Homework 4

Shao-Ting Chiu (UIN:433002162)

11/15/22

Table of contents

Homework Description

- Course: ECEN649, Fall2022
- Deadline: 2022/11/16, 11:59 pm > Problems from the Book > > 6.3 > > 6.5 > > 6.7 > > 7.1 > > 7.10 > > 6.12 (coding assignment) > > Problems 6.3-6.5 are worth 10 points each, Problem 7.10 and the coding assignment are worth 20 points each.

Computational Environment

Libraries

```python
import numpy as np
import tensorflow as tf
import sys
```

Versions

```python
print(np.__version__)
print(tf.__version__)
print (sys.version)
print(sys.executable)
```

```
1.23.4
2.10.0
3.9.12 (main, Apr  5 2022, 01:52:34)
[Clang 12.0.0 ]
```

/Users/stevenchiu/miniconda/bin/python

Problem 6.3

> Show that the decision regions produced by a neural network with k threshold sigmoids in the *first* hidden layer, no matter what nonlinearities are used in succeeding layers, are equal to the intersection of k half-spaces, i.e., the decision boundary is piecewise linear
>
> Hint: All neurons in the first hidden layer are perceptrons and the output of the layer is a binary vector.

Let $\bar{O}$ be the k output of first hidden layer, and there are 2^k types of binary vectors $[O_1, \ldots, O_k]$.

For each data point $x \in R^d$ where d is the feature space. the output of first layer is

$$O(x)_i = I_{g_i(x)}(x), \quad i = 1, \cdots, k \tag{1}$$

where $g_i(\cdot)$ is the perceptron function of neuron i. Thus, any point x belong to one type of $[I_{g_1(x)}(x), \ldots, I_{g_k(x)}(x)]$. For each O_i, the space forms a half-space with $\{x : g_i(x) > 0\}$, and there are k half space in total.

Problem 6.5

> For the VGG16 CNN architecture:

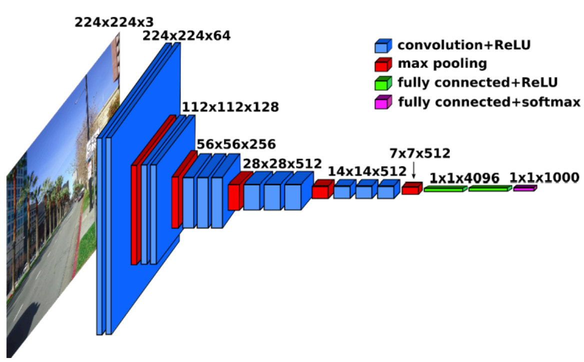


Figure 1: VGG16

(a)

Determine the number of filters used in each convolution layer.

- `Conv-1`: 64 filters (pre-depth: 3)
- `Conv-2`: 128 filters (pre-depth: 64)
- `Conv-3`: 256 filters (pre-depth: 128)
- `Conv-4`: 512 filters (pre-depth: 256)
- `Conv-5`: 512 filters (pre-depth: 512)

There are total

```
rs = np.array([3, 64, 128, 256, 512])
t_filters = np.array([64, 128, 256, 512, 512])
np.sum(t_filters)
```

```
1472
```

filters.

(b)

Based on the fact that all filters are of size $3 \times 3 \times r$, where r is the depth of the previous layer, determine the total number of convolution weights in the entire network.

```
CONV1 = (3*3*3)*64 + (3*3*64)*64
CONV1
```

```
38592
```

```
CONV2 = (3*3*64)*128 + (3*3*128)*128
CONV2
```

```
221184
```

```
CONV3 = (3*3*128)*256 + (3*3*256)*256 + (3*3*256)*256
CONV3
```

```
1474560
```

```
CONV4 = (3*3*256)*512 + (3*3*512)*512 + (3*3*512)*512
CONV4
```

```
5898240
```

```
CONV5 = (3*3*512)*512 *3
CONV5
```

```
7077888
```

```
fc1 = 512 * 7 * 7 * 4096
fc1
```

```
102760448
```

```
fc2 = 4096 * 4096
fc2
```

```
16777216
```

```
fc3 = 4096 * 1000
fc3
```

```
4096000
```

(c)

> Add the weights used in the fully-connected layers to obtain the total number of weights used by VGG16.

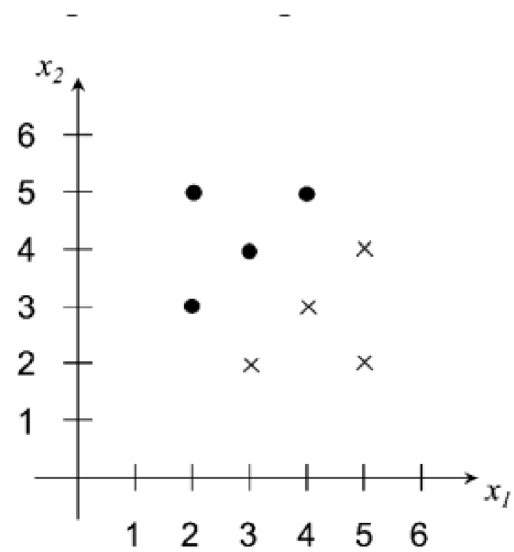
Total of weights

```
total = np.sum([CONV1, CONV2, CONV3, CONV4, CONV5, fc1, fc2, fc3])
total
```

```
138344128
```

Problem 6.7

Consider the training data set given in the figure below.



(a)

By inspection, find the coefficients of the linear SVM hyperplane $a_1 x_1 + a_2 x_2 + a_0 = 0$ and plot it. What is the value of the margin? Say as much as you can about the values of the Lagrange multipliers associated with each of the points.

The boundary passes by $\frac{1}{2}((3,3)+(3,2)) = (3,2.5)$ and $\frac{1}{2}((3,4)+(4,3)) = (3.5,3.5)$

- $a_1 = 2.5 - 3.5 = -1$
- $a_2 = 3.5 - 3 = 0.5$
- $a_0 = 3 \cdot 3.5 - 3.5 \cdot 2.5 = 1.75$
- The boundary is

$$-x_1 + 0.5x_2 + 1.75 = 0$$

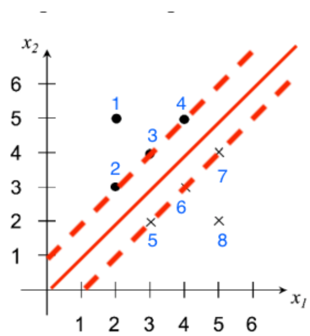


Figure 2: SVM boundry

In Figure 2, there are 6 support vectors that are λ_2 to λ_7. The KKT conditions[1] state that

$$\lambda_i = 0 \Rightarrow y_i E_i \leq 0 \tag{2}$$
$$0 < \lambda_i < C \Rightarrow y_i E_i = 0 \tag{3}$$
$$\lambda_i = C \Rightarrow y_i E_i \geq 0 \tag{4}$$

- Lagrange multipliers
  - $\lambda_1 = 0$
  - $\lambda_2 \in (0, C)$
  - $\lambda_3 \in (0, C)$
  - $\lambda_4 \in (0, C)$
  - $\lambda_5 \in (0, C)$
  - $\lambda_6 \in (0, C)$
  - $\lambda_7 \in (0, C)$
  - $\lambda_8 = 0$

where C is the pentalty term.

(b)

> Apply the CART rule, using the misclassification impurity, and stop after finding one splitting node (this is the "1R" or "stump" rule). If ther eis a tie between best splits, pick one that makes at most one error in each class. Plot this classifier as a decision boundary superimposed on the training data and also as a binary decision tree showing the splitting and leaf nodes.

where $\bullet$ labelled as 1; $\circ$ labelled as 0.

