Food(3) Image Active Learning with Resnet50 on 3 Categories

We are experimenting with 3 categories to improve label performance - Bread, Meat, Veg-Fruits.

Originally , Food-11 is a dataset containing 16643 food images grouped in 11 major food categories. The 11 categories are Bread, Dairy product, Dessert, Egg, Fried food, Meat, Noodles/Pasta, Rice, Seafood, Soup, and Vegetable/Fruit. Similar as Food-5K dataset, the whole dataset is divided in three parts: training, validation and evaluation. The same naming convention is used, where ID 0-10 refers to the 11 food categories respectively.

Conclusion: Uncertainty Sampling out performed all strategies with a **16%** lift in f1 score.

Experiment results Summary

Overall Classification Metrics Comparison of Base model and Active Learning strategies:

Query Strategy	Initial Model f1 Score	Active Learning f1 Score
Random Sampling	0.71	0.72
Uncertainty Sampling	0.71	0.87
Entropy Sampling	0.71	0.83
Margin Sampling	0.71	0.81

Class-wise Precision Comparison with and without Active Learning:

Query Strategy	Bread *Initial Model precision*	Bread *Active Learning precision*	Meat *Initial Model precision*	Meat *Active Learning precision*
Random Sampling	0.93	0.95	0.60	0.61
Uncertainty Sampling	0.93	0.94	0.60	0.79
Entropy Sampling	0.93	0.94	0.60	0.84
Margin Sampling	0.93	0.97	0.60	0.70

Class-wise Recall Comparison with and without Active Learning:

Query Strategy	Bread *Initial Model recall*	Bread *Active Learning recall*	Meat *Initial Model recall*	Meat *Active Learning recall*	Vegetables- *Initial Model*
Random Sampling	0.34	0.39	0.99	0.99	0.79
Uncertainty Sampling	0.34	0.74	0.99	0.95	0.79
Entropy Sampling	0.34	0.71	0.99	0.87	0.79
Margin Sampling	0.34	0.61	0.99	0.99	0.79

10% (250 images out of 2500) of training data was used for initial training and 11% (250 images out of 2250) of remaining for Query Pool.

Original Training Dataset Size -

1. Bread = 994
2. Meat = 1,325
3. Vegetables-Fruits = 709

Datasize used for this Experiment -

1. Bread = 800
2. Meat = 1,000
3. Vegetables-Fruits = 700

> (Bread and Meat images were selected by ranking images by image size and eliminating large sized images)

This notebook tries to implement -
Active learning using modAL library - The implementation uses an initial set of labeled examples in order to train an initial model, and then queries unlabeled data to use the best possible examples to train the next model. The notebook uses skorch (scikit-learn compatible neural network library that wraps PyTorch) to handle the training mechanism - to be explored further.

Querying Strategies:

1. Uncertainty Sampling: Selects the least sure instances for labelling.

> a. Classifier uncertainty = 1 - P(prediction is correct)

2. Entropy Sampling: Selects the instances where the class probabilities have the largest entropy.

> a. Entropy of the class probabilities. Example, Confusing predictions with higher probabilities for multiple classes will have higher entropy. Predictions with single dominating probability class will have lower entropy.

3. Margin Sampling: Selects the instances where the difference between the first most likely and second most likely classes are the smallest.

> a. Margin uncertainty, which is the difference of the probabilities of first and second most likely predictions.

