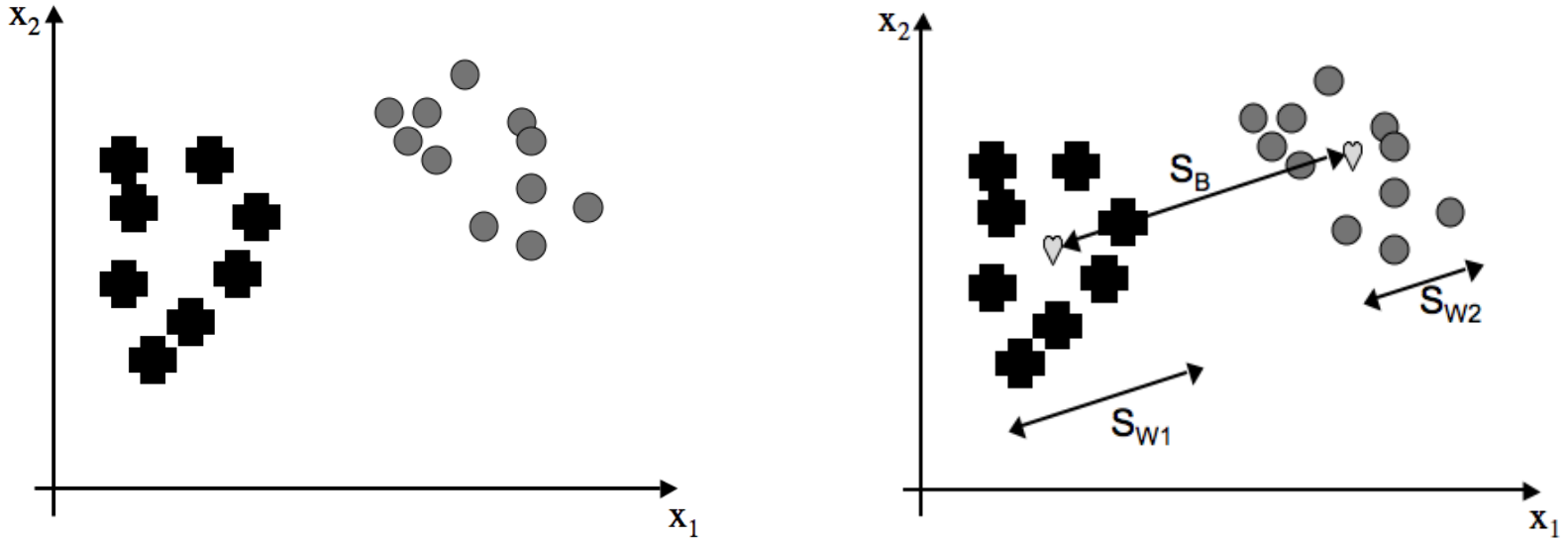
$$\bar{x} = \frac{1}{N}\sum_{i=1}^{g} N_i \bar{x}_i = \frac{1}{N}\sum_{i=1}^{g}\sum_{j=1}^{N_i} x_{i,j} \qquad N = N_1 + N_2 + \ldots + N_g$$

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Method

The objective of LDA is to find a projection matrix P_{lda} that maximises the ratio of the determinant of S_b to the determinant of S_w (Fisher's criterion) as:

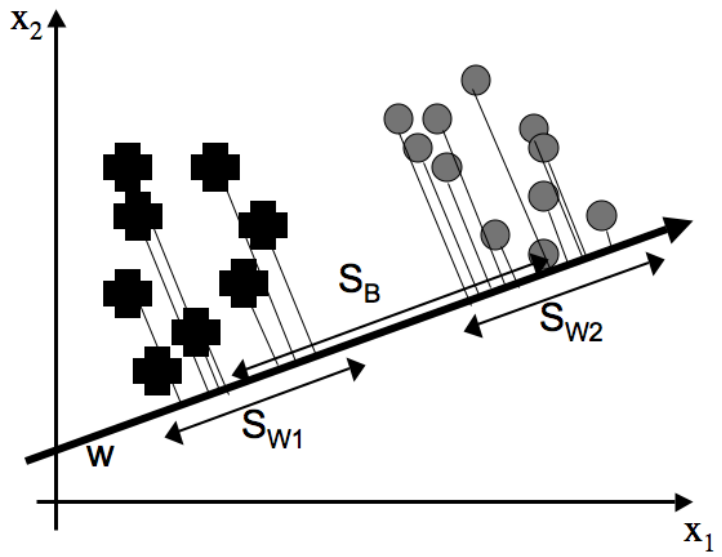
$$P_{lda} = \arg\max_{P} \frac{\left| P^T S_b P \right|}{\left| P^T S_w P \right|}$$