[1]Intro. to SVM: https://article.sciencepublishinggroup.com/html/10.11648.j.acm.s.2017060401.11.html

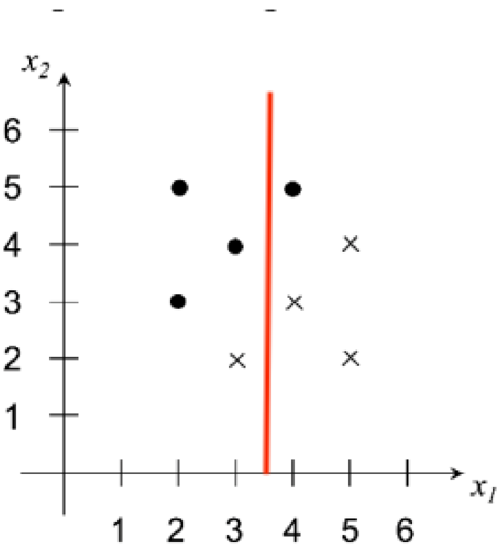


Figure 3: Decision boundary

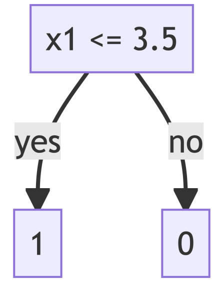


Figure 4: Apply CART rule

(c)

How do you compare the classifiers in (a) and (b) ? Which one is more likely to have a smaller classification error in this problem?

- SVM of (a) yields smaller classification error than (b) because it allow any slope of decision boundary.

Problem 7.1

Suppose that the classification error ϵ_n and an error estimator $\hat{\epsilon}_n$ are jointly Gaussian, such

$$\epsilon_n \sim N(\epsilon^* + \frac{1}{n}, \frac{1}{n^2}), \hat{\epsilon}_n \sim N(\epsilon^* - \frac{1}{n}, \frac{1}{n^2}), Cov(\epsilon_n, \hat{\epsilon}_n) = \frac{1}{2n^2}$$

where ϵ^* is the Bayes error. Find the bias, deviation variance, RMS, correlation coefficient and tail probabilities $P(\hat{\epsilon}_n - \epsilon_n < -\tau)$ and $P(\hat{\epsilon}_n - \epsilon_n > \tau)$ of $\hat{\epsilon}_n$. Is this estimator optimistically or pessimistically biased? Does performance improve as sample size increases? Is the estimator consistent?

Bias

Use Eq. 7.3 (Braga-Neto 2020, 154),

$$Bias(\hat{\epsilon}_n) = E[\hat{\epsilon}_n] - E[\epsilon_n]$$

- $E[\hat{\epsilon}_n] = \epsilon^* - \frac{1}{n}$
- $E[\epsilon_n] = \epsilon^* + \frac{1}{n}$

Thus,

$$Bias(\hat{\epsilon}_n) = \frac{-2}{n} < 0$$

This estimator is *optimisitcally biased*.

Deviation variance

Use Eq. 7.4 (Braga-Neto 2020, 154),

$$Var_{dev}(\hat{\epsilon}_n) = Var(\hat{\epsilon}_n, \epsilon_n) = Var(\hat{\epsilon}_n) + Var(\epsilon_n) - 2Cov(\epsilon_n, \hat{\epsilon}_n)$$

- $Var(\hat{\epsilon}_n) = \frac{1}{n^2}$
- $Var(\epsilon_n) = \frac{1}{n^2}$
- $Cov(\epsilon_n, \hat{\epsilon}_n) = \frac{1}{2n^2}$

Thus,

$$Var_{dev}(\hat{\epsilon}_n) = \frac{1}{n^2} + \frac{1}{n^2} - 2\frac{1}{2n^2} = \frac{1}{n^2}$$

The deviation variance reduces as sample size increases.

Root mean-square error

Use Eq. 7.5 (Braga-Neto 2020, 154),

$$RMS(\hat{\epsilon}_n) = \sqrt{E[(\hat{\epsilon}_n - \epsilon_n)^2]} = \sqrt{Bias(\hat{\epsilon}_n)^2 + Var_{dev}(\hat{\epsilon}_n)}$$

Apply previous results,

$$RMS(\hat{\epsilon}_n) = \sqrt{\frac{4}{n^2} + \frac{1}{n^2}} = \frac{\sqrt{5}}{n}$$

Correlation coefficient

Use the pearson correlation coefficient[2]

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

- $Cov(\epsilon_n, \hat{\epsilon}_n) = \frac{1}{2n^2}$
- $\sigma_{\epsilon_n} = \frac{1}{n}$
- $\sigma_{\hat{\epsilon}_n} = \frac{1}{n}$

$$\rho_{\epsilon_n, \hat{\epsilon}_n} = \frac{1}{2}$$

Correlation coefficient is a constant and independent from sample size.

Tail probabilities

Use Eq. 7.6 (Braga-Neto 2020, 154),

$$P(|\hat{\epsilon}_n - \epsilon_n| \geq \tau) = P(\hat{\epsilon}_n - \epsilon_n \geq \tau) + P(\hat{\epsilon}_n - \epsilon_n \leq -\tau), \quad \text{for } \tau > 0$$

The normal difference distribution[3] of $\hat{\epsilon}_n - \epsilon_n$

$$\hat{\epsilon}_n - \epsilon_n \sim N(\frac{-2}{n}, \frac{2}{n^2}) = N(\mu, \sigma^2)$$

[2]Correlation coefficient: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[3]Normal difference distribution: https://mathworld.wolfram.com/NormalDifferenceDistribution.html

That $\Delta\epsilon_n = \hat{\epsilon}_n - \epsilon_n$

$$\begin{aligned}
P(\Delta\epsilon_n \leq -\tau) &= P(\frac{\Delta\epsilon_n - \mu}{\sigma} \leq \frac{-\tau - \mu}{\sigma}) && (5)\\
&= \Phi(\frac{-\tau - \mu}{\sigma}) && (6)\\
&= \Phi(\frac{-\tau + 2/n}{\sqrt{2}/n}) && (7)\\
&= \Phi(\frac{-n\tau + 2}{\sqrt{2}}) && (8)\\
& && (9)
\end{aligned}$$

$$\begin{aligned}
P(\Delta\epsilon_n \geq \tau) &= P(\frac{\Delta\epsilon_n - \mu}{\sigma} \geq \frac{\tau - \mu}{\sigma}) && (10)\\
&= 1 - P(\frac{\Delta\epsilon_n - \mu}{\sigma} < \frac{\tau - \mu}{\sigma}) && (11)\\
&= 1 - \Phi(\frac{\tau - \mu}{\sigma}) && (12)\\
&= 1 - \Phi(\frac{n\tau - 2}{1}) && (13)
\end{aligned}$$

Thus, when $n \to \infty$

$$\begin{aligned}
\lim_{n\to\infty} P(\Delta\epsilon_n \leq -\tau) = 0 && (14)\\
\lim_{n\to\infty} P(\Delta\epsilon_n \geq \tau) = 0 && (15)
\end{aligned}$$

This can be conluced to

$$\lim_{n\to\infty} P(|\hat{\epsilon}_n - \epsilon_n| \geq \tau) = 0$$

The estimator is *consistent*.

Problem 7.10

This problem illustrates the very poor (even paradoxical) performance of cross-validation with very small sample sizes. Consider the resubstitution and leave-one-out estimators $\hat{\epsilon}_n^r$ and $\hat{\epsilon}_n^l$ for the 3NN classification rule, with a sample of size $n = 4$ from a mixture of two equally-likely Gaussian populations $\Pi_0 \sim N_d(\mu_0, \Sigma)$

and $\Pi_1 \sim N_d(\mu_1, \Sigma)$. Assume that μ_0 and μ_1 are far enough apart to make $\delta = \sqrt{(\mu_1 - \mu_0)^T \Sigma^{-1} (\mu_1 = \mu_0)} \gg 0$ (in which case the Bayes error is $\epsilon_{\text{bay}} = \Phi(-\frac{\delta}{2}) \approx 0$).

(a)

For a sample S_n with $N_0 = N_1 = 2$, which occurs $P(N_0 = 2) = \binom{4}{2} 2^{-4} = 37.5\%$ of the time, show that $\epsilon_n \approx 0$ but $\hat{\epsilon}_n^l = 1$

(b)

Show that $E[\epsilon_n] \approx \frac{5}{16} = 0.3125$, but $E[\hat{\epsilon}_n^l] = 0.5$, so that $\text{Bias}(\hat{\epsilon}_n^l) \approx \frac{3}{16} = 0.1875$, and the leave-one-out estimator is far from unbiased.

