Content-Based Image Retrieval for Pore Images

Christopher Thiele, Detlef Hohl, Nishank Saxena
Shell International Exploration & Production, Inc.

Overview

- What is content-based image retrieval (CBIR)?
- CBIR for pore images: Motivation
- Getting familiar with the data
- Feature extraction and neighborhood search
- Traditional feature extraction and neural networks
- Various network architectures and training approaches
- Outlook on scalability and 3D images
- Conclusions
- Code demonstration

What is content-based image retrieval (CBIR)?

Given an input or query image, our goal is to retrieve “similar” images from a database.

How do we measure similarity?

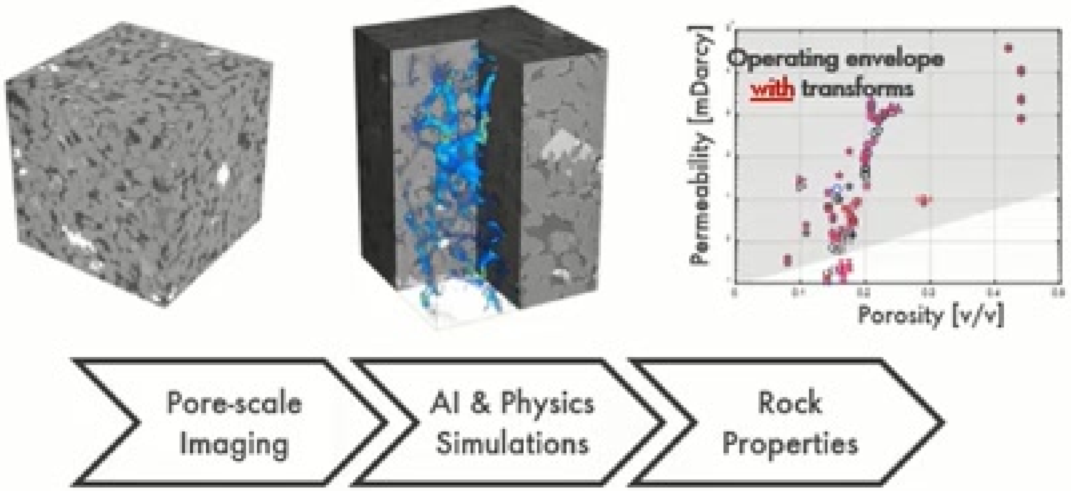
- No rigorous definition in general
- Similarity should be measured based on image content (CBIR) rather than metadata (keywords, etc.).
- Our measure of similarity should not be sensitive to rotation, translation, (moderate) noise, etc.

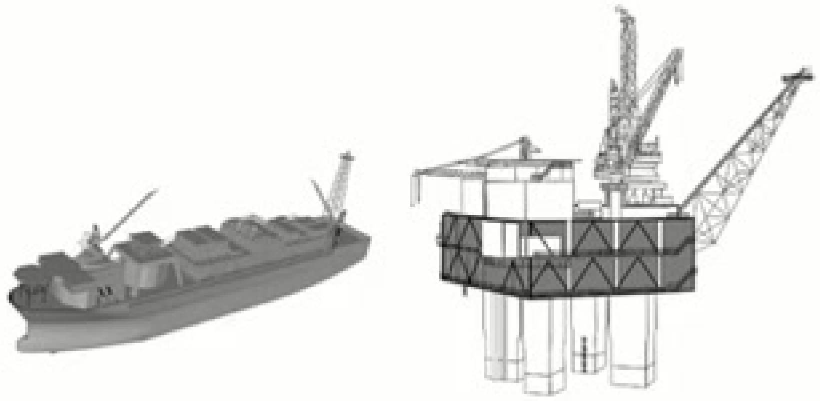
Digital Rock

- Predict rock properties from digital images.
- Incorporates numerical simulations and machine learning approaches.
- Using HPC and cloud resources.

Reservoir Rock

From whole core, sidewall core, or drill cutting





More - improved modeling and efficient analysis
Faster - accelerate delivery of data
Cheaper - use cuttings/side wall cores

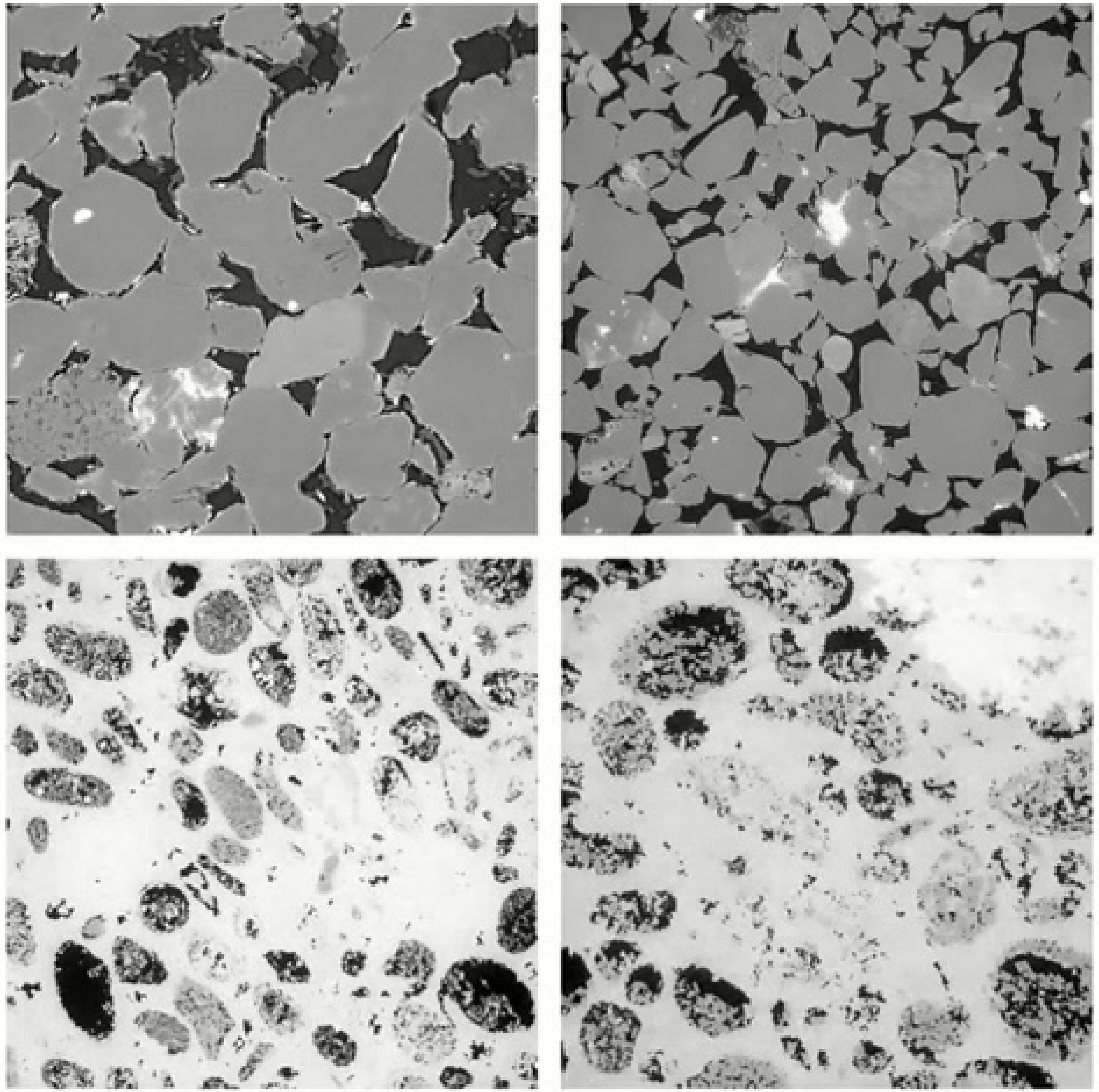
Images provided by Nishank Saxena.

CBIR for pore images: Motivation

The properties of porous rocks in oil and gas reservoirs greatly influence production strategies.

Lab experiments and simulations help to understand these properties, but they are time-consuming and expensive.

How can we identify similar rocks or reservoirs to access existing experimental results?
→ Content-based image retrieval

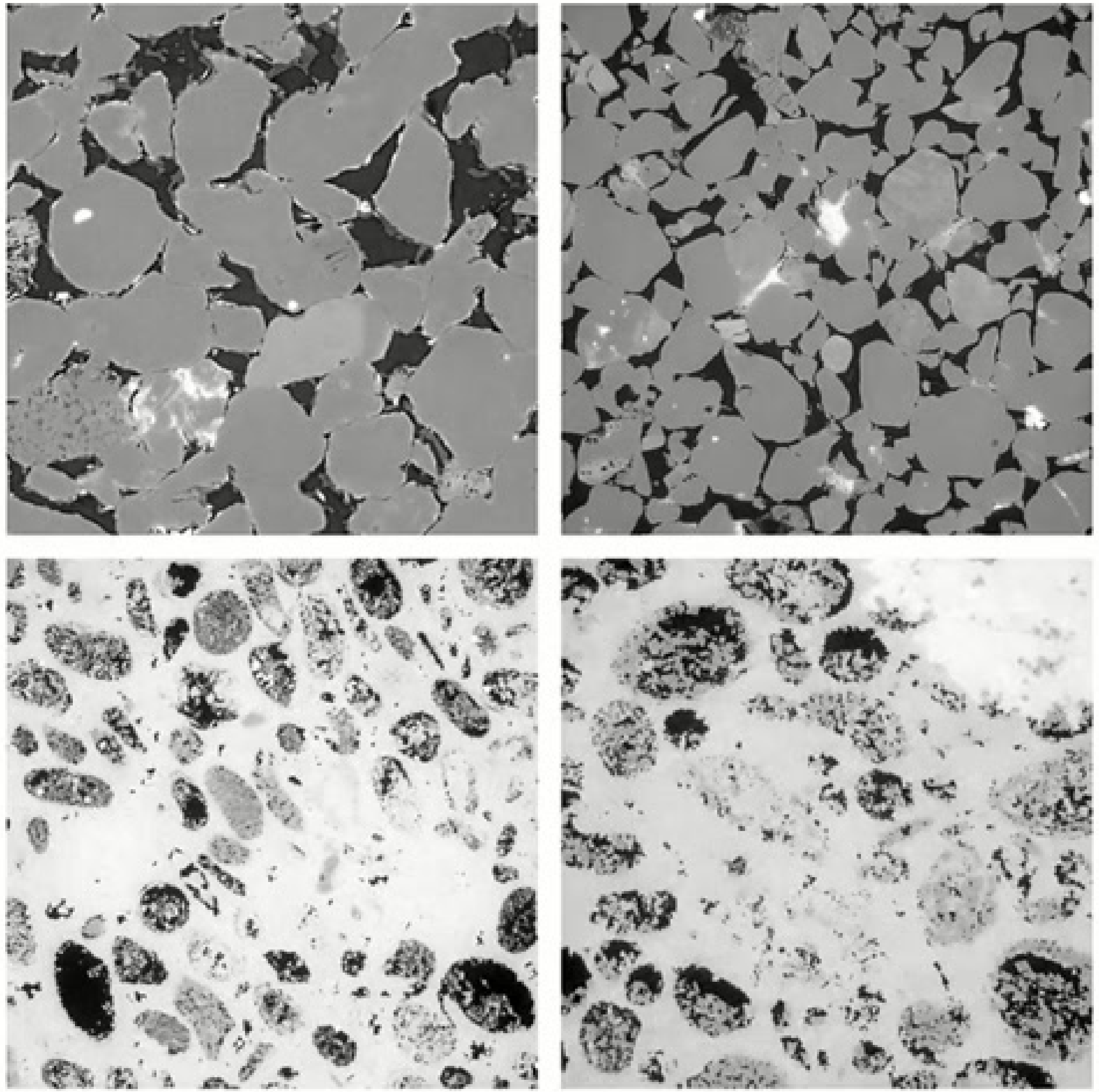


CBIR for pore images: Motivation

CBIR without the aid of computers is infeasible due to the amount of data available.
→ Even more difficult in 3D!

Concept-based image retrieval with labels, keywords, etc. is inherently limited.