Additional resources -

1. https://dagshub.com/blog/active-learning-your-way-to-better-models/
2. https://arxiv.org/pdf/2203.13450.pdf
3. https://burrsettles.com/pub/settles.activelearning.pdf
4. https://www.youtube.com/watch?v=W2bJH0iXTKc&t=1011s
5. https://skorch.readthedocs.io/en/stable/
6. https://arxiv.org/pdf/2203.07034.pdf
7. https://paperswithcode.com/paper/active-learning-in-annotating-micro-blogs

Active Learning Experiments

Load Folder and Libraries

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install modAL
!pip install -U skorch
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting modAL
  Downloading modAL-0.4.1-py3-none-any.whl (27 kB)
Requirement already satisfied: numpy>=1.13 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pandas>=1.1.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (f
Installing collected packages: modAL
Successfully installed modAL-0.4.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
```

```
Collecting skorch
  Downloading skorch-0.11.0-py3-none-any.whl (155 kB)
     |████████████████████████████████| 155 kB 5.0 MB/s
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn>=0.19.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: tqdm>=4.14.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pac
Installing collected packages: skorch
Successfully installed skorch-0.11.0
```

```
from modAL.models import ActiveLearner
from modAL.uncertainty import uncertainty_sampling, entropy_sampling, margin_sampling
import numpy as np
import pandas as pd

import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
# import torchvision.transforms as transforms
from torchsummary import summary
import os
import time
import shutil

from imutils import paths
from tqdm import tqdm
import matplotlib.pyplot as plt

import torch
from torch import optim
from torchvision import models
from torchvision import transforms
from torch.cuda.amp import autocast
from torchvision import datasets
from torch.utils.data import DataLoader, Subset
from torchvision.models import densenet121
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from pathlib import Path
import copy

from skorch import NeuralNetClassifier

import warnings
warnings.filterwarnings('ignore')
```

Set Parameters

```
# define path to the original dataset
folder = Path('/content/drive/MyDrive/Colab Notebooks/active learning')
DATA_PATH = "/content/drive/MyDrive/Colab Notebooks/active learning/Food-3"

# define base path to store our modified dataset
BASE_PATH = "/content/drive/MyDrive/Colab Notebooks/active learning/Food-3"

# define paths to separate train, validation, and test splits
TRAIN = os.path.join(BASE_PATH, "training")
VAL = os.path.join(BASE_PATH, "validation")
TEST = os.path.join(BASE_PATH, "evaluation")

# initialize the list of class label names
CLASSES = ["Bread", "Meat", "Vegetable-Fruit"]

# specify ImageNet mean and standard deviation and image size
MEAN = [0.485, 0.456, 0.406]
STD = [0.229, 0.224, 0.225]
IMAGE_SIZE = 128

# set the device to be used for training and evaluation
DEVICE = torch.device("cuda")

# specify training hyperparameters
LOCAL_BATCH_SIZE = 128
PRED_BATCH_SIZE = 4
EPOCHS = 20
LR = 0.0001

# determine the number of GPUs we have
NUM_GPU = torch.cuda.device_count()
print(f"[INFO] number of GPUs found: {NUM_GPU}...")

# determine the batch size based on the number of GPUs
BATCH_SIZE = LOCAL_BATCH_SIZE * NUM_GPU
print(f"[INFO] using a batch size of {BATCH_SIZE}...")