(c)

Show that $Var_d(\hat{\epsilon}_n^l) \approx \frac{103}{256} \approx 0.402$, which corresponds to a standard deviation of $\sqrt{0.402} = 0.634$. The leave-one-out estimator is therefore highly-biased and highly-variable in this case.

(d)

Consider the correlation coefficient of an error estimator $\hat{\epsilon}_n$ with the true error ϵ_n:

$$\rho(\epsilon_n, \hat{\epsilon}_n) = \frac{Cov(\epsilon_n, \hat{\epsilon}_n)}{Std(\epsilon_n) Std(\hat{\epsilon}_n)}$$

Show that $\rho(\epsilon_n, \hat{\epsilon}_n^l \approx 0.98)$, i.e., the leave-one-out estimator is almost perfectly negatively correlated with the true error.

(e)

For comparison, show that, although $E[\hat{\epsilon}_n^r] = \frac{1}{8} = 0.125$, so that $\text{Bias}(\hat{\epsilon}_n^r) \approx \frac{-3}{16} = -0.1875$, which is exactly the negative of the bias of leave-one-out, we have $Var_d(\hat{\epsilon}_n^r) \approx \frac{7}{256} \approx 0.027$, for a standard deviation of $\frac{\sqrt{7}}{16} \approx 0.165$, which is several times smaller than the leave-one-out variance, and $\rho(\epsilon_n, \hat{\epsilon}_n^r) \approx \sqrt{\frac{3}{5}} \approx 0.775$, showing that the resubstitution estimator is highly positively correlated with the true error.

Problem 6.12

```python
import tensorflow as tf
import numpy as np
import PIL
import cv2
import os
import sklearn
import pandas as pd
import pickle
import platform
from tqdm.notebook import tqdm
from sklearn.multiclass import OneVsOneClassifier
from sklearn import preprocessing
from sklearn import svm
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from scipy import stats as st
```

Computational Environment

```python
physical_devices = tf.config.list_physical_devices('GPU')
my_system = platform.uname()
print(physical_devices)
print(f"System: {my_system.system}")
print(f"Node Name: {my_system.node}")
print(f"Release: {my_system.release}")
print(f"Version: {my_system.version}")
print(f"Machine: {my_system.machine}")
print(f"Processor: {my_system.processor}")
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
System: Darwin
Node Name: qiushaotings-MacBook-Pro-2.local
Release: 21.5.0
Version: Darwin Kernel Version 21.5.0: Tue Apr 26 21:08:29 PDT 2022; root:xnu-8020.121.3~4/R
Machine: arm64
Processor: i386
```

Helper function

```
def load_image(path, width=484, preprocess_input=tf.keras.applications.vgg16.preprocess_in
    """
    Load and Preprocessing image
    """
    img = tf.keras.utils.load_img(path)
    x = tf.keras.utils.img_to_array(img)
    x = x[0:width,:,:]
    x = np.expand_dims(x, axis=0)
    return tf.keras.applications.vgg16.preprocess_input(x)
```

Data inspection

```
dpath = os.path.join("data", "CMU-UHCS_Dataset")
pic_path = os.path.join(dpath, "images")
df_micro = pd.read_csv( os.path.join(dpath, "micrograph.csv"))
df_micro = df_micro[["path", "primary_microconstituent"]]

## Save paths
paths =  dict(zip(["train", "test"],\
        [os.path.join(dpath, "feature_{}.pkl".format(n))\
         for n in ["train", "test"]]))

for i in range(0, len(df_micro)):
    img_ph = os.path.join(pic_path,df_micro.iloc[i][0])
    assert os.path.exists(img_ph)
    df_micro.iloc[i][0] = img_ph
df_micro2 = df_micro.copy()
CLS_rm = ["pearlite+widmanstatten", "martensite", "pearlite+spheroidite"] #(type, sample s
```

```
for c in CLS_rm:
    df_micro.drop(df_micro[df_micro["primary_microconstituent"] == c].index, inplace=True)
```

```
# labels
name_lbs = df_micro["primary_microconstituent"].unique()
le = preprocessing.LabelEncoder()
le.fit(name_lbs)
```

```python
list(le.classes_)
```

```
['network', 'pearlite', 'spheroidite', 'spheroidite+widmanstatten']
```

```python
dlabel = le.transform(df_micro["primary_microconstituent"])
df_micro.insert(2, "label", dlabel)
df_micro
```

```
/Users/stevenchiu/miniconda/lib/python3.9/site-packages/IPython/core/formatters.py:342: Futu
  return method()
```

	path	primary_microconstituent	label
0	data/CMU-UHCS_Dataset/images/micrograph1.tif	pearlite	1
1	data/CMU-UHCS_Dataset/images/micrograph2.tif	spheroidite	2
3	data/CMU-UHCS_Dataset/images/micrograph5.tif	pearlite	1
4	data/CMU-UHCS_Dataset/images/micrograph6.tif	spheroidite	2
5	data/CMU-UHCS_Dataset/images/micrograph7.tif	spheroidite+widmanstatten	3
6	data/CMU-UHCS_Dataset/images/micrograph8.tif	network	0
7	data/CMU-UHCS_Dataset/images/micrograph9.tif	network	0
8	data/CMU-UHCS_Dataset/images/micrograph10.png	spheroidite	2
9	data/CMU-UHCS_Dataset/images/micrograph11.tif	spheroidite	2
11	data/CMU-UHCS_Dataset/images/micrograph13.tif	network	0
12	data/CMU-UHCS_Dataset/images/micrograph14.tif	network	0
14	data/CMU-UHCS_Dataset/images/micrograph18.png	network	0
16	data/CMU-UHCS_Dataset/images/micrograph21.tif	network	0
17	data/CMU-UHCS_Dataset/images/micrograph24.tif	network	0
18	data/CMU-UHCS_Dataset/images/micrograph26.tif	spheroidite+widmanstatten	3
19	data/CMU-UHCS_Dataset/images/micrograph28.tif	network	0
20	data/CMU-UHCS_Dataset/images/micrograph29.tif	spheroidite	2
21	data/CMU-UHCS_Dataset/images/micrograph30.tif	spheroidite	2
22	data/CMU-UHCS_Dataset/images/micrograph31.tif	network	0
23	data/CMU-UHCS_Dataset/images/micrograph32.tif	spheroidite	2
24	data/CMU-UHCS_Dataset/images/micrograph33.tif	network	0
25	data/CMU-UHCS_Dataset/images/micrograph35.tif	network	0
26	data/CMU-UHCS_Dataset/images/micrograph36.tif	network	0
27	data/CMU-UHCS_Dataset/images/micrograph37.tif	spheroidite+widmanstatten	3
28	data/CMU-UHCS_Dataset/images/micrograph39.tif	spheroidite	2
30	data/CMU-UHCS_Dataset/images/micrograph42.tif	network	0
32	data/CMU-UHCS_Dataset/images/micrograph46.tif	network	0
33	data/CMU-UHCS_Dataset/images/micrograph47.tif	network	0
34	data/CMU-UHCS_Dataset/images/micrograph48.tif	spheroidite	2
35	data/CMU-UHCS_Dataset/images/micrograph49.tif	spheroidite	2
36	data/CMU-UHCS_Dataset/images/micrograph51.tif	network	0
37	data/CMU-UHCS_Dataset/images/micrograph52.tif	pearlite	1
38	data/CMU-UHCS_Dataset/images/micrograph53.tif	spheroidite	2
39	data/CMU-UHCS_Dataset/images/micrograph58.tif	spheroidite+widmanstatten	3
40	data/CMU-UHCS_Dataset/images/micrograph59.tif	pearlite	1
41	data/CMU-UHCS_Dataset/images/micrograph60.tif	network	0
43	data/CMU-UHCS_Dataset/images/micrograph65.tif	network	0
44	data/CMU-UHCS_Dataset/images/micrograph67.tif	spheroidite	2
45	data/CMU-UHCS_Dataset/images/micrograph68.tif	spheroidite	2
46	data/CMU-UHCS_Dataset/images/micrograph69.tif	spheroidite	2
47	data/CMU-UHCS_Dataset/images/micrograph70.tif	network	0
49	data/CMU-UHCS_Dataset/images/micrograph72.tif	network	0
50	data/CMU-UHCS_Dataset/images/micrograph74.tif	pearlite	1
51	data/CMU-UHCS_Dataset/images/micrograph75.tif	spheroidite	2
52	data/CMU-UHCS_Dataset/images/micrograph76.tif	spheroidite	2
53	data/CMU-UHCS_Dataset/images/micrograph80.tif	network	0
54	data/CMU-UHCS_Dataset/images/micrograph82.tif	spheroidite	2
55	data/CMU-UHCS_Dataset/images/micrograph86.tif	network	0
56	data/CMU-UHCS_Dataset/images/micrograph88.tif	network	0
57	data/CMU-UHCS_Dataset/images/micrograph89.tif	network	0
58	data/CMU-UHCS_Dataset/images/micrograph90.tif	spheroidite	2
59	data/CMU-UHCS_Dataset/images/micrograph91.tif	pearlite	1