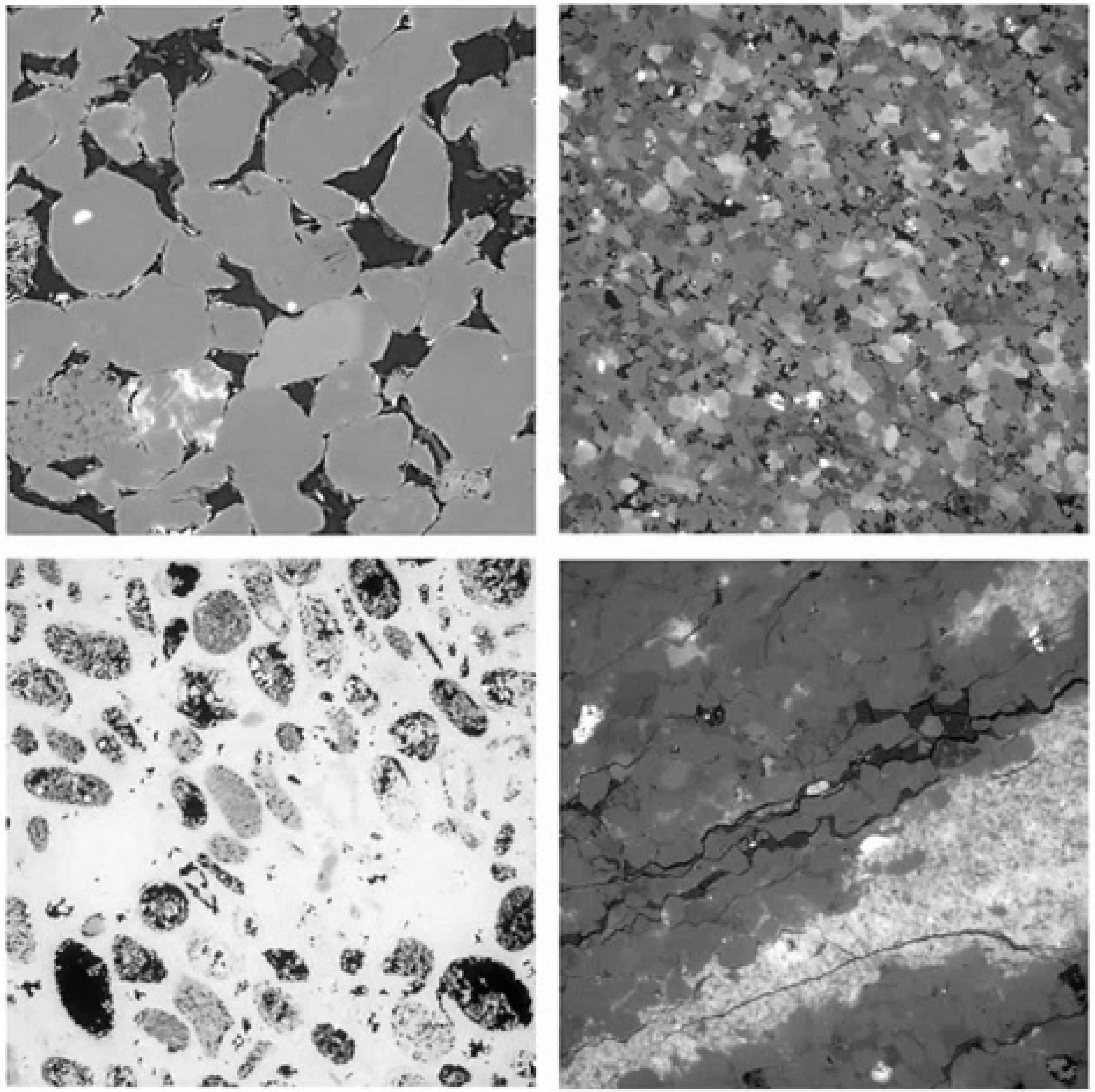
We need computer-aided CBIR to narrow the options for human experts to consider.



Getting familiar with the data

We start with 484 micro-CT images representing slices of 3D rock samples.

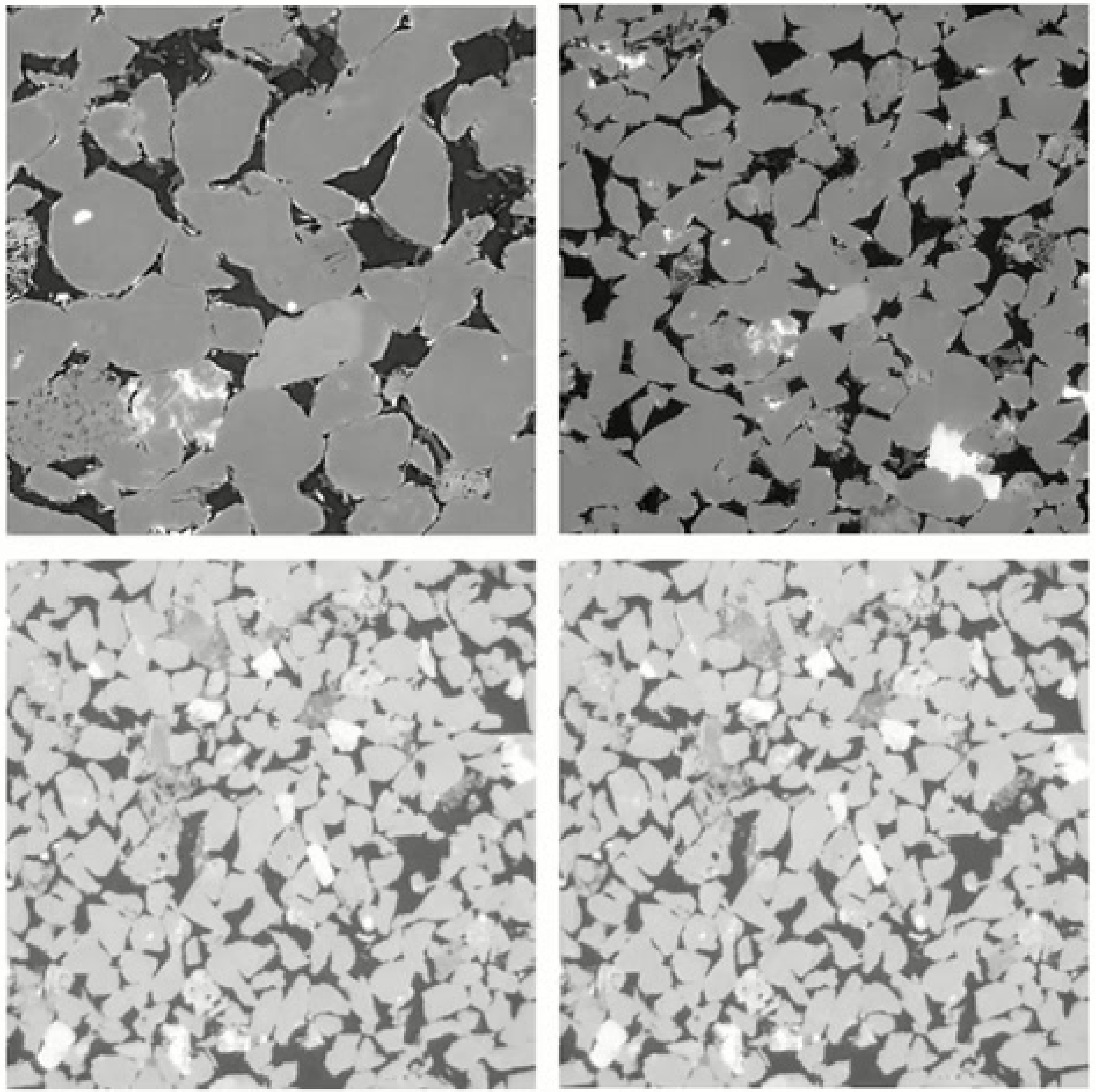
- Resolution: roughly 1,000x1,000 to 2,000x2,000
- 8-bit or 16-bit grayscale



Getting familiar with the data

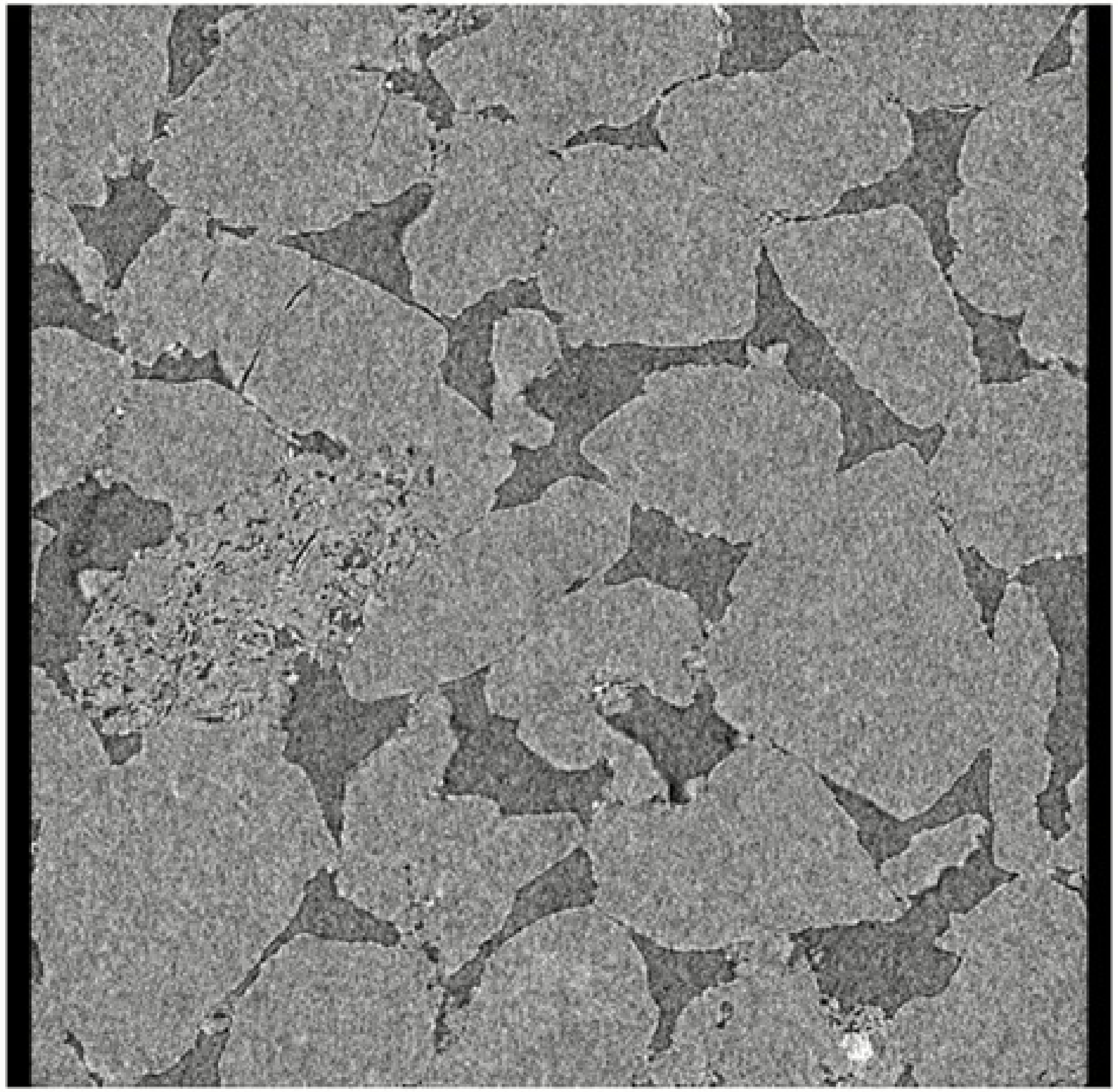
Different images may show the same rock sample with

- different levels of zoom, or
- other small differences.



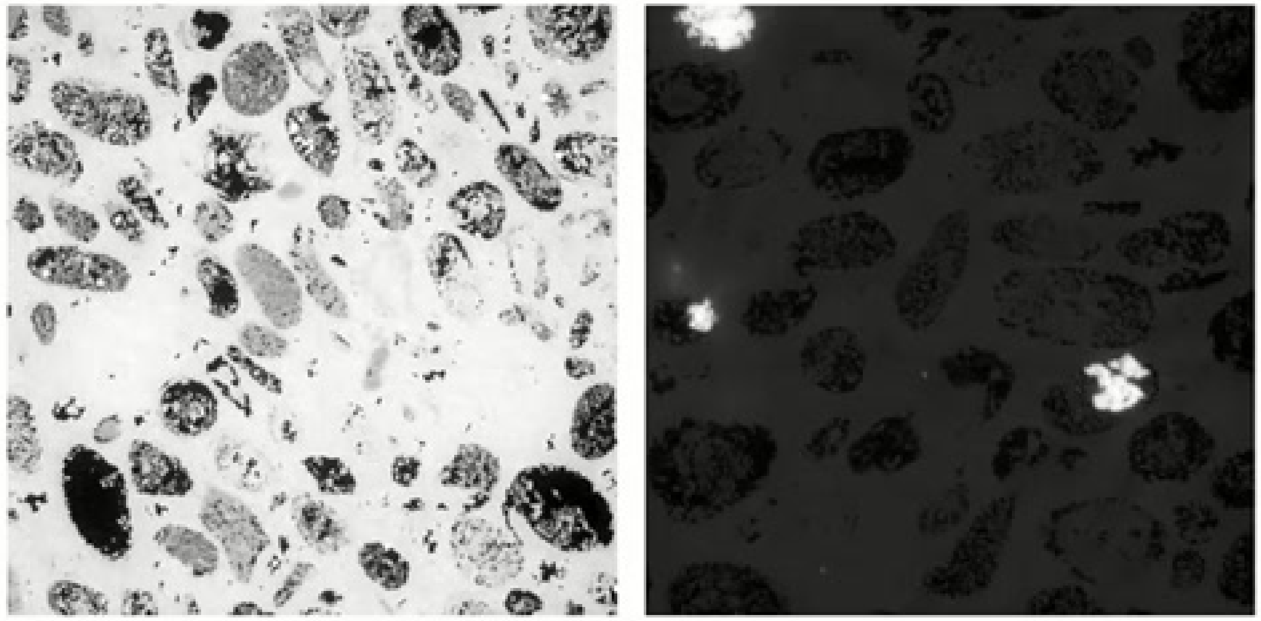
Getting familiar with the data

Images may be of different quality.