# Assign device
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

# Set Seed
torch.manual_seed(42)
np.random.seed(42)
```

```
[INFO] number of GPUs found: 1...
[INFO] using a batch size of 128...
```

```
cuda:0
```

Dataset Transformation and Loaders

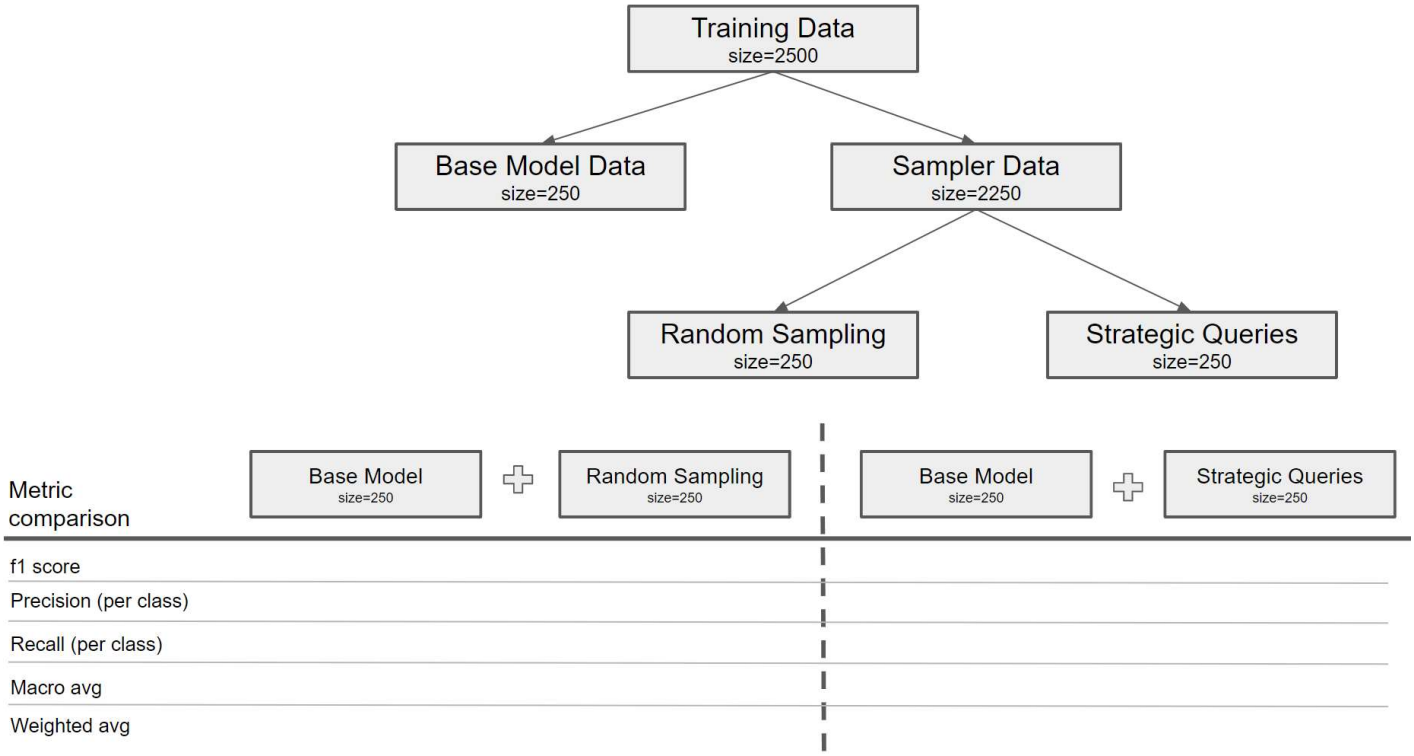
```
# define augmentation pipelines
trainTansform = transforms.Compose([
    transforms.RandomResizedCrop(IMAGE_SIZE),
    # transforms.RandomHorizontalFlip(),
    # transforms.RandomRotation(90),
    transforms.ToTensor(),
    transforms.Normalize(mean=MEAN, std=STD)
])

testTransform = transforms.Compose([
    transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=MEAN, std=STD)
])
```

Sampling Original Training data for Experimentation and scoring

```
image_datasets = {
    'train':
    datasets.ImageFolder(TRAIN, transform=trainTansform),
    'validation':
    datasets.ImageFolder(VAL, transform=testTransform),
    'test':
    datasets.ImageFolder(TEST, transform=testTransform),
    'sampler':
    datasets.ImageFolder(TRAIN, transform=trainTansform)
}

image_trainfolder = image_datasets['train']
train_sampler_idx, base_train_idx, target_sampler, base_target = train_test_split(range(len(i

train_ds = Subset(image_trainfolder, base_train_idx)
sampler_ds = Subset(image_trainfolder, train_sampler_idx)


dataloaders = {
    'train':
    torch.utils.data.DataLoader(train_ds #image_datasets['train']
                                ,batch_size=250,
                                shuffle=False,
                                num_workers=0 ),
    'validation':
```