Data Processing

```
# Train-test split
df_test = df_micro.copy()
df_train = pd.DataFrame(columns = df_micro.keys())

split_info = [("spheroidite", 100),\
              ("network", 100),\
              ("pearlite", 100),\
              ("spheroidite+widmanstatten", 60)] #(type, sample size)



for ln in split_info:
    label, n = ln
    id_train = df_micro[df_micro["primary_microconstituent"] == label][0:n].index
    df_test.drop(id_train, axis=0, inplace=True)
    df_train = pd.concat([df_train, df_micro.loc[id_train]])
```

```
print(df_train.to_string())
```

```
print(df_test.to_string())
```

Feature Extraction

```
# VGG16

base_model = tf.keras.applications.vgg16.VGG16(
    include_top=False,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation='softmax'
)

base_model.summary()
```

```
Model: "vgg16"

_________________________________________________________________

 Layer (type)                Output Shape              Param #

=================================================================

 input_1 (InputLayer)        [(None, None, None, 3)]   0



 block1_conv1 (Conv2D)       (None, None, None, 64)    1792



 block1_conv2 (Conv2D)       (None, None, None, 64)    36928



 block1_pool (MaxPooling2D)  (None, None, None, 64)    0



 block2_conv1 (Conv2D)       (None, None, None, 128)   73856



 block2_conv2 (Conv2D)       (None, None, None, 128)   147584



 block2_pool (MaxPooling2D)  (None, None, None, 128)   0



 block3_conv1 (Conv2D)       (None, None, None, 256)   295168
```

```
block3_conv2 (Conv2D)       (None, None, None, 256)   590080

block3_conv3 (Conv2D)       (None, None, None, 256)   590080

block3_pool (MaxPooling2D)  (None, None, None, 256)   0

block4_conv1 (Conv2D)       (None, None, None, 512)   1180160

block4_conv2 (Conv2D)       (None, None, None, 512)   2359808

block4_conv3 (Conv2D)       (None, None, None, 512)   2359808

block4_pool (MaxPooling2D)  (None, None, None, 512)   0

block5_conv1 (Conv2D)       (None, None, None, 512)   2359808

block5_conv2 (Conv2D)       (None, None, None, 512)   2359808
```

```
 block5_conv3 (Conv2D)       (None, None, None, 512)   2359808



 block5_pool (MaxPooling2D)  (None, None, None, 512)   0



=================================================================

Total params: 14,714,688

Trainable params: 14,714,688

Non-trainable params: 0

_________________________________________________________________
```

Use five layers

```
out_layer_ns = ["block{}_pool".format(i) for i in range(1,6)]
out_layer_ns
```

```
['block1_pool', 'block2_pool', 'block3_pool', 'block4_pool', 'block5_pool']
```

```
# Construct 5 models for feature extraction
extmodel = dict(zip(out_layer_ns, [tf.keras.Model(
    inputs= base_model.input,
    outputs=base_model.get_layer(bk_name).output
) for bk_name in out_layer_ns]))

extmodel
```

```
{'block1_pool': <keras.engine.functional.Functional at 0x16e96ba30>,
 'block2_pool': <keras.engine.functional.Functional at 0x17dc23f10>,
 'block3_pool': <keras.engine.functional.Functional at 0x17ce1a580>,
 'block4_pool': <keras.engine.functional.Functional at 0x16e96b400>,
 'block5_pool': <keras.engine.functional.Functional at 0x17ce2a400>}
```

```python
# Display output dimensions
out_shapes = [extmodel[m].output_shape[-1] for m in extmodel.keys()]
out_shapes
```

```
[64, 128, 256, 512, 512]
```

```python
# Initiate feature maps for testing and training
fs_train = [np.zeros((df_train.shape[0], n_f)) for n_f in out_shapes]
fs_test = [np.zeros((df_test.shape[0], n_f)) for n_f in out_shapes]

features_train = dict(zip(out_layer_ns, fs_train))
features_test = dict(zip(out_layer_ns, fs_test))

features_train
```

```
{'block1_pool': array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 'block2_pool': array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 'block3_pool': array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 'block4_pool': array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
```

```
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 'block5_pool': array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])}
```

```python
# Feature extraction with VGG16
if os.path.exists(os.path.join(dpath, "feature_train.pkl")) == False:
    for m in tqdm(extmodel.keys()):
        for i, df in enumerate([df_train, df_test]):
            for j, ph in tqdm(enumerate(df["path"])):
                x = load_image(ph)
                xb = extmodel[m].predict(x, verbose = 0) # silence output
                F = np.mean(xb,axis=(0,1,2))
                # Save features
                if i ==0:
                    features_train[m][j, :] = F
                else:
                    features_test[m][j, :] = F
    #save file
    paths =  dict(zip(["train", "test"],\
        [os.path.join(dpath, "feature_{}.pkl".format(n))\
         for n in ["train", "test"]]))
    ## Create new files
    f_train = open(paths["train"], "wb")
    f_test = open(paths["test"], "wb")
    ## Write
    pickle.dump(features_train, f_train)
    pickle.dump(features_test, f_test)
    ## Close files
    f_train.close()
    f_test.close()
```

SVM

```
# load data
ftn = open(paths["train"], "rb")
ftt = open(paths["test"], "rb")
featn = pickle.load(ftn) # train feature
featt = pickle.load(ftt) # test feature
ftn.close()
ftt.close()

# label
ltrain = df_train[["primary_microconstituent", "label"]].reset_index()
ltest = df_test[["primary_microconstituent", "label"]].reset_index()
```

```
ltrain
```

```
/Users/stevenchiu/miniconda/lib/python3.9/site-packages/IPython/core/formatters.py:342: Futu
  return method()
```

	index	primary_microconstituent	label
0	1	spheroidite	2
1	4	spheroidite	2
2	8	spheroidite	2
3	9	spheroidite	2
4	20	spheroidite	2
5	21	spheroidite	2
6	23	spheroidite	2
7	28	spheroidite	2
8	34	spheroidite	2
9	35	spheroidite	2
10	38	spheroidite	2
11	44	spheroidite	2
12	45	spheroidite	2
13	46	spheroidite	2
14	51	spheroidite	2
15	52	spheroidite	2
16	54	spheroidite	2
17	58	spheroidite	2
18	62	spheroidite	2
19	64	spheroidite	2
20	65	spheroidite	2
21	66	spheroidite	2
22	74	spheroidite	2
23	75	spheroidite	2
24	82	spheroidite	2
25	84	spheroidite	2
26	85	spheroidite	2
27	87	spheroidite	2
28	88	spheroidite	2
29	89	spheroidite	2
30	91	spheroidite	2
31	92	spheroidite	2
32	93	spheroidite	2
33	96	spheroidite	2
34	97	spheroidite	2
35	98	spheroidite	2
36	99	spheroidite	2
37	102	spheroidite	2
38	103	spheroidite	2
39	105	spheroidite	2
40	106	spheroidite	2
41	111	spheroidite	2
42	114	spheroidite	2
43	115	spheroidite	2
44	119	spheroidite	2
45	121	spheroidite	2
46	122	spheroidite	2
47	123	spheroidite	2
48	128	spheroidite	2
49	131	spheroidite	2
50	134	spheroidite	2
51	136	spheroidite	2

```
ltest
```

```
/Users/stevenchiu/miniconda/lib/python3.9/site-packages/IPython/core/formatters.py:342: Futu
  return method()
```