Getting familiar with the data

Images may be similar in some ways, but very different in other ways.

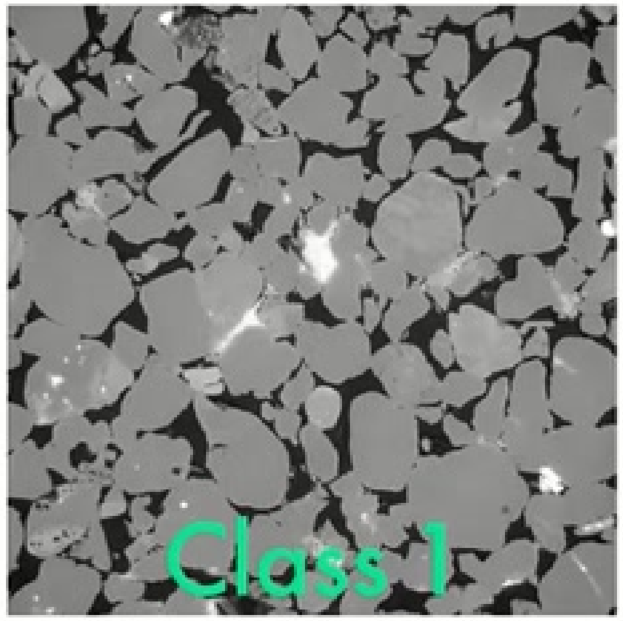


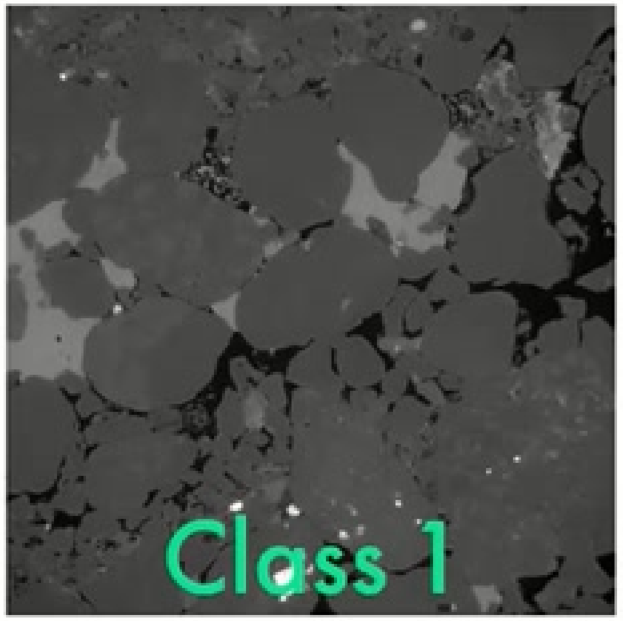
Getting familiar with the data

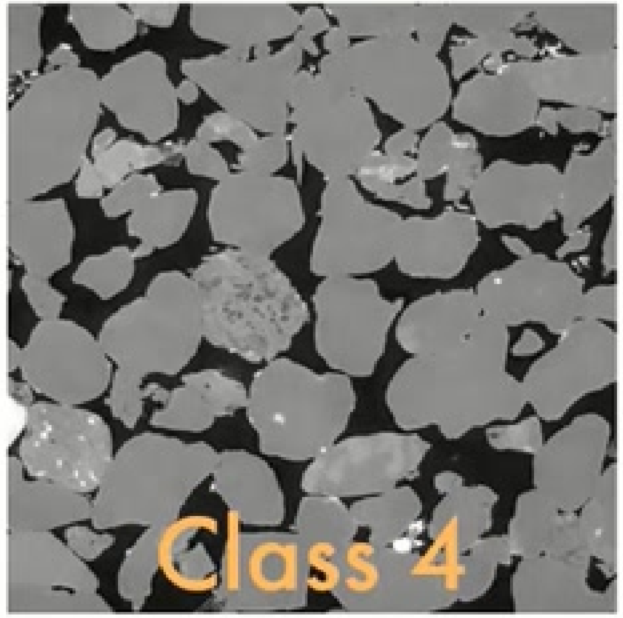
Images are grouped into "anonymized" classes, which may serve as ground truth...

...but can be misleading!

Classes can also be very different in size with 1 to more than 40 images per class.







Feature extraction and neighborhood search

Let us view images as vectors $x_1, \dots, x_n \in \mathbb{R}^{d_1}$. Then we can view CBIR as a two-step procedure:

1. Extract *features* by applying a mapping
$$f: \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$$
to each image, where ideally $d_2 \ll d_1$.
2. Given a query image $x \in \mathbb{R}^{d_1}$, find images $\{x_i\} \subset \mathbb{R}^{d_1}$ such that the feature vectors $f(x_i)$ are close to $f(x)$ according to some distance measure in $\mathbb{R}^{d_2}$, e.g., a metric, a norm, etc.

→ Map images to a low-dimensional space in which image similarity corresponds to distance.

Feature extraction and neighborhood search

Let us view images as vectors $x_1, \dots, x_n \in \mathbb{R}^{d_1}$. Then we can view CBIR as a two-step procedure:

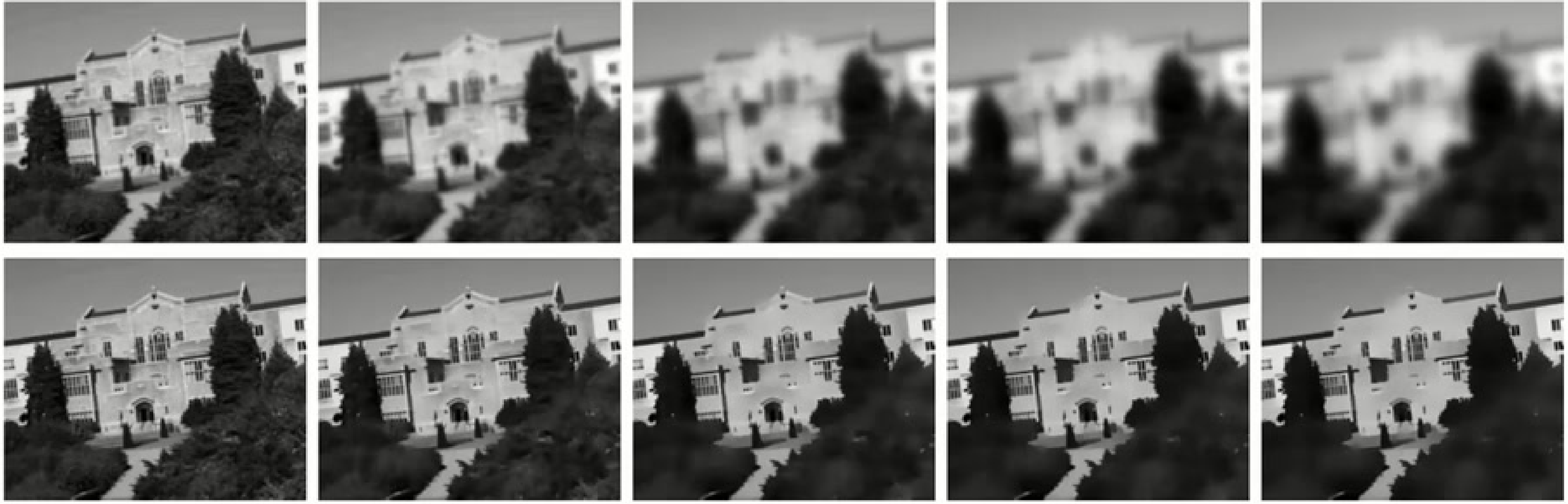
1. Extract *features* by applying a mapping
$$f\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$$
to each image, where ideally $d_2 \ll d_1$.
2. Given a query image $x \in \mathbb{R}^{d_1}$, find images $\{x_i\} \subset \mathbb{R}^{d_1}$ such that the feature vectors $f(x_i)$ are close to $f(x)$ according to some distance measure in $\mathbb{R}^{d_2}$, e.g., a metric, a norm, etc.

→ Map images to a low-dimensional space in which image similarity corresponds to distance.

Traditional feature extraction

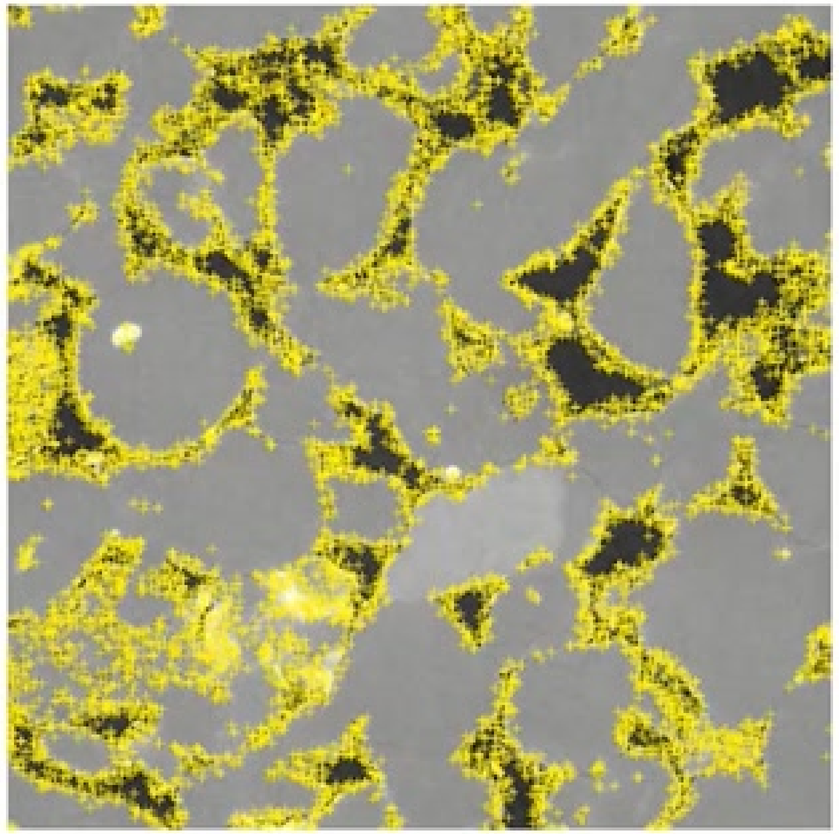
Traditional feature extraction methods construct a *scale* space using filters that reveal features.

Example: Gaussian filters (top) and nonlinear diffusion filters (KAZE[1], bottom)



[1] Image source: Alcantarilla, P.F., Bartoli, A. and Davison, A.J., 2012, October. KAZE features. In European Conference on Computer Vision (pp. 214-227). Springer, Berlin, Heidelberg.

Traditional feature extraction



Filtering reveals key points in the image.
The number of key points depends on the image.

A local image descriptor is computed for each key point (KAZE: normalized vectors in $\mathbb{R}^{64}$).

How can we find images with KAZE features?

1. We cluster features from *all* images in our database using k-means clustering.
2. The cluster centroids form a vocabulary of k (visual) words (bag of features).
3. Each image can be described by a vector in $\mathbb{R}^k$ whose entries are the frequencies of each word.

→ Compare features in $\mathbb{R}^{d_2} = \mathbb{R}^k$.

Feature extraction with neural networks

We can replace the feature extraction mapping $f: \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ with a neural network that has been trained to distinguish (pore) images.

How do we select and train the neural network?

- Existing, pretrained networks (ResNet, VGG, etc.)
- Existing architectures, retrained on pore images
- Custom architectures and training approaches

Feature extraction with neural networks

We can replace the feature extraction mapping $f\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ with a neural network that has been trained to distinguish (pore) images.

How do we select and train the neural network?

- Existing, pretrained networks (ResNet, VGG, etc.)
- Existing architectures, retrained on pore images
- Custom architectures and training approaches

MATLAB implementation

MATLAB made it easy to implement and test various methods:

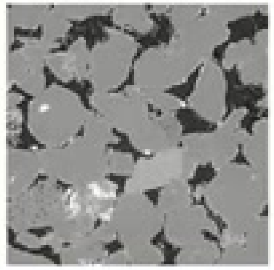
- General image processing functions (Image Processing Toolbox)
- KAZE and bag of features implementation (Computer Vision Toolbox)
- Neural network building blocks and pretrained networks (Deep Learning Toolbox)
- Clustering algorithms (Statistics and Machine Learning Toolbox)

All methods used image data stores as a common starting point.