```
    torch.utils.data.DataLoader(image_datasets['validation'],
                                batch_size=1043,
                                shuffle=False,
                                num_workers=0),
    'test':
    torch.utils.data.DataLoader(image_datasets['test'],
                                batch_size=1031,
                                shuffle=False,
                                num_workers=0),
    'sampler':
    torch.utils.data.DataLoader(sampler_ds #image_datasets['train']
                                ,batch_size=2250,
                                shuffle=False,
                                num_workers=0)
}
```

Experiments Overview



RESNET50 Model Training

```
# Load model and define training parameters
model = models.resnet50(pretrained=True).to(device)

for param in model.parameters():
```

```
        param.requires_grad = False

    model.fc = nn.Sequential(
                   nn.Linear(2048, 128),
                   nn.ReLU(inplace=True),
                   nn.Linear(128, 3)).to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.fc.parameters())
```

```
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/
100%                                  97.8M/97.8M [00:00<00:00, 223MB/s]
```

```
    # Define Model training function with 'n' epochs

    def train_model(model, criterion, optimizer, num_epochs=3):
        for epoch in range(num_epochs):
            print('Epoch {}/{}'.format(epoch+1, num_epochs))
            print('-' * 10)

            for phase in ['train', 'validation']:
                if phase == 'train':
                    model.train()
                else:
                    model.eval()

                running_loss = 0.0
                running_corrects = 0

                for inputs, labels in dataloaders[phase]:
                    inputs = inputs.to(device)
                    labels = labels.to(device)

                    outputs = model(inputs)
                    loss = criterion(outputs, labels)

                    if phase == 'train':
                        optimizer.zero_grad()
                        loss.backward()
                        optimizer.step()

                    _, preds = torch.max(outputs, 1)
                    running_loss += loss.item() * inputs.size(0)
                    running_corrects += torch.sum(preds == labels.data)

                epoch_loss = running_loss / len(image_datasets[phase])
                epoch_acc = running_corrects.double() / len(image_datasets[phase])

                print('{} loss: {:.4f}, acc: {:.4f}'.format(phase,
                                                            epoch_loss,
```

```
                                                    epoch_acc))
    return model
```

Train model and save weights for future pre-trained usage

```
# Uncommment the below code lines in case if you want to train the RESNET50 model

# model_trained = train_model(model, criterion, optimizer, num_epochs=3)

# # # !mkdir models

# torch.save(model_trained.state_dict(), '/content/drive/MyDrive/Colab Notebooks/models/weigh
```

Load pre-train RESNET50

```
# Load model from weights when not training on the data.
model.load_state_dict(torch.load('/content/drive/MyDrive/Colab Notebooks/models/weightsv1_2.h

# Define Classifier
classifier = NeuralNetClassifier(model, criterion=criterion)


# Creating deep copies of original Classifier for Experimentation (Torch models/classifiers a

classifier_random_sampling = copy.deepcopy(classifier)
classifier_uncertainity = copy.deepcopy(classifier)
classifier_entropy = copy.deepcopy(classifier)
classifier_margin = copy.deepcopy(classifier)


# Train loader to Array
trainLoader = dataloaders['train']
X, y = next(iter(trainLoader))
X = X.detach().cpu().numpy()
y = y.detach().cpu().numpy()

print('Train data shape: ', X.shape, '\n',
      'Number of classes: ', len(set(y)),'\n',
      'all classes : ', trainLoader.dataset.dataset.class_to_idx)

# Test loader to Array
testLoader = dataloaders['test']
X_test, y_test = next(iter(testLoader))
X_test = X_test.detach().cpu().numpy()
y_test = y_test.detach().cpu().numpy()

print('\n\nTest data shape: ', X_test.shape, '\n',
      'Number of classes: ', len(set(y)),'\n',
```

```
      'all classes : ', testLoader.dataset.class_to_idx)