	index	primary_microconstituent	label
0	237	spheroidite	2
1	238	spheroidite	2
2	239	spheroidite	2
3	241	spheroidite	2
4	242	spheroidite	2
5	244	spheroidite	2
6	245	spheroidite	2
7	246	spheroidite	2
8	249	spheroidite	2
9	250	spheroidite	2
10	252	spheroidite	2
11	253	spheroidite	2
12	254	spheroidite	2
13	256	spheroidite	2
14	257	spheroidite	2
15	259	spheroidite	2
16	260	spheroidite	2
17	268	spheroidite	2
18	271	spheroidite	2
19	274	spheroidite	2
20	275	spheroidite	2
21	282	spheroidite	2
22	285	spheroidite	2
23	290	spheroidite	2
24	292	spheroidite	2
25	294	spheroidite	2
26	296	spheroidite	2
27	299	spheroidite	2
28	303	spheroidite	2
29	307	spheroidite	2
30	308	spheroidite	2
31	309	spheroidite	2
32	310	spheroidite	2
33	311	spheroidite	2
34	313	spheroidite	2
35	314	spheroidite	2
36	316	spheroidite	2
37	321	spheroidite	2
38	323	spheroidite	2
39	324	spheroidite	2
40	326	spheroidite	2
41	328	spheroidite	2
42	331	spheroidite	2
43	335	spheroidite	2
44	337	spheroidite	2
45	341	spheroidite	2
46	342	spheroidite	2
47	344	spheroidite	2
48	346	spheroidite	2
49	355	spheroidite	2
50	359	spheroidite	2
51	360	spheroidite	2

One-to-One SVM

```
class One2OneSVM:
    def __init__(self, n_class=4):
        self.n_class = n_class
        self.clfs = [[svm.SVC(kernel="rbf", C=1., gamma="auto")\
                         for i in range(0,self.n_class)]\
                         for j in range(0,self.n_class)]
        self.cv = np.zeros((self.n_class,self.n_class))
    def train(self, ltrain, feature, fold=10):
        # traversal all features
        for i in range(0, self.n_class-1):
            lis = ltrain[ltrain["label"] == i].index.to_numpy()
            for j in range(i+1, self.n_class):
                ljs = ltrain[ltrain["label"] == j].index.to_numpy()
                # Data
                X = np.concatenate(\
                  (feature[lis,:],\
                   feature[ljs,:]), axis=0)
                Y = np.concatenate((np.ones(len(lis))*i,np.ones(len(ljs))*j))
                # Train SVM
                scores = sklearn.model_selection.cross_val_score(self.clfs[i][j], X, Y, cv
                self.clfs[i][j].fit(X,Y)
                self.cv[i][j] = np.max(scores)

    def test_1v1_error(self, ltest, feature):
        # traversal all features
        errM = np.zeros((self.n_class, self.n_class))
        for i in range(0, self.n_class-1):
            lis = ltest[ltest["label"] == i].index.to_numpy()
            for j in range(i+1, self.n_class):
                ljs = ltest[ltest["label"] == j].index.to_numpy()
                # Data
                X = np.concatenate(\
                  (feature[lis,:],\
                   feature[ljs,:]), axis=0)
                Y = np.concatenate((np.ones(len(lis))*i,np.ones(len(ljs))*j))
                # Train SVM
                y_pred = self.clfs[i][j].predict(X)
                errM[i,j] = error(Y, y_pred)
        return errM

```

```
    def predict(self, feature):
        predM = np.zeros(( int(self.n_class * (self.n_class -1)/2) , feature.shape[0]))
        c = 0
        for i in range(0, self.n_class-1):
            for j in range(i+1, self.n_class):
                predM[c,:] = self.clfs[i][j].predict(feature)
                c += 1
        return st.mode(predM, axis=0, keepdims=True).mode[0,:] #majority voting

def error(ans, pred):
    assert len(ans) == len(pred)
    return (ans != pred).sum()/float(ans.size)
```

(a)

The convolution layer used and the cross-validated error estimate for each of the six pairwise two-label classifiers

(b)

Separate test error rates on the unused micrographs of each of the four categories, for the pairwise two-label classifiers and the multilabel one-vs-one voting classifier described previously. For the pairwise classifiers use only the test micrographs with the two labels used to train the classifier. For the multilabel classifier, use the test micrographs with the corresponding four labels.

```
def df_cv(m, clf, info=""):
    var1 = []
    var2 = []
    cvs = []
    errs = []
    for i in range(0, m.shape[0]-1):
        for j in range(i+1, m.shape[0]):
            var1.append(i)
            var2.append(j)
            cvs.append(clf.cv[i,j])
            errs.append(m[i,j])
    infos = [info] * len(errs)
    return pd.DataFrame({"Info": infos, "Label 1": var1, "Label 2": var2, "Test error": er
```

Pair-wise classifier

```
df_errors = []
for b in out_layer_ns:
    clf1 = One2OneSVM()
    clf1.train(ltrain, features_train[b])
    errs = clf1.test_1v1_error(ltest, features_test[b])
    df_errors.append(df_cv(errs, clf1, b))

res_error = pd.concat(df_errors)
print(res_error.to_string())
```

```
          Info  Label 1  Label 2  Test error  Cross Validation Score
0  block1_pool        0        1    0.823529                   0.500
1  block1_pool        0        2    0.290155                   0.500
2  block1_pool        0        3    0.157895                   0.625
3  block1_pool        1        2    0.080537                   0.500
4  block1_pool        1        3    0.466667                   0.625
5  block1_pool        2        3    0.071186                   0.625
0  block2_pool        0        1    0.823529                   0.500
1  block2_pool        0        2    0.290155                   0.500
2  block2_pool        0        3    0.157895                   0.625
3  block2_pool        1        2    0.080537                   0.500
4  block2_pool        1        3    0.466667                   0.625
5  block2_pool        2        3    0.071186                   0.625
0  block3_pool        0        1    0.823529                   0.500
1  block3_pool        0        2    0.290155                   0.500
2  block3_pool        0        3    0.157895                   0.625
3  block3_pool        1        2    0.080537                   0.500
4  block3_pool        1        3    0.466667                   0.625
5  block3_pool        2        3    0.071186                   0.625
0  block4_pool        0        1    0.823529                   0.500
1  block4_pool        0        2    0.290155                   0.500
2  block4_pool        0        3    0.157895                   0.625
3  block4_pool        1        2    0.080537                   0.500
4  block4_pool        1        3    0.466667                   0.625
5  block4_pool        2        3    0.071186                   0.625
0  block5_pool        0        1    0.823529                   0.500
1  block5_pool        0        2    0.290155                   0.500
2  block5_pool        0        3    0.157895                   0.625
3  block5_pool        1        2    0.080537                   0.500
4  block5_pool        1        3    0.466667                   0.625
```

```
5  block5_pool        2        3    0.071186                        0.625
```

Multiple one-vs-one classifier

```python
# Multiclass one-vs-one
dfm_errors = []
for b in out_layer_ns:
    clf = OneVsOneClassifier(svm.SVC(kernel="rbf", C=1., gamma="auto").fit(features_train[
          ltrain["label"].to_numpy(int)))
    clf.fit(features_train[b],\
          ltrain["label"].to_numpy(int))
    y_predm = clf.predict(features_test[b])
    dfm_errors.append(1 - error(y_predm, ltest["label"].to_numpy()))

# Display result
res_multi1v1 = pd.DataFrame({"Info": out_layer_ns, "Score": dfm_errors})
print(res_multi1v1.to_string())
```

```
          Info     Score
0  block1_pool  0.635731
1  block2_pool  0.635731
2  block3_pool  0.635731
3  block4_pool  0.635731
4  block5_pool  0.635731
```

(c)

For the mixed pearlite + spheroidite test micrographs, apply the trained pairwise classifier for pearlite vs. spheroidite and the multilabel voting classifier. Print the predicted labels by these two classifiers side by side (one row for each test micrograph). Comment your results

```python
ltestm = ltest[(ltest["primary_microconstituent"] == "pearlite") |\
      (ltest["primary_microconstituent"] == "spheroidite")]
feature_m = features_test["block5_pool"][ltestm.index.to_numpy(), :]
l = le.transform(["pearlite", "spheroidite"])

pred_pairs = clf1.clfs[l[0]][l[1]].predict(feature_m)
pred_multi = clf.predict(feature_m)