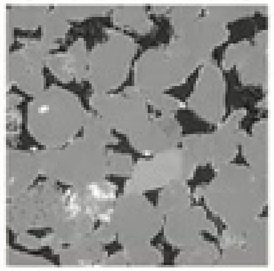
Automatic CPU- and GPU-parallelism (Parallel Computing Toolbox)

Baseline: KAZE features

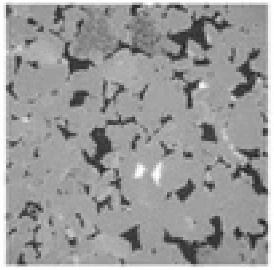
Query image



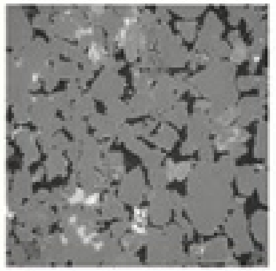
(score: 9.913923e-01)



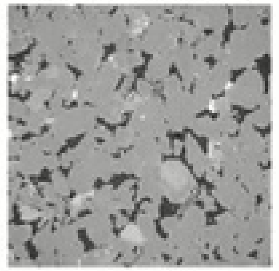
(score: 7.140290e-01)



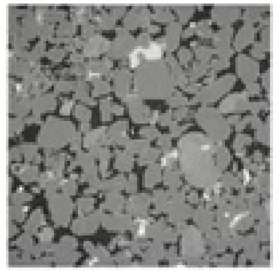
(score: 6.882563e-01)



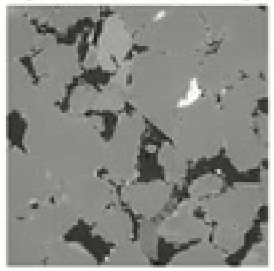
(score: 6.787782e-01)



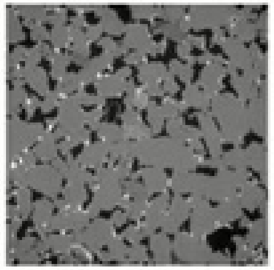
(score: 6.629210e-01)



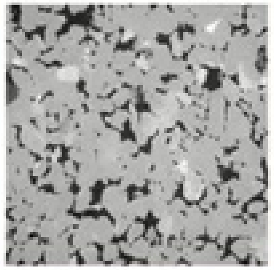
(score: 6.574258e-01)



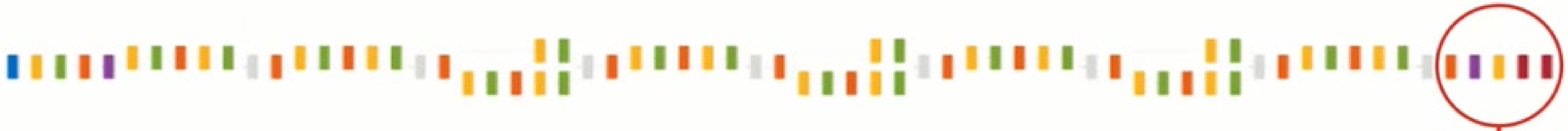
(score: 6.474022e-01)



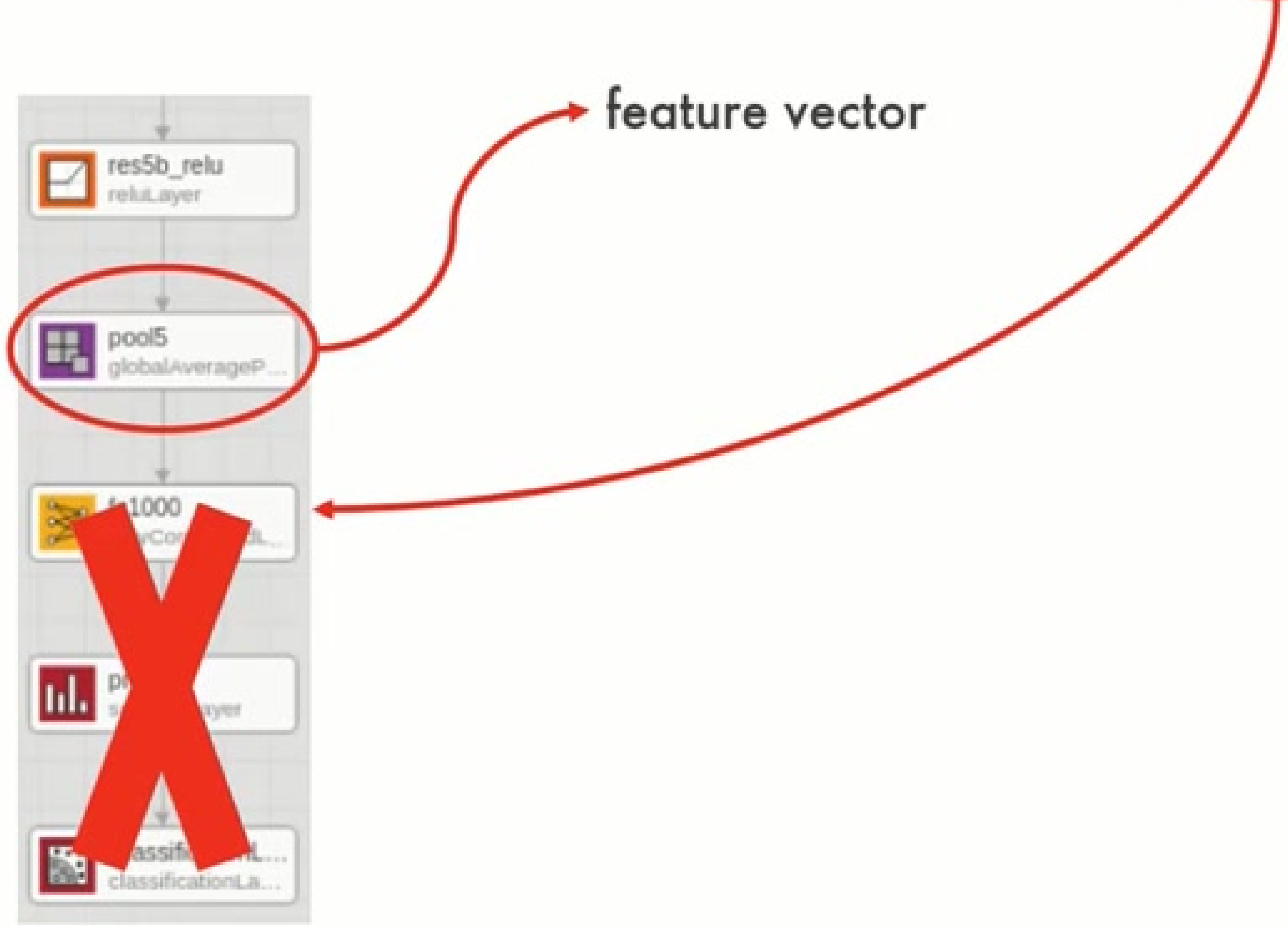
(score: 6.468519e-01)



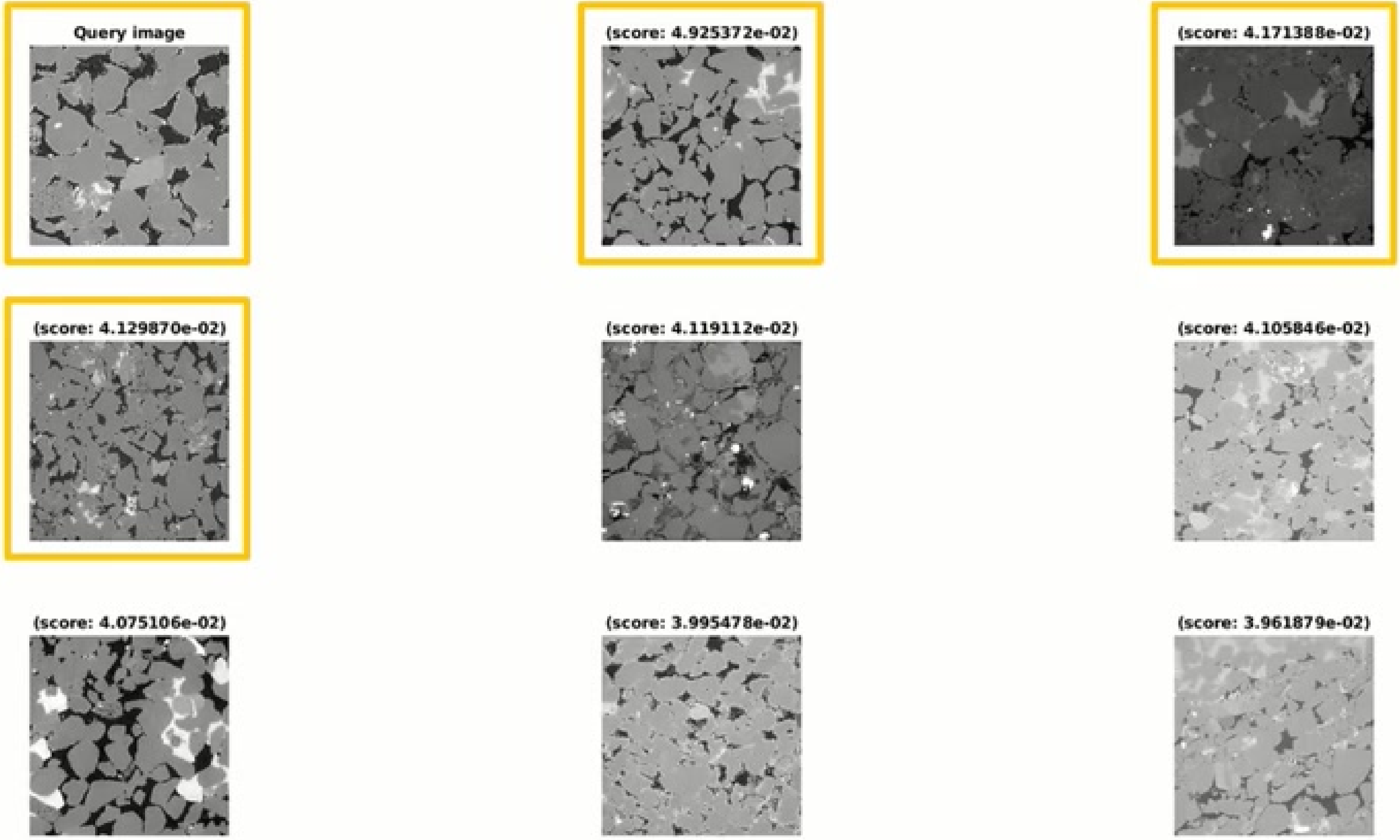
Pretrained networks: ResNet



- Use a ResNet that was trained on the ImageNet data set.
- Remove the classification layers.
- Extract 2048-dimensional features from last pooling layer.



Pretrained networks: ResNet-50



Retraining the ResNet-50

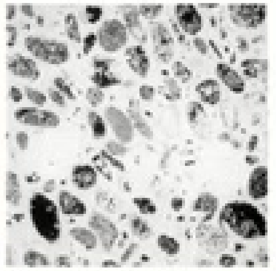
The ResNet-50 architecture was retrained:

- Retrained *with* classification layers, which were discarded after training (as before).
- Used pore image classes as labels.
- Initial weights from training on ImageNet data set.
- Trained with 432 images plus random rotations, added noise, etc.

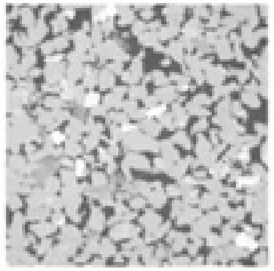
Retraining the ResNet-50

Risk of overfitting!

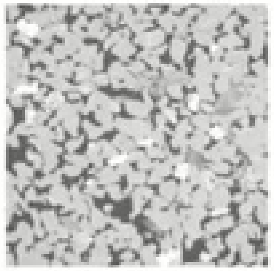
Classification task unsuitable for feature extraction.



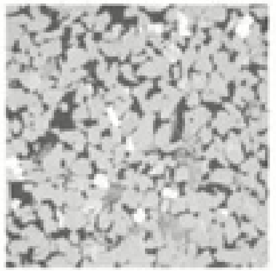
Query image



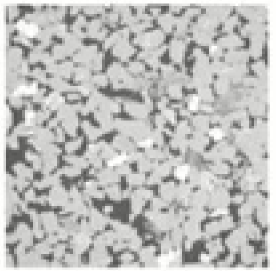
(score: 7.885741e-02)



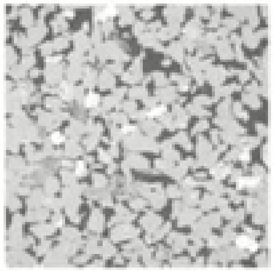
(score: 7.780290e-02)



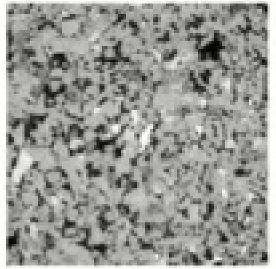
(score: 7.776626e-02)



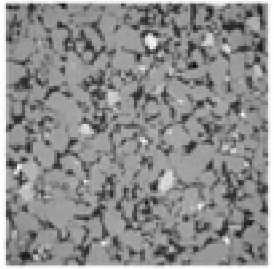
(score: 7.733728e-02)



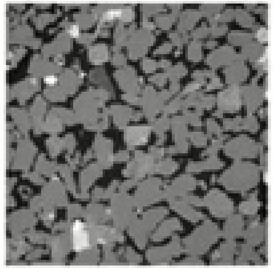
(score: 7.688961e-02)



(score: 7.669538e-02)



(score: 7.642521e-02)



(score: 7.605897e-02)

Custom neural networks for pore images

Two main question:

- Which network architectures are suitable?
- How to train the neural network?

Option 1: Convolutional neural networks

- Trained on classification problem using pore image classes.

Option 2: Convolutional denoising autoencoders

- Learn image encodings and reconstructions in an unsupervised manner.