# Sampler loader to Array
samplerLoader = dataloaders['sampler']
X_sampler, y_sampler = next(iter(samplerLoader))
X_sampler = X_sampler.detach().cpu().numpy()
y_sampler = y_sampler.detach().cpu().numpy()

print('\n\nSampler data shape: ', X_sampler.shape, '\n',
      'Number of classes: ', len(set(y_sampler)),'\n',
      'all classes : ', samplerLoader.dataset.dataset.class_to_idx)
```

```
Train data shape:  (250, 3, 128, 128)
 Number of classes:  3
 all classes :  {'Bread': 0, 'Meat': 1, 'Vegetable-Fruit': 2}


Test data shape:  (1031, 3, 128, 128)
 Number of classes:  3
 all classes :  {'Bread': 0, 'Meat': 1, 'Vegetable-Fruit': 2}


Sampler data shape:  (2250, 3, 128, 128)
 Number of classes:  3
 all classes :  {'Bread': 0, 'Meat': 1, 'Vegetable-Fruit': 2}
```

Define custom Query/Sampling functions for Active Learner (modAL)

```
# Function for random sampling (stratified) for Active learner query

def random_samp(clf, X_samp, y_samp, n_instances):
  # Randomly sample the remaining training data
  # INPUTS
  # clf : learner [modAL learner]
  # X_samp : [array] data to be queries or sampled
  # y_samp : [array] Targets of data to be sampled
  # n_instances : Number of images to be sampled
  # RETURNS [array] of indices of the queried X_samp, [array] Sampled output data from X_samp

  test_size_sampling = n_instances/len(X_samp) #Calculate the split ratio for the query
  _t, query_idx, _y, _ = train_test_split(range(len(X_samp)), y_samp, test_size=test_size_sam
  return query_idx, X_samp[query_idx]


# Function 2


# Function 3
```

Define image visualization functions for use during Query experiment

```
# Display sample image predictions, n correct predictions and n wrong predictions

def display_predictions(nsamples, X_in, y_in, y_pred, classmap):
  correct = []
  wrong = []
  for i in range(len(y_in)):
    if y_in[i] == y_pred[i]:
      correct.append(i)
    else:
      wrong.append(i)

  # Choose random n samples for each correct/wrong predictions
  cor_idx = list(np.random.choice(correct, size=nsamples))
  wrn_idx = list(np.random.choice(wrong, size=nsamples))

  idx = cor_idx + wrn_idx
  arr = X_in.copy()
  # arr = np.reshape((3, 128,128), (128, 128, 3))
  arr = arr.swapaxes(1, 2)
  arr = arr.swapaxes(2, 3)
  fig, axes = plt.subplots(1, nsamples*2, figsize=(24,8))
  axes = axes.flatten()
  for i, ax in enumerate(axes):
    cnt = 1
    ind = idx[i]
    class_name = [name for name, num in classmap.items() if num == y_in[ind]][0]
    pred_name = [name for name, num in classmap.items() if num == y_pred[ind]][0]
    # print("\nGround Truth = ", class_name, ",  Prediction = ", pred_name)
    num_cols = 1
    num_rows = 1
    scale = 2
    Image = np.clip(np.array(((arr[ind]+1)/2)*255, np.int32),  0, 255)
    ax.imshow(Image)
    title = "GTruth:" + class_name + "\nPred:" + pred_name
    ax.title.set_text(title)
    # ax.titlesize(2)
    # ax = plt.subplot(nsamples*2, 1, cnt)
    # plt.imshow(arr[i], cmap='')
    cnt+=1
  return None


# Display top 5 and tail 5 images while looping Queries
warnings.filterwarnings( "ignore", module = "matplotlib\..*" )

def query_headtail(X_in, y_in, classmap, ntop, ntail):
  idx = list(range(ntop)) + list(range(-ntail, 0))
  arr = X_in.copy()
  # arr = np.reshape((3, 128,128), (128, 128, 3))
  arr = arr.swapaxes(1, 2)
```