```

```python
res_ps = pd.DataFrame({"Test Label": le.inverse_transform(ltestm["label"]),\
                "Pairwise (pearlite vs. spheroidite)": le.inverse_transform(pred_pairs.astyp
                "Multi-OnevsOne": le.inverse_transform(pred_multi)})

print(res_ps.to_string())
```

```
     Test Label Pairwise (pearlite vs. spheroidite) Multi-OnevsOne
0    spheroidite                         spheroidite    spheroidite
1    spheroidite                         spheroidite    spheroidite
2    spheroidite                         spheroidite    spheroidite
3    spheroidite                         spheroidite    spheroidite
4    spheroidite                         spheroidite    spheroidite
5    spheroidite                         spheroidite    spheroidite
6    spheroidite                         spheroidite    spheroidite
7    spheroidite                         spheroidite    spheroidite
8    spheroidite                         spheroidite    spheroidite
9    spheroidite                         spheroidite    spheroidite
10   spheroidite                         spheroidite    spheroidite
11   spheroidite                         spheroidite    spheroidite
12   spheroidite                         spheroidite    spheroidite
13   spheroidite                         spheroidite    spheroidite
14   spheroidite                         spheroidite    spheroidite
15   spheroidite                         spheroidite    spheroidite
16   spheroidite                         spheroidite    spheroidite
17   spheroidite                         spheroidite    spheroidite
18   spheroidite                         spheroidite    spheroidite
19   spheroidite                         spheroidite    spheroidite
20   spheroidite                         spheroidite    spheroidite
21   spheroidite                         spheroidite    spheroidite
22   spheroidite                         spheroidite    spheroidite
23   spheroidite                         spheroidite    spheroidite
24   spheroidite                         spheroidite    spheroidite
25   spheroidite                         spheroidite    spheroidite
26   spheroidite                         spheroidite    spheroidite
27   spheroidite                         spheroidite    spheroidite
28   spheroidite                         spheroidite    spheroidite
29   spheroidite                         spheroidite    spheroidite
30   spheroidite                         spheroidite    spheroidite
31   spheroidite                         spheroidite    spheroidite
32   spheroidite                         spheroidite    spheroidite
33   spheroidite                         spheroidite    spheroidite
```

34	spheroidite	spheroidite	spheroidite
35	spheroidite	spheroidite	spheroidite
36	spheroidite	spheroidite	spheroidite
37	spheroidite	spheroidite	spheroidite
38	spheroidite	spheroidite	spheroidite
39	spheroidite	spheroidite	spheroidite
40	spheroidite	spheroidite	spheroidite
41	spheroidite	spheroidite	spheroidite
42	spheroidite	spheroidite	spheroidite
43	spheroidite	spheroidite	spheroidite
44	spheroidite	spheroidite	spheroidite
45	spheroidite	spheroidite	spheroidite
46	spheroidite	spheroidite	spheroidite
47	spheroidite	spheroidite	spheroidite
48	spheroidite	spheroidite	spheroidite
49	spheroidite	spheroidite	spheroidite
50	spheroidite	spheroidite	spheroidite
51	spheroidite	spheroidite	spheroidite
52	spheroidite	spheroidite	spheroidite
53	spheroidite	spheroidite	spheroidite
54	spheroidite	spheroidite	spheroidite
55	spheroidite	spheroidite	spheroidite
56	spheroidite	spheroidite	spheroidite
57	spheroidite	spheroidite	spheroidite
58	spheroidite	spheroidite	spheroidite
59	spheroidite	spheroidite	spheroidite
60	spheroidite	spheroidite	spheroidite
61	spheroidite	spheroidite	spheroidite
62	spheroidite	spheroidite	spheroidite
63	spheroidite	spheroidite	spheroidite
64	spheroidite	spheroidite	spheroidite
65	spheroidite	spheroidite	spheroidite
66	spheroidite	spheroidite	spheroidite
67	spheroidite	spheroidite	spheroidite
68	spheroidite	spheroidite	spheroidite
69	spheroidite	spheroidite	spheroidite
70	spheroidite	spheroidite	spheroidite
71	spheroidite	spheroidite	spheroidite
72	spheroidite	spheroidite	spheroidite
73	spheroidite	spheroidite	spheroidite
74	spheroidite	spheroidite	spheroidite
75	spheroidite	spheroidite	spheroidite
76	spheroidite	spheroidite	spheroidite

77	spheroidite	spheroidite	spheroidite
78	spheroidite	spheroidite	spheroidite
79	spheroidite	spheroidite	spheroidite
80	spheroidite	spheroidite	spheroidite
81	spheroidite	spheroidite	spheroidite
82	spheroidite	spheroidite	spheroidite
83	spheroidite	spheroidite	spheroidite
84	spheroidite	spheroidite	spheroidite
85	spheroidite	spheroidite	spheroidite
86	spheroidite	spheroidite	spheroidite
87	spheroidite	spheroidite	spheroidite
88	spheroidite	spheroidite	spheroidite
89	spheroidite	spheroidite	spheroidite
90	spheroidite	spheroidite	spheroidite
91	spheroidite	spheroidite	spheroidite
92	spheroidite	spheroidite	spheroidite
93	spheroidite	spheroidite	spheroidite
94	spheroidite	spheroidite	spheroidite
95	spheroidite	spheroidite	spheroidite
96	spheroidite	spheroidite	spheroidite
97	spheroidite	spheroidite	spheroidite
98	spheroidite	spheroidite	spheroidite
99	spheroidite	spheroidite	spheroidite
100	spheroidite	spheroidite	spheroidite
101	spheroidite	spheroidite	spheroidite
102	spheroidite	spheroidite	spheroidite
103	spheroidite	spheroidite	spheroidite
104	spheroidite	spheroidite	spheroidite
105	spheroidite	spheroidite	spheroidite
106	spheroidite	spheroidite	spheroidite
107	spheroidite	spheroidite	spheroidite
108	spheroidite	spheroidite	spheroidite
109	spheroidite	spheroidite	spheroidite
110	spheroidite	spheroidite	spheroidite
111	spheroidite	spheroidite	spheroidite
112	spheroidite	spheroidite	spheroidite
113	spheroidite	spheroidite	spheroidite
114	spheroidite	spheroidite	spheroidite
115	spheroidite	spheroidite	spheroidite
116	spheroidite	spheroidite	spheroidite
117	spheroidite	spheroidite	spheroidite
118	spheroidite	spheroidite	spheroidite
119	spheroidite	spheroidite	spheroidite

120	spheroidite	spheroidite	spheroidite
121	spheroidite	spheroidite	spheroidite
122	spheroidite	spheroidite	spheroidite
123	spheroidite	spheroidite	spheroidite
124	spheroidite	spheroidite	spheroidite
125	spheroidite	spheroidite	spheroidite
126	spheroidite	spheroidite	spheroidite
127	spheroidite	spheroidite	spheroidite
128	spheroidite	spheroidite	spheroidite
129	spheroidite	spheroidite	spheroidite
130	spheroidite	spheroidite	spheroidite
131	spheroidite	spheroidite	spheroidite
132	spheroidite	spheroidite	spheroidite
133	spheroidite	spheroidite	spheroidite
134	spheroidite	spheroidite	spheroidite
135	spheroidite	spheroidite	spheroidite
136	spheroidite	spheroidite	spheroidite
137	spheroidite	spheroidite	spheroidite
138	spheroidite	spheroidite	spheroidite
139	spheroidite	spheroidite	spheroidite
140	spheroidite	spheroidite	spheroidite
141	spheroidite	spheroidite	spheroidite
142	spheroidite	spheroidite	spheroidite
143	spheroidite	spheroidite	spheroidite
144	spheroidite	spheroidite	spheroidite
145	spheroidite	spheroidite	spheroidite
146	spheroidite	spheroidite	spheroidite
147	spheroidite	spheroidite	spheroidite
148	spheroidite	spheroidite	spheroidite
149	spheroidite	spheroidite	spheroidite
150	spheroidite	spheroidite	spheroidite
151	spheroidite	spheroidite	spheroidite
152	spheroidite	spheroidite	spheroidite
153	spheroidite	spheroidite	spheroidite
154	spheroidite	spheroidite	spheroidite
155	spheroidite	spheroidite	spheroidite
156	spheroidite	spheroidite	spheroidite
157	spheroidite	spheroidite	spheroidite
158	spheroidite	spheroidite	spheroidite
159	spheroidite	spheroidite	spheroidite
160	spheroidite	spheroidite	spheroidite
161	spheroidite	spheroidite	spheroidite
162	spheroidite	spheroidite	spheroidite