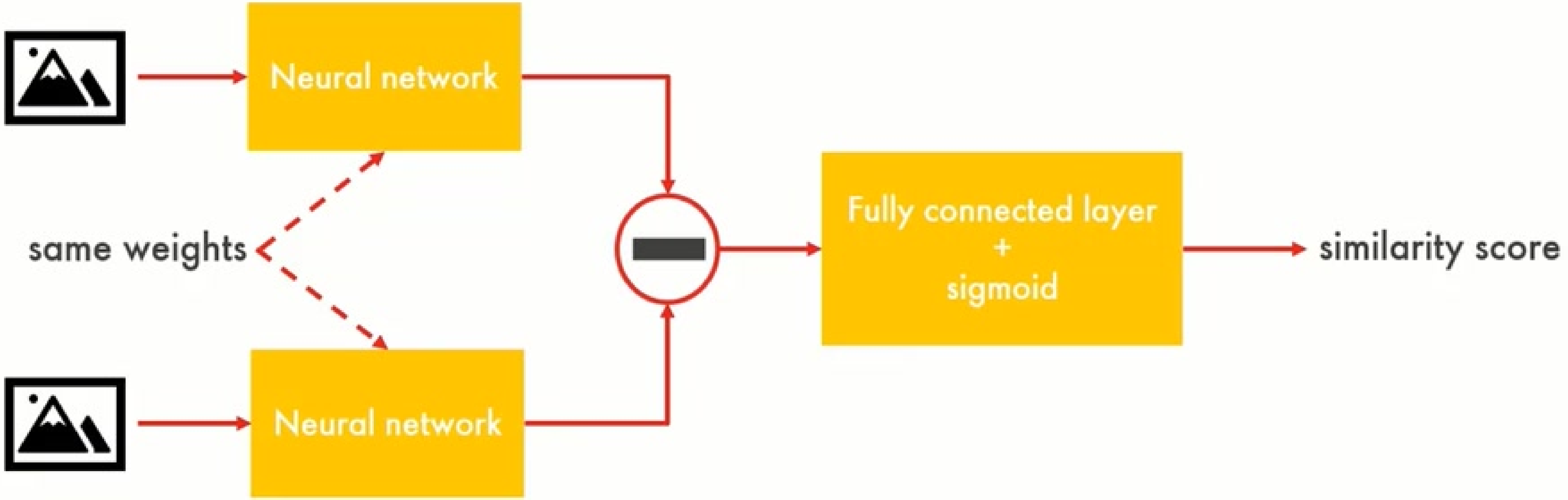
Option 3: Siamese twin networks

- Learn a notion of similarity instead of somewhat artificial classifications or simple reconstructions.

Siamese twin networks

Is there a neural network architecture that can learn similarity directly?

→ Siamese twin networks



MATLAB tutorial: https://www.mathworks.com/help/deeplearning/ug/train-a-siamese-network-to-compare-images.html

Siamese twin networks

During training, the Siamese twin network is presented with pairs of similar or dissimilar images.

How to obtain such pairs?

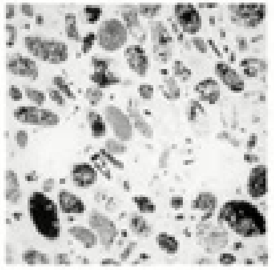
- Option 1: Use pore image classes treat images as similar if and only if they are in the same class.
- Option 2: Create labels manually, i.e., label random pairs.

When loading images, we apply random rotations, add noise, etc. to improve robustness.

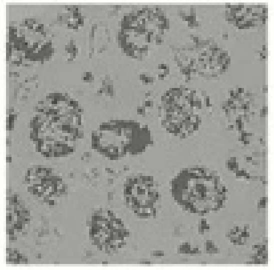
Siamese twin networks

Twin network using the ResNet-50, trained with pore image classes. Feature comparison with *learned* metric.

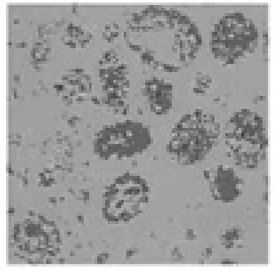
Query image



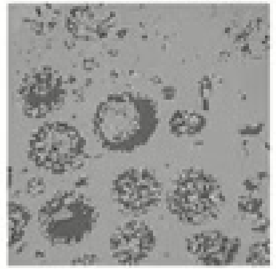
(score: 9.992630e-01)



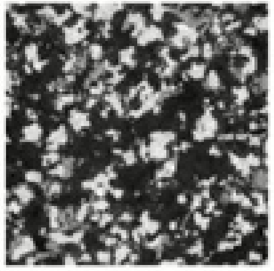
(score: 9.963391e-01)



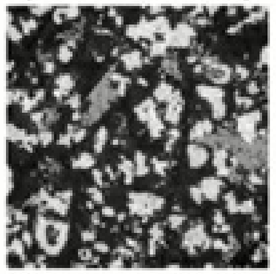
(score: 9.956891e-01)



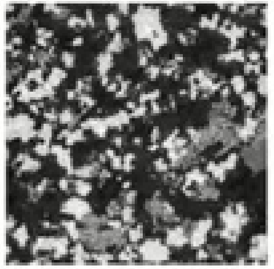
(score: 9.937483e-01)



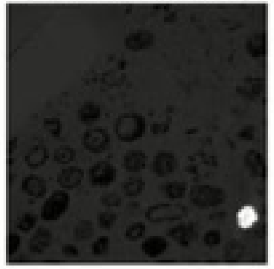
(score: 9.929998e-01)



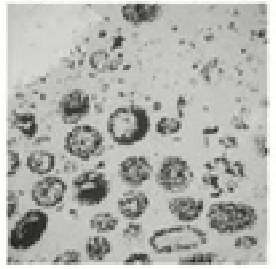
(score: 9.772045e-01)



(score: 9.713053e-01)

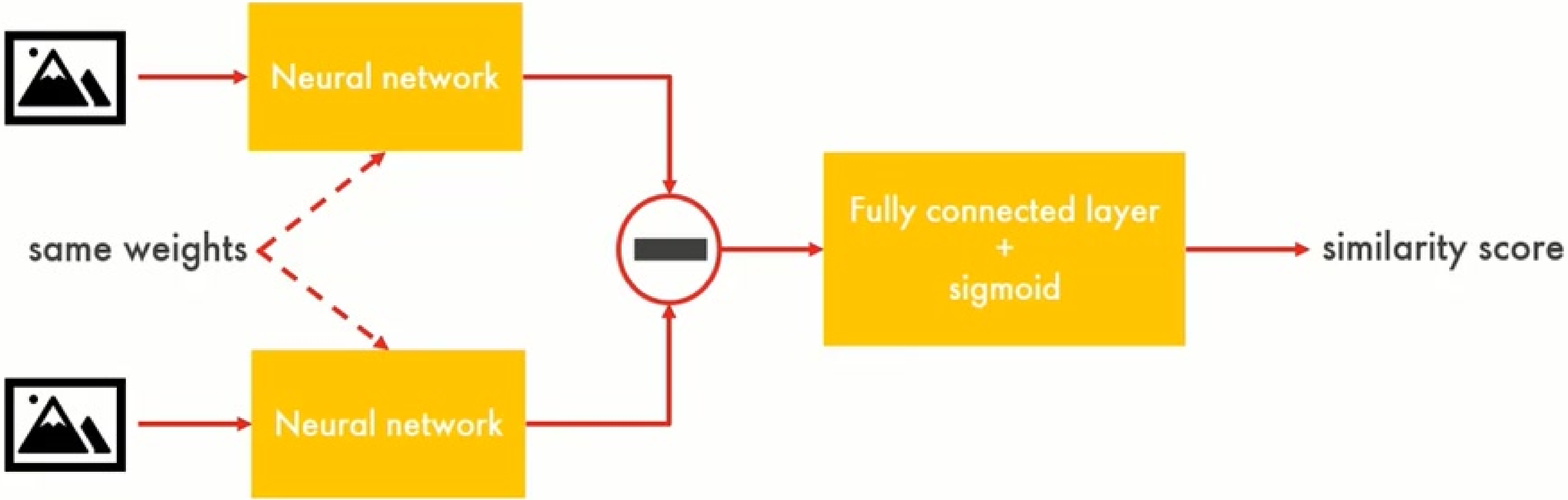


(score: 9.661391e-01)



Siamese twin networks

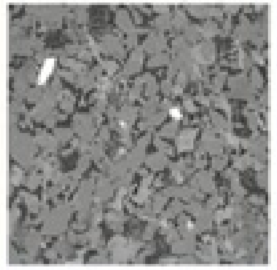
Does the final fully connected layer provide a good metric for feature distances?



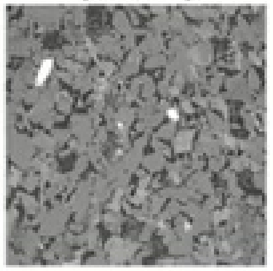
Siamese twin networks

Twin network using the ResNet-50, trained with pore image classes. Feature comparison with L2-norm.

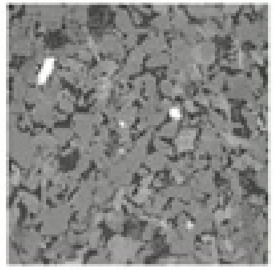
Query image



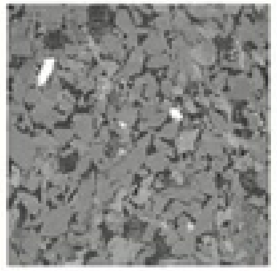
KB09/center-sliceX
(score: Inf)



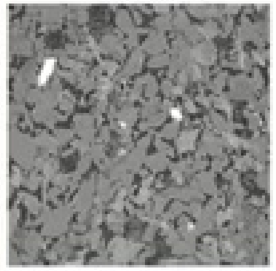
KB10/center-sliceX
(score: Inf)



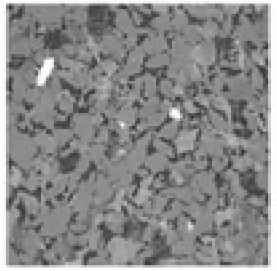
KB11/center-sliceX
(score: Inf)



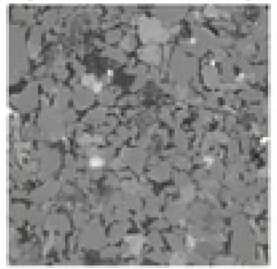
KB12/center-sliceX
(score: Inf)



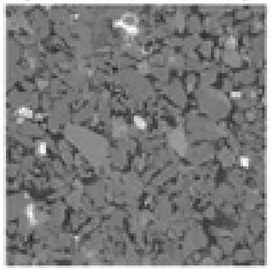
KB13/center-sliceX
(score: Inf)



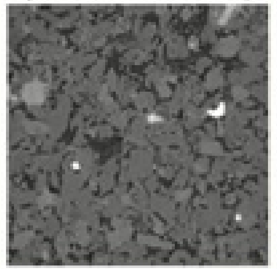
KB03/center-sliceX
(score: 1.237265e-03)



UM02/center-sliceX
(score: 1.159407e-03)



KB08/center-sliceZ
(score: 1.110900e-03)



Outlook on scalability

Applicability of algorithms to 3D images is challenging:

- Convolutional autoencoders and Siamese twin networks should still work.
- Are pretrained networks like ResNet-50 available for 3D images?
- Unclear whether KAZE features extend to 3D.

CBIR for 3D images requires additional parallelism.

Outlook on scalability

Applicability of algorithms to 3D images is challenging:

- Convolutional autoencoders and Siamese twin networks should still work.
- Are pretrained networks like ResNet-50 available for 3D images?
- Unclear whether KAZE features extend to 3D.

CBIR for 3D images requires additional parallelism.

Conclusions

- Content-based image retrieval (CBIR) for pore image is an important component of the digital rock effort.
- Several approaches yield promising results:
  - Traditional feature extraction (KAZE)
  - Pretrained convolutional networks (ResNet-50)
  - Siamese twin networks
- MATLAB made it easy to evaluate and compare these approaches.
- 3D images will present new challenges.