```
  arr = arr.swapaxes(2, 3)
  fig, axes = plt.subplots(1, 10, figsize=(24,8))
  axes = axes.flatten()
  for i, ax in enumerate(axes):
  # for i in idx:
    cnt = 0
    ind = idx[i]
    class_name = [name for name, num in classmap.items() if num == y_in[ind]][0]
    # print("\nGround Truth = ", class_name)
    num_cols = 1
    num_rows = 1
    scale = 2
    # plt.subplot(2,10,cnt+1)    # the number of images in the grid is 5*5 (25)
    Image = np.clip(np.array(((arr[ind]+1)/2)*255, np.int32), 0, 255)
    ax.imshow(Image)
    ax.title.set_text(class_name)
    # ax.axis('off')
    # ax = plt.subplot(nsamples*2, 1, cnt)
    # plt.imshow(arr[i], cmap='')
    cnt+=1
  return fig.show()
```

Initialize modAL Learner for Base model

```
# initialize ActiveLearner
learner = ActiveLearner(
    estimator = classifier_random_sampling,
    X_training = X, y_training = y
    , query_strategy=random_samp,
)

class_map = trainLoader.dataset.dataset.class_to_idx
```

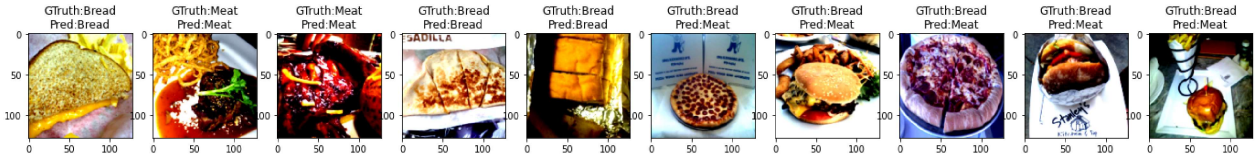
epoch	train_loss	valid_acc	valid_loss	dur
1	0.9375	0.5800	0.8269	15.9385
2	0.8771	0.7600	0.7890	15.4435
3	0.8390	0.9000	0.7700	15.4500
4	0.8119	0.8800	0.7600	15.3978
5	0.7903	0.8600	0.7521	15.4736
6	0.7713	0.8000	0.7451	15.5910
7	0.7531	0.7800	0.7384	15.3451
8	0.7355	0.7800	0.7314	15.3260
9	0.7181	0.8000	0.7235	15.2957
10	0.7011	0.8400	0.7150	15.4077

Initial **Base Model** Performance

```
# Predict on test split using initial model learner
pred1 = learner.predict(X_test)
print(classification_report(y_test, pred1))
```

```
                  precision    recall  f1-score   support

               0       0.93      0.34      0.49       368
               1       0.60      0.99      0.75       432
               2       0.96      0.79      0.86       231

        accuracy                           0.71      1031
       macro avg       0.83      0.70      0.70      1031
    weighted avg       0.80      0.71      0.68      1031
```

```
# Display test data predictions
display_predictions(5, X_test, y_test, pred1, class_map)
```



EXPERIMENT 1

Random Sampling on Resnet50 pre-trained

```
# Active Learning on pre-trained Resnet50 with 250 new random sampled images from labeled dat

learner_randomsampling = copy.deepcopy(learner)  # Deep copy of original learner

# Number of total instances to be sampled
num_instances = 250

# Random sampling Query
query_idx, query_instance = learner_randomsampling.query(X_sampler, y_sampler, num_instances)
print("ids: ",query_idx, "Instances: ", len(query_instance))