163	spheroidite	spheroidite	spheroidite
164	spheroidite	spheroidite	spheroidite
165	spheroidite	spheroidite	spheroidite
166	spheroidite	spheroidite	spheroidite
167	spheroidite	spheroidite	spheroidite
168	spheroidite	spheroidite	spheroidite
169	spheroidite	spheroidite	spheroidite
170	spheroidite	spheroidite	spheroidite
171	spheroidite	spheroidite	spheroidite
172	spheroidite	spheroidite	spheroidite
173	spheroidite	spheroidite	spheroidite
174	spheroidite	spheroidite	spheroidite
175	spheroidite	spheroidite	spheroidite
176	spheroidite	spheroidite	spheroidite
177	spheroidite	spheroidite	spheroidite
178	spheroidite	spheroidite	spheroidite
179	spheroidite	spheroidite	spheroidite
180	spheroidite	spheroidite	spheroidite
181	spheroidite	spheroidite	spheroidite
182	spheroidite	spheroidite	spheroidite
183	spheroidite	spheroidite	spheroidite
184	spheroidite	spheroidite	spheroidite
185	spheroidite	spheroidite	spheroidite
186	spheroidite	spheroidite	spheroidite
187	spheroidite	spheroidite	spheroidite
188	spheroidite	spheroidite	spheroidite
189	spheroidite	spheroidite	spheroidite
190	spheroidite	spheroidite	spheroidite
191	spheroidite	spheroidite	spheroidite
192	spheroidite	spheroidite	spheroidite
193	spheroidite	spheroidite	spheroidite
194	spheroidite	spheroidite	spheroidite
195	spheroidite	spheroidite	spheroidite
196	spheroidite	spheroidite	spheroidite
197	spheroidite	spheroidite	spheroidite
198	spheroidite	spheroidite	spheroidite
199	spheroidite	spheroidite	spheroidite
200	spheroidite	spheroidite	spheroidite
201	spheroidite	spheroidite	spheroidite
202	spheroidite	spheroidite	spheroidite
203	spheroidite	spheroidite	spheroidite
204	spheroidite	spheroidite	spheroidite
205	spheroidite	spheroidite	spheroidite

206	spheroidite	spheroidite	spheroidite
207	spheroidite	spheroidite	spheroidite
208	spheroidite	spheroidite	spheroidite
209	spheroidite	spheroidite	spheroidite
210	spheroidite	spheroidite	spheroidite
211	spheroidite	spheroidite	spheroidite
212	spheroidite	spheroidite	spheroidite
213	spheroidite	spheroidite	spheroidite
214	spheroidite	spheroidite	spheroidite
215	spheroidite	spheroidite	spheroidite
216	spheroidite	spheroidite	spheroidite
217	spheroidite	spheroidite	spheroidite
218	spheroidite	spheroidite	spheroidite
219	pearlite	spheroidite	spheroidite
220	spheroidite	spheroidite	spheroidite
221	spheroidite	spheroidite	spheroidite
222	spheroidite	spheroidite	spheroidite
223	pearlite	spheroidite	spheroidite
224	pearlite	spheroidite	spheroidite
225	spheroidite	spheroidite	spheroidite
226	spheroidite	spheroidite	spheroidite
227	pearlite	spheroidite	spheroidite
228	spheroidite	spheroidite	spheroidite
229	spheroidite	spheroidite	spheroidite
230	spheroidite	spheroidite	spheroidite
231	spheroidite	spheroidite	spheroidite
232	spheroidite	spheroidite	spheroidite
233	spheroidite	spheroidite	spheroidite
234	spheroidite	spheroidite	spheroidite
235	spheroidite	spheroidite	spheroidite
236	spheroidite	spheroidite	spheroidite
237	spheroidite	spheroidite	spheroidite
238	pearlite	spheroidite	spheroidite
239	pearlite	spheroidite	spheroidite
240	spheroidite	spheroidite	spheroidite
241	pearlite	spheroidite	spheroidite
242	pearlite	spheroidite	spheroidite
243	spheroidite	spheroidite	spheroidite
244	spheroidite	spheroidite	spheroidite
245	spheroidite	spheroidite	spheroidite
246	spheroidite	spheroidite	spheroidite
247	spheroidite	spheroidite	spheroidite
248	spheroidite	spheroidite	spheroidite

249	spheroidite	spheroidite	spheroidite
250	spheroidite	spheroidite	spheroidite
251	spheroidite	spheroidite	spheroidite
252	pearlite	spheroidite	spheroidite
253	pearlite	spheroidite	spheroidite
254	pearlite	spheroidite	spheroidite
255	pearlite	spheroidite	spheroidite
256	pearlite	spheroidite	spheroidite
257	spheroidite	spheroidite	spheroidite
258	spheroidite	spheroidite	spheroidite
259	spheroidite	spheroidite	spheroidite
260	spheroidite	spheroidite	spheroidite
261	spheroidite	spheroidite	spheroidite
262	spheroidite	spheroidite	spheroidite
263	spheroidite	spheroidite	spheroidite
264	spheroidite	spheroidite	spheroidite
265	spheroidite	spheroidite	spheroidite
266	spheroidite	spheroidite	spheroidite
267	pearlite	spheroidite	spheroidite
268	pearlite	spheroidite	spheroidite
269	spheroidite	spheroidite	spheroidite
270	pearlite	spheroidite	spheroidite
271	spheroidite	spheroidite	spheroidite
272	spheroidite	spheroidite	spheroidite
273	spheroidite	spheroidite	spheroidite
274	pearlite	spheroidite	spheroidite
275	pearlite	spheroidite	spheroidite
276	spheroidite	spheroidite	spheroidite
277	pearlite	spheroidite	spheroidite
278	spheroidite	spheroidite	spheroidite
279	spheroidite	spheroidite	spheroidite
280	spheroidite	spheroidite	spheroidite
281	spheroidite	spheroidite	spheroidite
282	spheroidite	spheroidite	spheroidite
283	spheroidite	spheroidite	spheroidite
284	spheroidite	spheroidite	spheroidite
285	spheroidite	spheroidite	spheroidite
286	pearlite	spheroidite	spheroidite
287	pearlite	spheroidite	spheroidite
288	spheroidite	spheroidite	spheroidite
289	spheroidite	spheroidite	spheroidite
290	spheroidite	spheroidite	spheroidite
291	spheroidite	spheroidite	spheroidite

292	spheroidite	spheroidite	spheroidite
293	pearlite	spheroidite	spheroidite
294	spheroidite	spheroidite	spheroidite
295	spheroidite	spheroidite	spheroidite
296	pearlite	spheroidite	spheroidite
297	pearlite	spheroidite	spheroidite

(d)