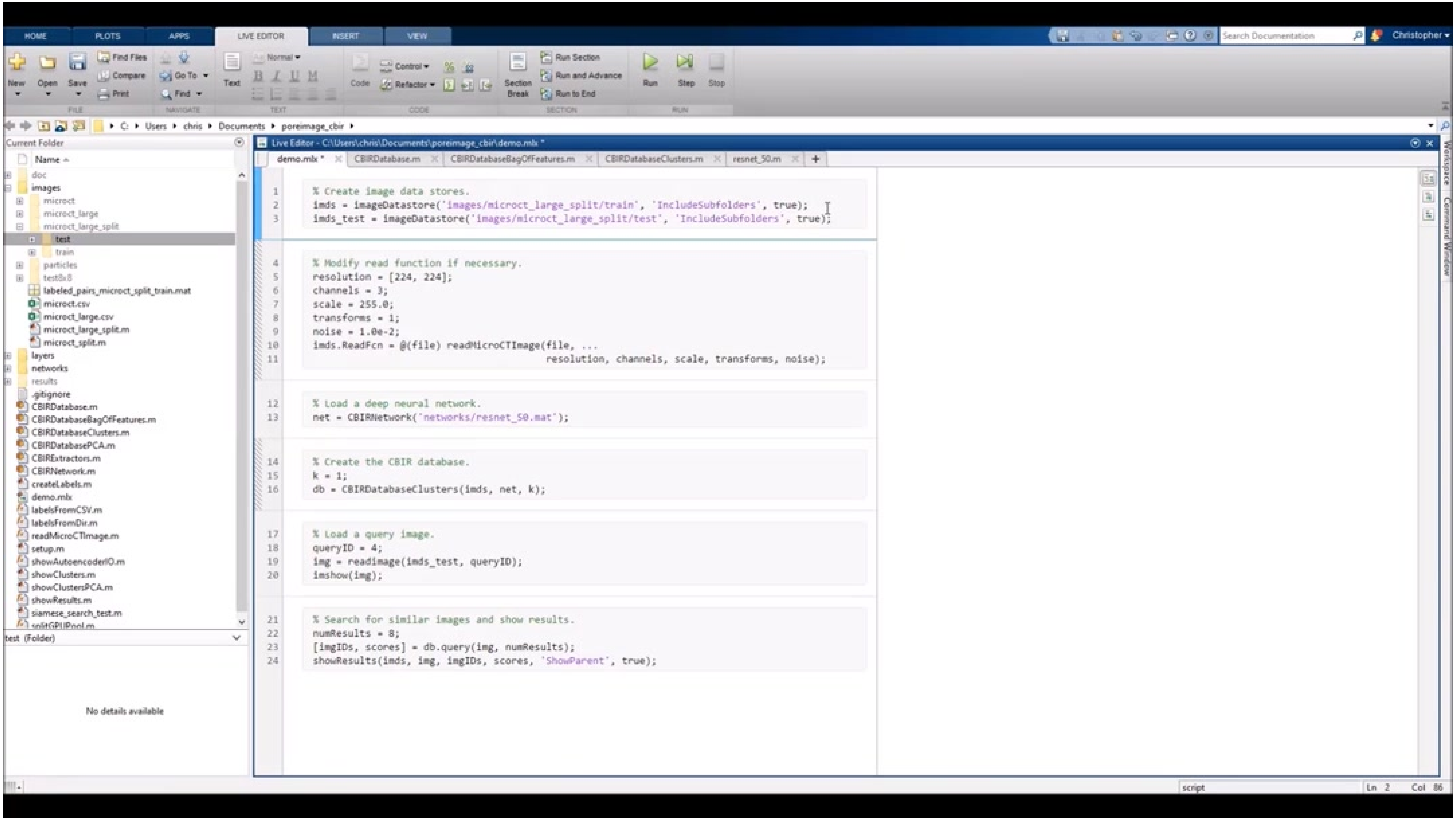
Acknowledgments

We would like to thank

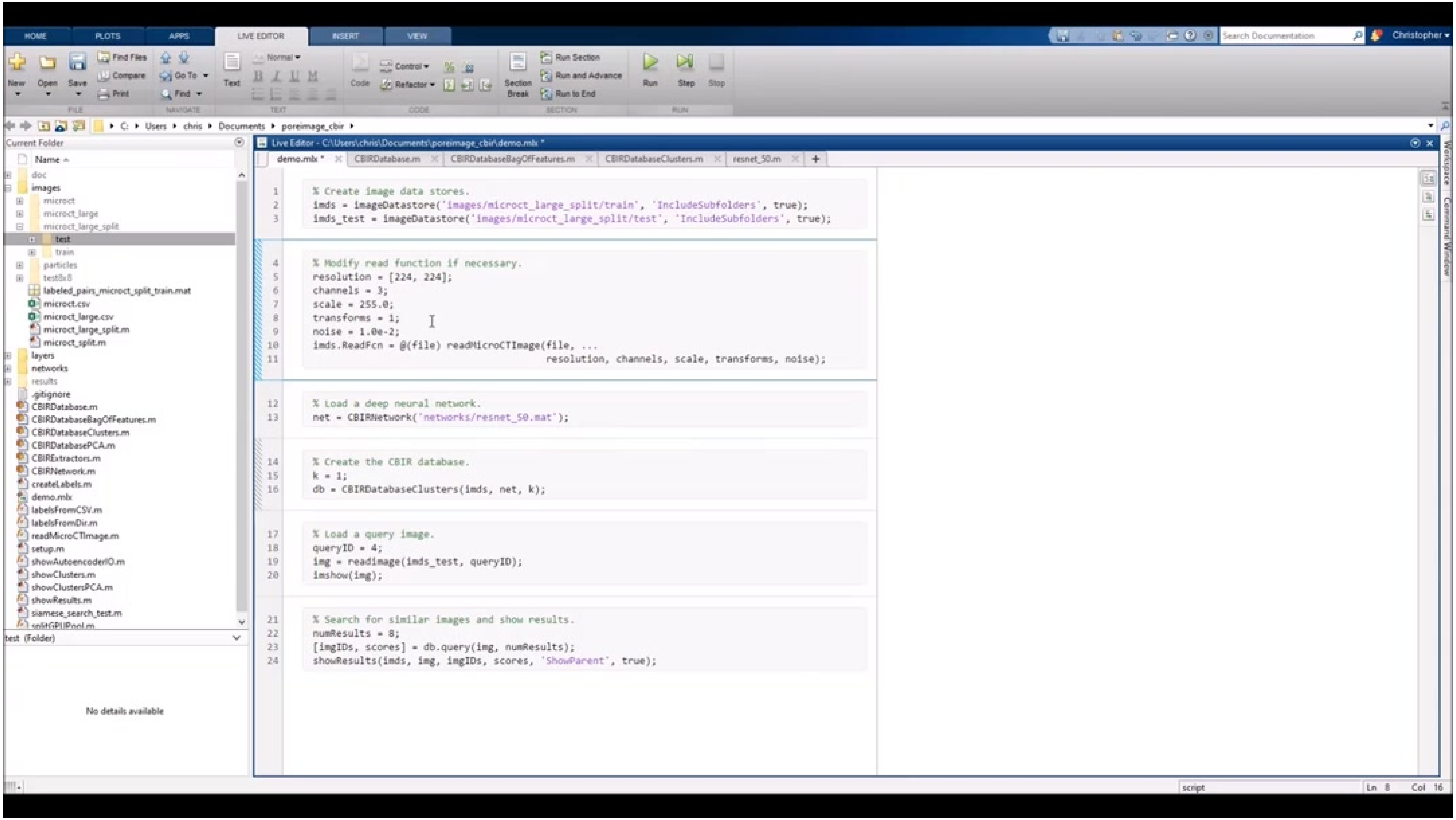
Kunj Tandon, Majeed Shaik, and Lalit Gupta (Bangalore),

Jesse Dietderich and Ronny Hofmann (Houston),

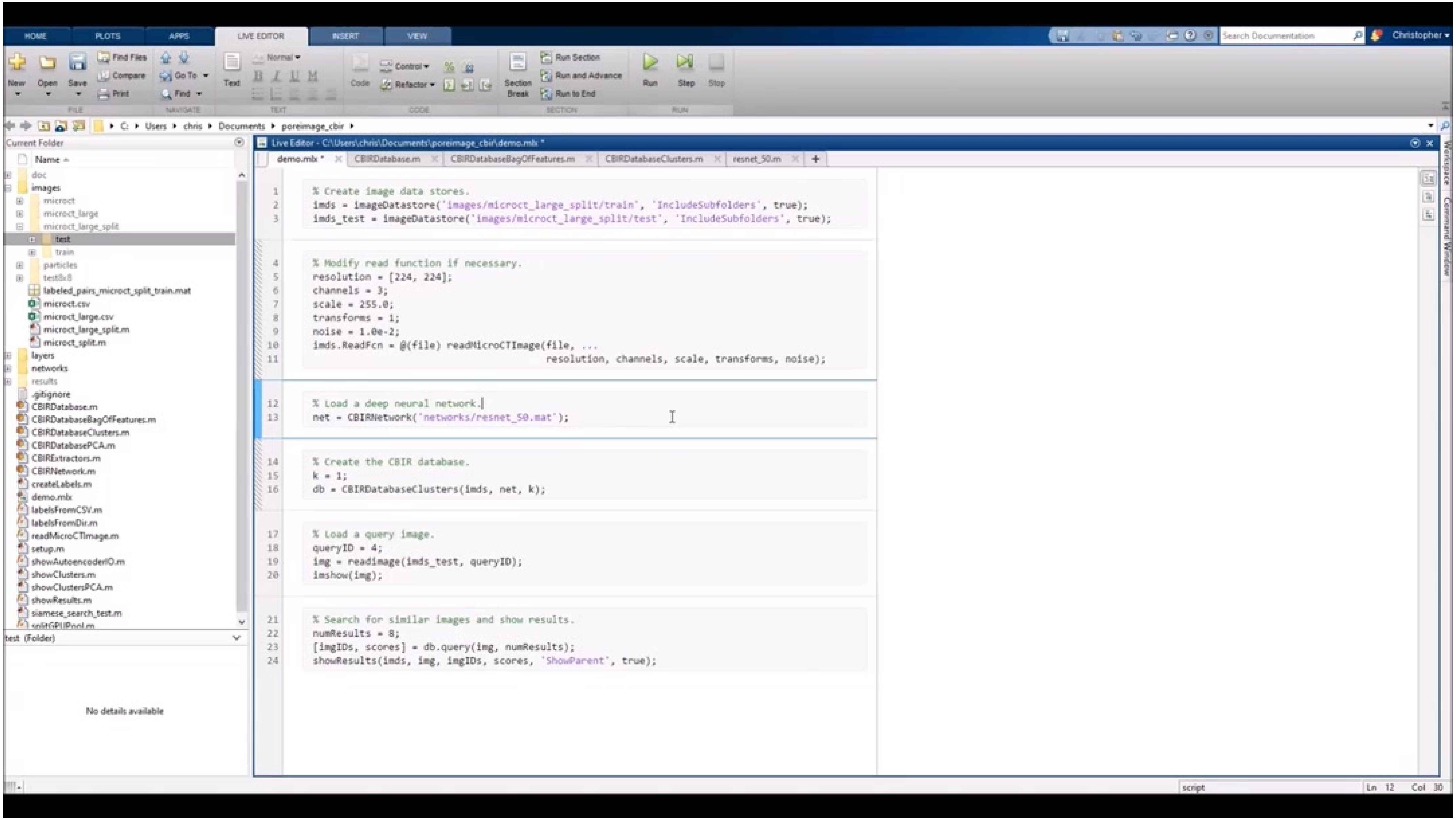
and others who supported this work with their expertise, data, and code.



```
% Create image data stores.
imds = imageDatastore('images/microct_large_split/train', 'IncludeSubfolders', true);
imds_test = imageDatastore('images/microct_large_split/test', 'IncludeSubfolders', true);

% Modify read function if necessary.
resolution = [224, 224];
channels = 3;
scale = 255.0;
transforms = 1;
noise = 1.0e-2;
imds.ReadFcn = @(file) readMicroCTImage(file, ...
                                        resolution, channels, scale, transforms, noise);

% Load a deep neural network.
net = CBIRNetwork('networks/resnet_50.mat');

% Create the CBIR database.
k = 1;
db = CBIRDatabaseClusters(imds, net, k);

% Load a query image.
queryID = 4;
img = readimage(imds_test, queryID);
imshow(img);

% Search for similar images and show results.
numResults = 8;
[imgIDs, scores] = db.query(img, numResults);
showResults(imds, img, imgIDs, scores, 'ShowParent', true);
```



```matlab
% Create image data stores.
imds = imageDatastore('images/microct_large_split/train', 'IncludeSubfolders', true);
imds_test = imageDatastore('images/microct_large_split/test', 'IncludeSubfolders', true);

% Modify read function if necessary.
resolution = [224, 224];
channels = 3;
scale = 255.0;
transforms = 1;
noise = 1.0e-2;
imds.ReadFcn = @(file) readMicroCTImage(file, ...
                                        resolution, channels, scale, transforms, noise);

% Load a deep neural network.
net = CBIRNetwork('networks/resnet_50.mat');

% Create the CBIR database.
k = 1;
db = CBIRDatabaseClusters(imds, net, k);

% Load a query image.
queryID = 4;
img = readimage(imds_test, queryID);
imshow(img);

% Search for similar images and show results.
numResults = 8;
[imgIDs, scores] = db.query(img, numResults);
showResults(imds, img, imgIDs, scores, 'ShowParent', true);
```



```matlab
% Create image data stores.
imds = imageDatastore('images/microct_large_split/train', 'IncludeSubfolders', true);
imds_test = imageDatastore('images/microct_large_split/test', 'IncludeSubfolders', true);
```

