In [5]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving IX-11-01917_0004_0001.JPG to IX-11-01917_0004_0001.JPG
Saving IX-11-01917_0004_0002.JPG to IX-11-01917_0004_0002.JPG
Saving IX-11-01917_0004_0003.JPG to IX-11-01917_0004_0003.JPG
Saving IX-11-01917_0004_0004.JPG to IX-11-01917_0004_0004.JPG
Saving IX-11-01917_0004_0005.JPG to IX-11-01917_0004_0005.JPG
Saving IX-11-01917_0004_0006.JPG to IX-11-01917_0004_0006.JPG
Saving IX-11-01917_0004_0007.JPG to IX-11-01917_0004_0007.JPG
Saving IX-11-01917_0004_0008.JPG to IX-11-01917_0004_0008.JPG
Saving IX-11-01917_0004_0009.JPG to IX-11-01917_0004_0009.JPG
Saving IX-11-01917_0004_0010.JPG to IX-11-01917_0004_0010.JPG
```

In [19]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

import tensorflow as tf

import numpy as np
import pandas as pd
import pathlib
from sklearn.model_selection import GroupKFold

from tqdm import tqdm
from random import choices
```

In [20]:
```
from keras.layers import Input, Conv2D, MaxPooling2D
from keras.layers import Dense, Flatten
from keras.models import Model

_input = Input((224,224,1))
```

```
conv1  = Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu")(_input)
conv2  = Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu")(conv1)
pool1  = MaxPooling2D((2, 2))(conv2)

conv3  = Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu")(pool1)
conv4  = Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu")(conv3)
pool2  = MaxPooling2D((2, 2))(conv4)

conv5  = Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu")(pool2)
conv6  = Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu")(conv5)
conv7  = Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu")(conv6)
pool3  = MaxPooling2D((2, 2))(conv7)

conv8  = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(pool3)
conv9  = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(conv8)
conv10 = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(conv9)
pool4  = MaxPooling2D((2, 2))(conv10)

conv11 = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(pool4)
conv12 = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(conv11)
conv13 = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(conv12)
pool5  = MaxPooling2D((2, 2))(conv13)

flat   = Flatten()(pool5)
dense1 = Dense(4096, activation="relu")(flat)
dense2 = Dense(4096, activation="relu")(dense1)
output = Dense(1000, activation="softmax")(dense2)

vgg16_model  = Model(inputs=_input, outputs=output)
```

In [21]:

```
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
import matplotlib.pyplot as plt
from PIL import Image
import seaborn as sns
import pandas as pd
import numpy as np
import os
```

In [22]:

```
img1 = "11-01917_0004_0007.JPG"
```

```
img2 = "11-01917_0004_0008.JPG"
img3 = "11-01917_0004_0009.JPG"
img4 = "11-01917_0004_0010.JPG"
imgs = [img1, img2, img3, img4]
```

In [23]:
```
def _load_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img

def _get_predictions(_model):
    f, ax = plt.subplots(1, 4)
    f.set_size_inches(80, 40)
    for i in range(4):
        ax[i].imshow(Image.open(imgs[i]).resize((200, 200), Image.ANTIALIAS))
    plt.show()

    f, axes = plt.subplots(1, 4)
    f.set_size_inches(80, 20)
    for i,img_path in enumerate(imgs):
        img = _load_image(img_path)
        preds  = decode_predictions(_model.predict(img), top=3)[0]
        b = sns.barplot(y=[c[1] for c in preds], x=[c[2] for c in preds], color="gray", ax=axes[i])
        b.tick_params(labelsize=55)
        f.tight_layout()
```

In [25]:
```
resnet50 = ResNet50(weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 1s 0us/step
```

In [ ]:
```
def _get_features(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_data = image.img_to_array(img)
    img_data = np.expand_dims(img_data, axis=0)
    img_data = preprocess_input(img_data)
    resnet_features = resnet50.predict(img_data)
    return resnet_features
```

```
img_path = ".jpg"
resnet_features = _get_features(img_path)
```

In [ ]:
```
features_representation_1 = resnet_features.flatten()
features_representation_2 = resnet_features.squeeze()

print ("Shape 1: ", features_representation_1.shape)
print ("Shape 2: ", features_representation_2.shape)
```

In [ ]:
```
features = {"" : [], "" : [], "" : []}
testimgs = []
for label, val in data.items():
    for k, each in enumerate(val):
        if label == "test" and k == 0:
            img_path = basepath + "//" + each
            testimgs.append(img_path)
        elif label == "test" and k == 1:
            img_path = basepath + "//" + each
            testimgs.append(img_path)
        else:
            img_path = basepath + label.title() + "/" + each
        feats = _get_features(img_path)
        features[label].append(feats.flatten())
dataset = pd.DataFrame()
for label, feats in features.items():
    temp_df = pd.DataFrame(feats)
    temp_df['label'] = label
    dataset = dataset.append(temp_df, ignore_index=True)
dataset.head()
```