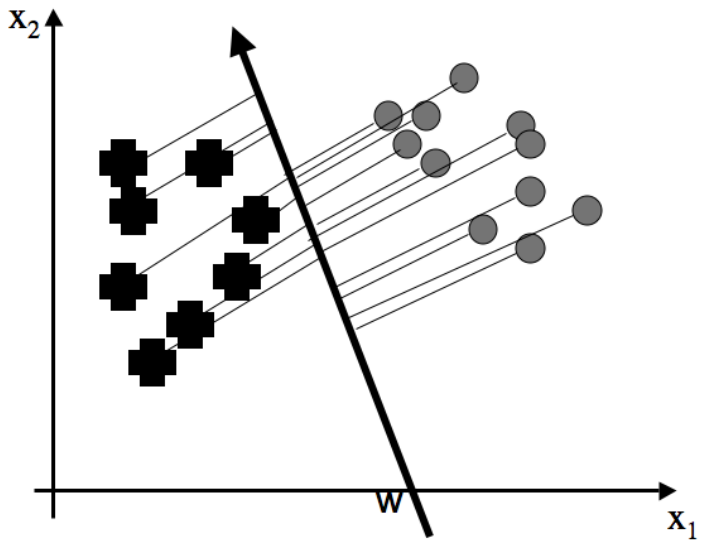
From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Method

- So Fisher's criterion tries to find the projection that:
  - Maximises the variance of the class means
  - Minimises the variance of the individual classes



Good projection



Bad projection

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

Classification using LDA

- The LDA is an axis projection.
- Once the projection is found all the data points can be transformed to the new axis system along with the class means and covariances.
- Allocation of a new point to a class can be done using a distance measure such as the Mahalanobis distance.

Given a vector $\mathbf{x}$, a mean vector μ, and a covariance matrix $\mathbf{\Sigma}$, the Mahalanobis Distance D_M is defined as:

$$D_M(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)}$$

where:

- $\mathbf{x}$ is the vector for which the Mahalanobis Distance is being computed.
- μ is the mean vector of the distribution.
- $\mathbf{\Sigma}$ is the covariance matrix of the distribution.
- ^{T} denotes a matrix transpose.
- $\mathbf{\Sigma}^{-1}$ is the inverse of the covariance matrix.

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA vs. PCA

- LDA is supervised ML method, PCA is unsupervised method
- LDA seeks directions that are efficient for *discriminating* data whereas PCA seeks directions that are efficient for *representing* data.
- The directions that are discarded by PCA might be exactly the directions that are necessary for distinguishing between groups.