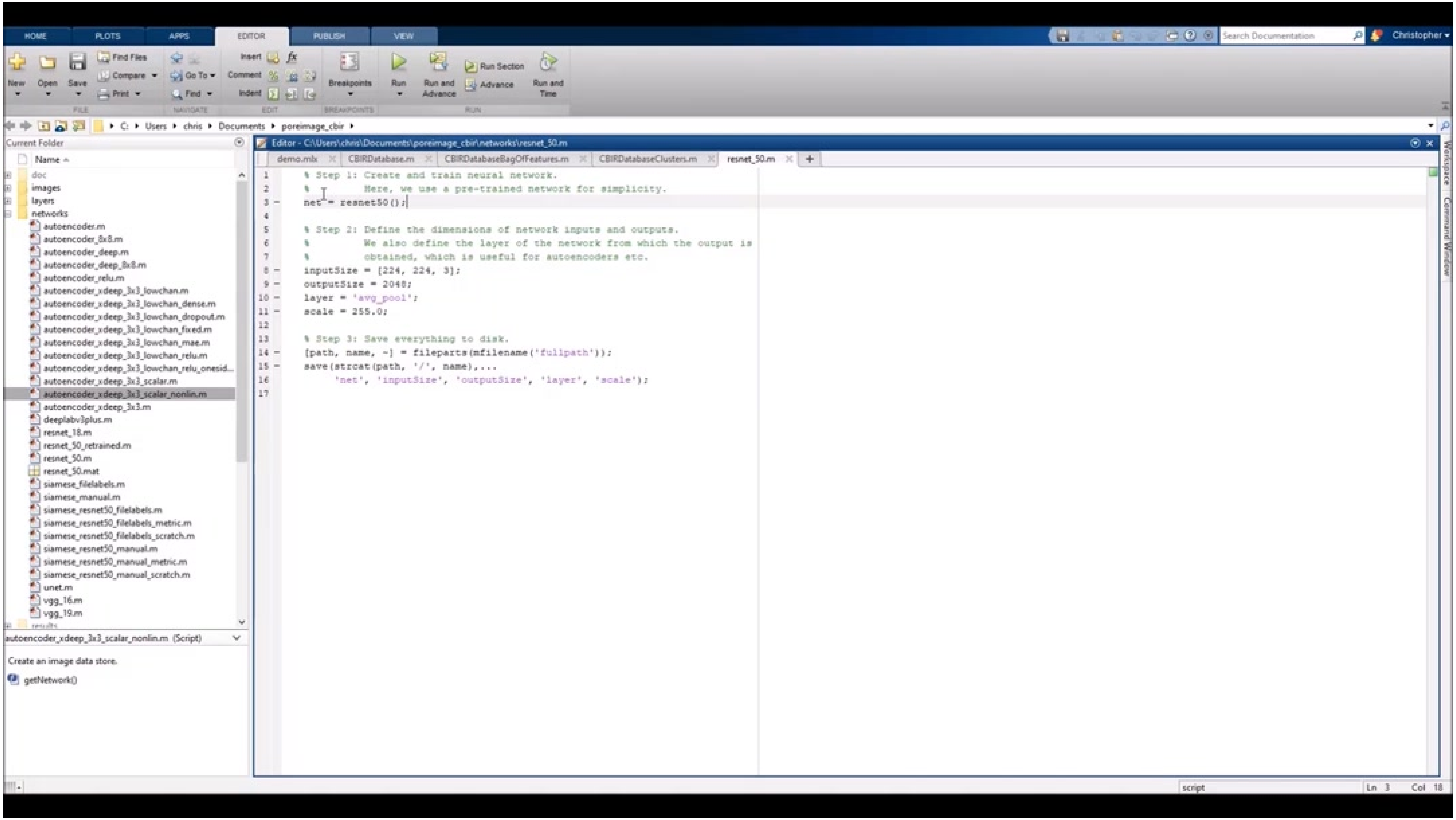
```matlab
% Modify read function if necessary.
resolution = [224, 224];
channels = 3;
scale = 255.0;
transforms = 1;
noise = 1.0e-2;
imds.ReadFcn = @(file) readMicroCTImage(file, ...
                                        resolution, channels, scale, transforms, noise);
```

```matlab
% Load a deep neural network.
net = CBIRNetwork('networks/resnet_50.mat');
```

```matlab
% Create the CBIR database.
k = 1;
db = CBIRDatabaseClusters(imds, net, k);
```

```matlab
% Load a query image.
queryID = 4;
img = readimage(imds_test, queryID);
imshow(img);
```

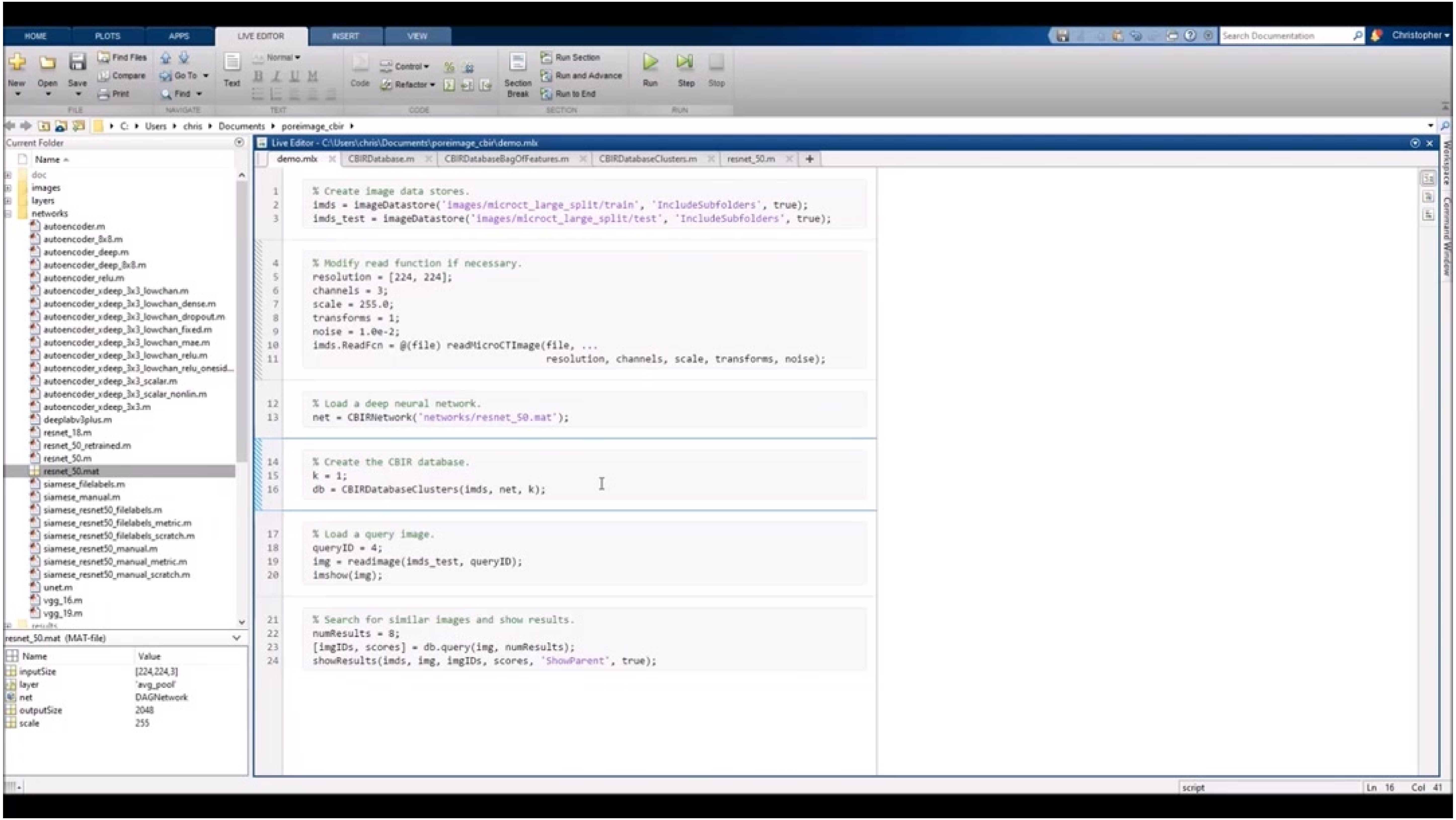
```matlab
% Search for similar images and show results.
numResults = 8;
[imgIDs, scores] = db.query(img, numResults);
showResults(imds, img, imgIDs, scores, 'ShowParent', true);
```


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Christopher
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
C: ▸ Users ▸ chris ▸ Documents ▸ poreimage_cbir ▸
Current Folder
Name
doc
images
layers
networks
autoencoder.m
autoencoder_8x8.m
autoencoder_deep.m
autoencoder_deep_8x8.m
autoencoder_relu.m
autoencoder_xdeep_3x3_lowchan.m
autoencoder_xdeep_3x3_lowchan_dense.m
autoencoder_xdeep_3x3_lowchan_dropout.m
autoencoder_xdeep_3x3_lowchan_fixed.m
autoencoder_xdeep_3x3_lowchan_mae.m
autoencoder_xdeep_3x3_lowchan_relu.m
autoencoder_xdeep_3x3_lowchan_relu_onesid...
autoencoder_xdeep_3x3_scalar.m
autoencoder_xdeep_3x3_scalar_nonlin.m
autoencoder_xdeep_3x3.m
deeplabv3plus.m
resnet_18.m
resnet_50_retrained.m
resnet_50.m
resnet_50.mat
siamese_filelabels.m
siamese_manual.m
siamese_resnet50_filelabels.m
siamese_resnet50_filelabels_metric.m
siamese_resnet50_filelabels_scratch.m
siamese_resnet50_manual.m
siamese_resnet50_manual_metric.m
siamese_resnet50_manual_scratch.m
unet.m
vgg_16.m
vgg_19.m
results
autoencoder_xdeep_3x3_scalar_nonlin.m (Script)
getNetwork()
demo.mlx
CBIRDatabase.m
CBIRDatabaseBagOfFeatures.m
CBIRDatabaseClusters.m
resnet_50.m
Workspace
Command Window
script
Ln 3
Col 18

Editor - C:\Users\chris\Documents\poreimage_cbir\networks\resnet_50.m

```
% Step 1: Create and train neural network.
%         Here, we use a pre-trained network for simplicity.
net = resnet50();

% Step 2: Define the dimensions of network inputs and outputs.
%         We also define the layer of the network from which the output is
%         obtained, which is useful for autoencoders etc.
inputSize = [224, 224, 3];
outputSize = 2048;
layer = 'avg_pool';
scale = 255.0;

% Step 3: Save everything to disk.
[path, name, ~] = fileparts(mfilename('fullpath'));
save(strcat(path, '/', name),...
    'net', 'inputSize', 'outputSize', 'layer', 'scale');
```

Create an image data store.