Now apply the multilabel classifier on the pearlite + Widmanst¨atten and martensite micrographs and print the predicted labels. Compare to the results in part (c)

```
df_micro2 = df_micro2[(df_micro2["primary_microconstituent"] == "pearlite+widmanstatten")
(df_micro2["primary_microconstituent"] == "martensite")]

# Encode labels
le2 = preprocessing.LabelEncoder()
le2.fit(df_micro2["primary_microconstituent"].unique())
list(le2.classes_)
```

```
['martensite', 'pearlite+widmanstatten']
```

```
dlabel2 = le2.transform(df_micro2["primary_microconstituent"])
df_micro2.insert(2, "label", dlabel2)
```

```
df_micro2
```

```
/Users/stevenchiu/miniconda/lib/python3.9/site-packages/IPython/core/formatters.py:342: Futu
  return method()
```

	path	primary_microconstituent	label
15	data/CMU-UHCS_Dataset/images/micrograph20.tif	martensite	0
29	data/CMU-UHCS_Dataset/images/micrograph41.tif	martensite	0
31	data/CMU-UHCS_Dataset/images/micrograph44.tif	martensite	0
63	data/CMU-UHCS_Dataset/images/micrograph99.tif	martensite	0
71	data/CMU-UHCS_Dataset/images/micrograph114.tif	martensite	0
95	data/CMU-UHCS_Dataset/images/micrograph168.tif	martensite	0
186	data/CMU-UHCS_Dataset/images/micrograph345.tif	martensite	0
187	data/CMU-UHCS_Dataset/images/micrograph346.tif	pearlite+widmanstatten	1
199	data/CMU-UHCS_Dataset/images/micrograph366.tif	martensite	0
208	data/CMU-UHCS_Dataset/images/micrograph381.tif	martensite	0
251	data/CMU-UHCS_Dataset/images/micrograph459.tif	martensite	0
272	data/CMU-UHCS_Dataset/images/micrograph490.tif	pearlite+widmanstatten	1
273	data/CMU-UHCS_Dataset/images/micrograph492.tif	martensite	0
276	data/CMU-UHCS_Dataset/images/micrograph496.tif	pearlite+widmanstatten	1
280	data/CMU-UHCS_Dataset/images/micrograph502.tif	martensite	0
289	data/CMU-UHCS_Dataset/images/micrograph523.tif	pearlite+widmanstatten	1
305	data/CMU-UHCS_Dataset/images/micrograph553.tif	pearlite+widmanstatten	1
405	data/CMU-UHCS_Dataset/images/micrograph753.tif	pearlite+widmanstatten	1
421	data/CMU-UHCS_Dataset/images/micrograph785.tif	martensite	0
445	data/CMU-UHCS_Dataset/images/micrograph836.tif	pearlite+widmanstatten	1
455	data/CMU-UHCS_Dataset/images/micrograph857.tif	pearlite+widmanstatten	1
466	data/CMU-UHCS_Dataset/images/micrograph875.tif	pearlite+widmanstatten	1
470	data/CMU-UHCS_Dataset/images/micrograph882.tif	pearlite+widmanstatten	1
489	data/CMU-UHCS_Dataset/images/micrograph911.tif	pearlite+widmanstatten	1
504	data/CMU-UHCS_Dataset/images/micrograph932.tif	pearlite+widmanstatten	1
513	data/CMU-UHCS_Dataset/images/micrograph951.tif	martensite	0
516	data/CMU-UHCS_Dataset/images/micrograph954.tif	martensite	0
521	data/CMU-UHCS_Dataset/images/micrograph963.tif	martensite	0
529	data/CMU-UHCS_Dataset/images/micrograph984.tif	martensite	0
545	data/CMU-UHCS_Dataset/images/micrograph1009.tif	martensite	0
548	data/CMU-UHCS_Dataset/images/micrograph1012.tif	martensite	0
561	data/CMU-UHCS_Dataset/images/micrograph1030.tif	martensite	0
570	data/CMU-UHCS_Dataset/images/micrograph1046.tif	pearlite+widmanstatten	1
571	data/CMU-UHCS_Dataset/images/micrograph1048.tif	martensite	0
576	data/CMU-UHCS_Dataset/images/micrograph1055.tif	martensite	0
590	data/CMU-UHCS_Dataset/images/micrograph1078.tif	pearlite+widmanstatten	1
591	data/CMU-UHCS_Dataset/images/micrograph1079.tif	martensite	0
599	data/CMU-UHCS_Dataset/images/micrograph1097.tif	pearlite+widmanstatten	1
605	data/CMU-UHCS_Dataset/images/micrograph1104.tif	pearlite+widmanstatten	1
614	data/CMU-UHCS_Dataset/images/micrograph1123.tif	pearlite+widmanstatten	1
642	data/CMU-UHCS_Dataset/images/micrograph1174.tif	martensite	0
644	data/CMU-UHCS_Dataset/images/micrograph1177.tif	martensite	0
662	data/CMU-UHCS_Dataset/images/micrograph1204.tif	pearlite+widmanstatten	1
668	data/CMU-UHCS_Dataset/images/micrograph1214.tif	pearlite+widmanstatten	1
669	data/CMU-UHCS_Dataset/images/micrograph1215.tif	pearlite+widmanstatten	1
690	data/CMU-UHCS_Dataset/images/micrograph1255.tif	pearlite+widmanstatten	1
699	data/CMU-UHCS_Dataset/images/micrograph1267.tif	pearlite+widmanstatten	1
707	data/CMU-UHCS_Dataset/images/micrograph1281.tif	martensite	0
711	data/CMU-UHCS_Dataset/images/micrograph1287.tif	pearlite+widmanstatten	1
726	data/CMU-UHCS_Dataset/images/micrograph1316.tif	martensite	0
755	data/CMU-UHCS_Dataset/images/micrograph1370.tif	pearlite+widmanstatten	1
771	data/CMU-UHCS_Dataset/images/micrograph1395.tif	pearlite+widmanstatten	1

```python
# Feature extraction with VGG16
if os.path.exists(os.path.join(dpath, "feature_test2.pkl")) == False:
    fs_test2 = np.zeros((df_micro2.shape[0], out_shapes[-1]))
    m = "block5_pool"
    for j, ph in tqdm(enumerate(df_micro2["path"])):
        x = load_image(ph)
        xb = extmodel[m].predict(x, verbose = 0) # silence output
        F = np.mean(xb,axis=(0,1,2))
        # Save features
        fs_test2[j, :] = F

    # Save data
    ## Create new files
    fs_test2_p = open(os.path.join(dpath, "feature_test2.pkl"), "wb")
    ## Write
    pickle.dump(fs_test2, fs_test2_p)
    ## Close files
    fs_test2_p.close()
```

```python
#load data
fs_test2_p  = open(os.path.join(dpath, "feature_test2.pkl"), "rb")
fs_test2 = pickle.load(fs_test2_p) # train feature
fs_test2_p .close()
```

```python
pred_multi2 = clf.predict(fs_test2)

res_ps2 = pd.DataFrame({"Test Label": le2.inverse_transform(df_micro2["label"]),\
              "Multi-OnevsOne": le.inverse_transform(pred_multi2)})

print(res_ps2.to_string())
```

```
                Test Label Multi-OnevsOne
0               martensite    spheroidite
1               martensite    spheroidite
2               martensite    spheroidite
3               martensite        network
4               martensite        network
5               martensite        network
6               martensite    spheroidite
7  pearlite+widmanstatten    spheroidite
```

8	martensite	spheroidite
9	martensite	network
10	martensite	network
11	pearlite+widmanstatten	spheroidite
12	martensite	network
13	pearlite+widmanstatten	spheroidite
14	martensite	network
15	pearlite+widmanstatten	spheroidite
16	pearlite+widmanstatten	spheroidite
17	pearlite+widmanstatten	spheroidite
18	martensite	network
19	pearlite+widmanstatten	network
20	pearlite+widmanstatten	network
21	pearlite+widmanstatten	network
22	pearlite+widmanstatten	network
23	pearlite+widmanstatten	spheroidite
24	pearlite+widmanstatten	spheroidite
25	martensite	network
26	martensite	network
27	martensite	network
28	martensite	network
29	martensite	spheroidite
30	martensite	spheroidite
31	martensite	network
32	pearlite+widmanstatten	network
33	martensite	network
34	martensite	spheroidite
35	pearlite+widmanstatten	network
36	martensite	spheroidite
37	pearlite+widmanstatten	network
38	pearlite+widmanstatten	network
39	pearlite+widmanstatten	spheroidite
40	martensite	network
41	martensite	spheroidite
42	pearlite+widmanstatten	spheroidite
43	pearlite+widmanstatten	network
44	pearlite+widmanstatten	network
45	pearlite+widmanstatten	spheroidite
46	pearlite+widmanstatten	network
47	martensite	network
48	pearlite+widmanstatten	spheroidite
49	martensite	network
50	pearlite+widmanstatten	network

```
51  pearlite+widmanstatten    spheroidite
52             martensite        network
53  pearlite+widmanstatten    spheroidite
54             martensite    spheroidite
55             martensite    spheroidite
56             martensite    spheroidite
57             martensite    spheroidite
58             martensite        network
59  pearlite+widmanstatten        network
60             martensite        network
61             martensite    spheroidite
62             martensite    spheroidite
```

References

Braga-Neto, Ulisses. 2020. *Fundamentals of Pattern Recognition and Machine Learning.* Springer.