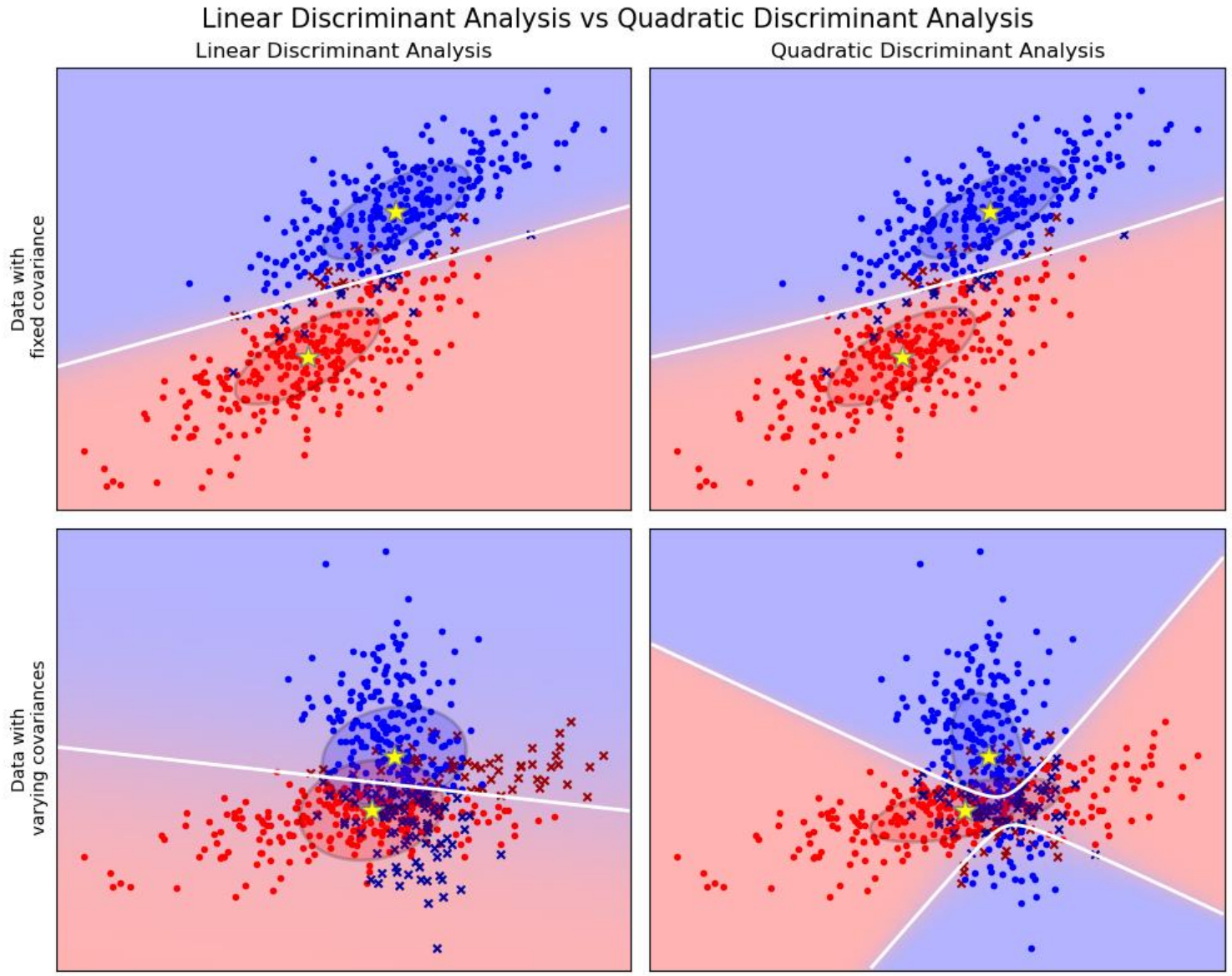
From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

sklearn.discriminant_analysis.LinearDiscriminantAnalysis

class sklearn.discriminant_analysis.**LinearDiscriminantAnalysis**(*solver='svd', shrinkage=None, priors=None, n_components=None, store_covariance=False, tol=0.0001, covariance_estimator=None*) [source]

decision_function(X)	Apply decision function to an array of samples.
fit(X, y)	Fit the Linear Discriminant Analysis model.
fit_transform(X[, y])	Fit to data, then transform it.
get_feature_names_out([input_features])	Get output feature names for transformation.
get_metadata_routing()	Get metadata routing of this object.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict class labels for samples in X.
predict_log_proba(X)	Estimate log probability.
predict_proba(X)	Estimate probability.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
set_score_request(*[, sample_weight])	Request metadata passed to the `score` method.
transform(X)	Project data to maximize class separation.

Quadratic Discriminant Analysis



https://scikit-learn.org/stable/modules/lda_qda.html#lda-qda

LDA vs. QDA

Bayesian classification framework:

$$P(y=k|x) = \frac{P(x|y=k)P(y=k)}{P(x)} = \frac{P(x|y=k)P(y=k)}{\sum_l P(x|y=l) \cdot P(y=l)}$$

QDA approximation:

$$P(x|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\right)$$

LDA is a special case of QDA, where the Gaussians for each class are assumed to share the same covariance matrix: $\Sigma_k = \Sigma$. If so

$$\log P(y=k|x) = \omega_k^t x + \omega_{k0} + Cst.$$

where $\omega_k = \Sigma^{-1}\mu_k$ and $\omega_{k0} = -\frac{1}{2}\mu_k^t \Sigma^{-1}\mu_k + \log P(y=k)$. These quantities correspond to the `coef_` and `intercept_` attributes, respectively.

Note: Relation with Gaussian Naive Bayes

If in the QDA model one assumes that the covariance matrices are diagonal, then the inputs are assumed to be conditionally independent in each class, and the resulting classifier is equivalent to the Gaussian Naive Bayes classifier `naive_bayes.GaussianNB`.

https://scikit-learn.org/stable/modules/lda_qda.html#lda-qda