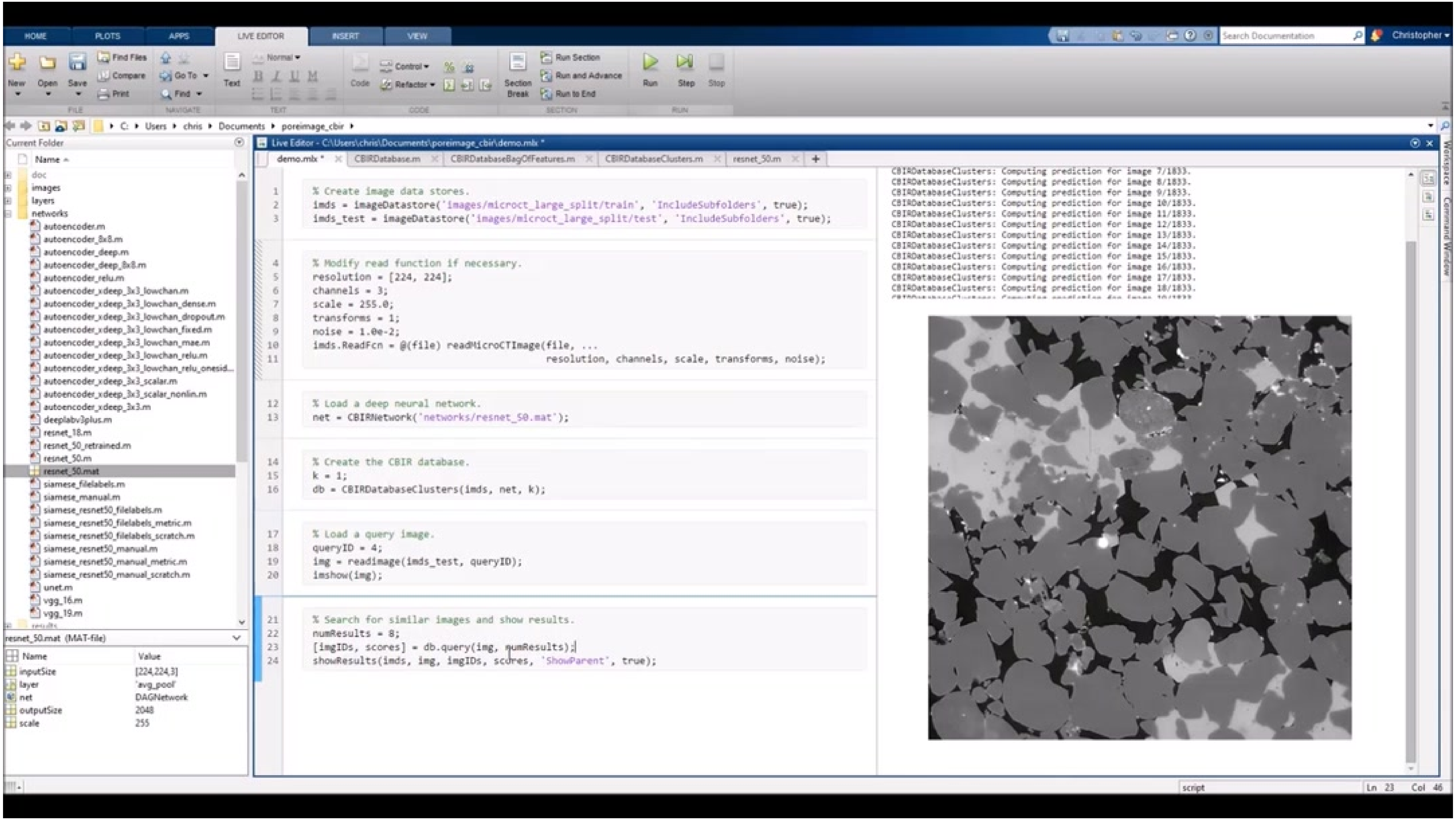


doc
images
layers
networks
autoencoder.m
autoencoder_8x8.m
autoencoder_deep.m
autoencoder_deep_8x8.m
autoencoder_relu.m
autoencoder_xdeep_3x3_lowchan.m
autoencoder_xdeep_3x3_lowchan_dense.m
autoencoder_xdeep_3x3_lowchan_dropout.m
autoencoder_xdeep_3x3_lowchan_fixed.m
autoencoder_xdeep_3x3_lowchan_mae.m
autoencoder_xdeep_3x3_lowchan_relu.m
autoencoder_xdeep_3x3_lowchan_relu_onesid...
autoencoder_xdeep_3x3_scalar.m
autoencoder_xdeep_3x3_scalar_nonlin.m
autoencoder_xdeep_3x3.m
deeplabv3plus.m
resnet_18.m
resnet_50_retrained.m
resnet_50.m
resnet_50.mat
siamese_filelabels.m
siamese_manual.m
siamese_resnet50_filelabels.m
siamese_resnet50_filelabels_metric.m
siamese_resnet50_filelabels_scratch.m
siamese_resnet50_manual.m
siamese_resnet50_manual_metric.m
siamese_resnet50_manual_scratch.m
unet.m
vgg_16.m
vgg_19.m
results

resnet_50.mat (MAT-file)

Name	Value
inputSize	[224,224,3]
layer	'avg_pool'
net	DAGNetwork
outputSize	2048
scale	255

```
% Create image data stores.
imds = imageDatastore('images/microct_large_split/train', 'IncludeSubfolders', true);
imds_test = imageDatastore('images/microct_large_split/test', 'IncludeSubfolders', true);

% Modify read function if necessary.
resolution = [224, 224];
channels = 3;
scale = 255.0;
transforms = 1;
noise = 1.0e-2;
imds.ReadFcn = @(file) readMicroCTImage(file, ...
                                        resolution, channels, scale, transforms, noise);

% Load a deep neural network.
net = CBIRNetwork('networks/resnet_50.mat');

% Create the CBIR database.
k = 1;
db = CBIRDatabaseClusters(imds, net, k);

% Load a query image.
queryID = 4;
img = readimage(imds_test, queryID);
imshow(img);

% Search for similar images and show results.
numResults = 8;
[imgIDs, scores] = db.query(img, numResults);
showResults(imds, img, imgIDs, scores, 'ShowParent', true);
```



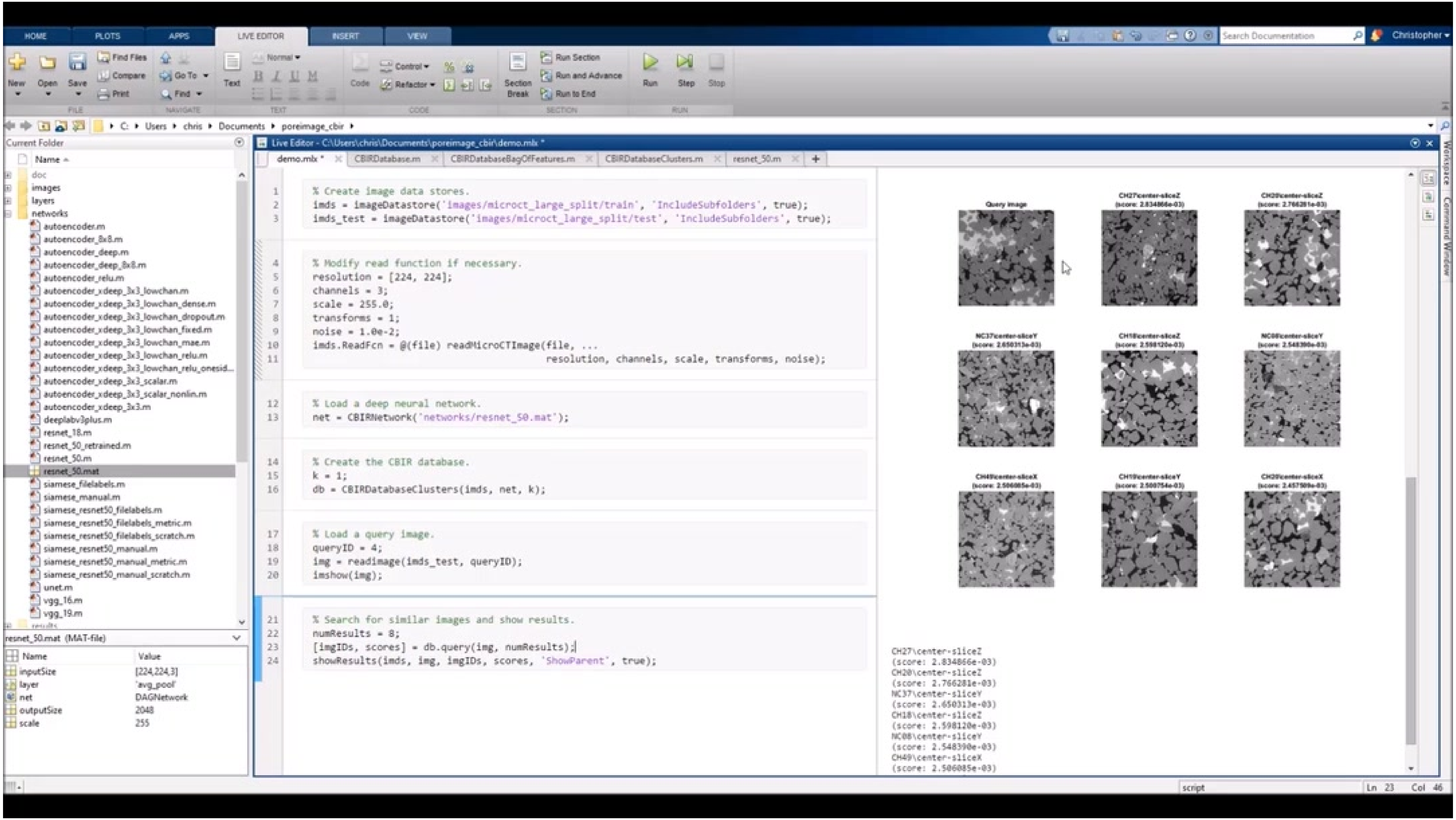
doc
images
layers
networks
autoencoder.m
autoencoder_8x8.m
autoencoder_deep.m
autoencoder_deep_8x8.m
autoencoder_relu.m
autoencoder_xdeep_3x3_lowchan.m
autoencoder_xdeep_3x3_lowchan_dense.m
autoencoder_xdeep_3x3_lowchan_dropout.m
autoencoder_xdeep_3x3_lowchan_fixed.m
autoencoder_xdeep_3x3_lowchan_mae.m
autoencoder_xdeep_3x3_lowchan_relu.m
autoencoder_xdeep_3x3_lowchan_relu_onesid...
autoencoder_xdeep_3x3_scalar.m
autoencoder_xdeep_3x3_scalar_nonlin.m
autoencoder_xdeep_3x3.m
deeplabv3plus.m
resnet_18.m
resnet_50_retrained.m
resnet_50.m
resnet_50.mat
siamese_filelabels.m
siamese_manual.m
siamese_resnet50_filelabels.m
siamese_resnet50_filelabels_metric.m
siamese_resnet50_filelabels_scratch.m
siamese_resnet50_manual.m
siamese_resnet50_manual_metric.m
siamese_resnet50_manual_scratch.m
unet.m
vgg_16.m
vgg_19.m
results

resnet_50.mat (MAT-file)

Name	Value
inputSize	[224,224,3]
layer	'avg_pool'
net	DAGNetwork
outputSize	2048
scale	255

```
% Create image data stores.
imds = imageDatastore('images/microct_large_split/train', 'IncludeSubfolders', true);
imds_test = imageDatastore('images/microct_large_split/test', 'IncludeSubfolders', true);

% Modify read function if necessary.
resolution = [224, 224];
channels = 3;
scale = 255.0;
transforms = 1;
noise = 1.0e-2;
imds.ReadFcn = @(file) readMicroCTImage(file, ...
                                        resolution, channels, scale, transforms, noise);

% Load a deep neural network.
net = CBIRNetwork('networks/resnet_50.mat');

% Create the CBIR database.
k = 1;
db = CBIRDatabaseClusters(imds, net, k);

% Load a query image.
queryID = 4;
img = readimage(imds_test, queryID);
imshow(img);

% Search for similar images and show results.
numResults = 8;
[imgIDs, scores] = db.query(img, numResults);
showResults(imds, img, imgIDs, scores, 'ShowParent', true);
```

```
CBIRDatabaseClusters: Computing prediction for image 7/1833.
CBIRDatabaseClusters: Computing prediction for image 8/1833.
CBIRDatabaseClusters: Computing prediction for image 9/1833.
CBIRDatabaseClusters: Computing prediction for image 10/1833.
CBIRDatabaseClusters: Computing prediction for image 11/1833.
CBIRDatabaseClusters: Computing prediction for image 12/1833.
CBIRDatabaseClusters: Computing prediction for image 13/1833.
CBIRDatabaseClusters: Computing prediction for image 14/1833.
CBIRDatabaseClusters: Computing prediction for image 15/1833.
CBIRDatabaseClusters: Computing prediction for image 16/1833.
CBIRDatabaseClusters: Computing prediction for image 17/1833.
CBIRDatabaseClusters: Computing prediction for image 18/1833.
```



C: ▸ Users ▸ chris ▸ Documents ▸ poreimage_cbir ▸

Live Editor - C:\Users\chris\Documents\poreimage_cbir\demo.mlx *

demo.mlx * | CBIRDatabase.m | CBIRDatabaseBagOfFeatures.m | CBIRDatabaseClusters.m | resnet_50.m

doc
images
layers
networks
autoencoder.m
autoencoder_8x8.m
autoencoder_deep.m
autoencoder_deep_8x8.m
autoencoder_relu.m
autoencoder_xdeep_3x3_lowchan.m
autoencoder_xdeep_3x3_lowchan_dense.m
autoencoder_xdeep_3x3_lowchan_dropout.m
autoencoder_xdeep_3x3_lowchan_fixed.m
autoencoder_xdeep_3x3_lowchan_mae.m
autoencoder_xdeep_3x3_lowchan_relu.m
autoencoder_xdeep_3x3_lowchan_relu_onesid...
autoencoder_xdeep_3x3_scalar.m
autoencoder_xdeep_3x3_scalar_nonlin.m
autoencoder_xdeep_3x3.m
deeplabv3plus.m
resnet_18.m
resnet_50_retrained.m
resnet_50.m
resnet_50.mat
siamese_filelabels.m
siamese_manual.m
siamese_resnet50_filelabels.m
siamese_resnet50_filelabels_metric.m
siamese_resnet50_filelabels_scratch.m
siamese_resnet50_manual.m
siamese_resnet50_manual_metric.m
siamese_resnet50_manual_scratch.m
unet.m
vgg_16.m
vgg_19.m
results

resnet_50.mat (MAT-file)

Name	Value
inputSize	[224,224,3]
layer	'avg_pool'
net	DAGNetwork
outputSize	2048
scale	255

```
% Create image data stores.
imds = imageDatastore('images/microct_large_split/train', 'IncludeSubfolders', true);
imds_test = imageDatastore('images/microct_large_split/test', 'IncludeSubfolders', true);

% Modify read function if necessary.
resolution = [224, 224];
channels = 3;
scale = 255.0;
transforms = 1;
noise = 1.0e-2;
imds.ReadFcn = @(file) readMicroCTImage(file, ...
                                         resolution, channels, scale, transforms, noise);

% Load a deep neural network.
net = CBIRNetwork('networks/resnet_50.mat');

% Create the CBIR database.
k = 1;
db = CBIRDatabaseClusters(imds, net, k);

% Load a query image.
queryID = 4;
img = readimage(imds_test, queryID);
imshow(img);

% Search for similar images and show results.
numResults = 8;
[imgIDs, scores] = db.query(img, numResults);
showResults(imds, img, imgIDs, scores, 'ShowParent', true);
```

```
CH27\center-sliceZ
(score: 2.834866e-03)
CH20\center-sliceZ
(score: 2.766281e-03)
NC37\center-sliceY
(score: 2.650313e-03)
CH18\center-sliceZ
(score: 2.598120e-03)
NC08\center-sliceY
(score: 2.548390e-03)
CH49\center-sliceX
(score: 2.506085e-03)
```