# Teach base model
learner_randomsampling.teach(
```

```
    X=X_sampler[query_idx], y=y_sampler[query_idx], only_new=False,
)
```

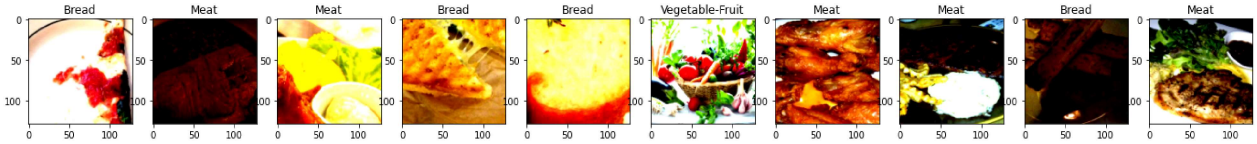
```
ids:  [91, 2170, 372, 261, 888, 1679, 1044, 2095, 1994, 1428, 502, 1872, 2149, 148, 115
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
```

epoch	train_loss	valid_acc	valid_loss	dur
1	0.7266	0.7500	0.6927	30.3187
2	0.7062	0.7500	0.6771	30.5113
3	0.6835	0.7600	0.6608	30.3657
4	0.6611	0.7600	0.6452	30.7584
5	0.6394	0.7700	0.6294	30.5938
6	0.6186	0.7800	0.6143	30.5040
7	0.5988	0.7800	0.5998	30.6114
8	0.5798	0.7900	0.5861	30.7210
9	0.5618	0.7900	0.5734	30.7793
10	0.5448	0.8000	0.5611	30.9169

Printing top 5 and last 5 images from the Queried data

```
print(query_headtail(X_sampler, y_sampler, class_map, 5, 5))
```

```
None
```



Random Sampling Performance

```
pred0 = learner_randomsampling.predict(X_test)
print(classification_report(y_test, pred0))
```

	precision	recall	f1-score	support
0	0.95	0.39	0.56	368
1	0.61	0.99	0.75	432
2	0.98	0.74	0.85	231
accuracy			0.72	1031
macro avg	0.85	0.71	0.72	1031

```
weighted avg       0.82      0.72      0.70      1031
```

EXPERIMENT 2

Initialize modAL for Uncertainty Sampling

```
# initialize ActiveLearner 1 using pre-trained RESNET50 original classifier
learner1 = ActiveLearner(
    estimator = classifier_uncertainity,
    X_training = X, y_training = y
    , query_strategy=uncertainty_sampling
)
```

epoch	train_loss	valid_acc	valid_loss	dur
1	0.9375	0.5800	0.8269	15.1159
2	0.8771	0.7600	0.7890	15.1730
3	0.8390	0.9000	0.7700	15.2002
4	0.8119	0.8800	0.7600	15.1407
5	0.7903	0.8600	0.7521	15.2765
6	0.7713	0.8000	0.7451	15.2772
7	0.7531	0.7800	0.7384	15.0318
8	0.7355	0.7800	0.7314	15.1531
9	0.7181	0.8000	0.7235	15.1058
10	0.7011	0.8400	0.7150	15.1634

Uncertainty Sampling on Resnet50 pre-trained

```
# Copy sampler data
X_sampler_copy = X_sampler.copy()
y_sampler_copy = y_sampler.copy()

# Number of Queries and instances
n_queries = 5
num_instances = 50

# Running Queries
for idx in range(n_queries):
    print('Query no. %d' % (idx + 1))
    # Uncertainty sampling query
    query_idx, query_instance = learner1.query(X_sampler_copy, n_instances=num_instances)
    print("ids: ",query_idx, "Instances: ", len(query_instance))
    print("Printing top and bottom Queried images: \n")
    print(query_headtail(X_sampler_copy[query_idx], y_sampler_copy[query_idx], class_map, 5,
```

```
# Teach base model
learner1.teach(
    X=X_sampler_copy[query_idx], y=y_sampler_copy[query_idx], only_new=False,
)
y_prediction = learner1.predict(X_test)
print("Report after ", num_instances, " new instances:\n", classification_report(y_test,
# remove queried instance from sampler pool
X_sampler_copy = np.delete(X_sampler_copy, query_idx, axis=0)
y_sampler_copy = np.delete(y_sampler_copy, query_idx, axis=0)
print(X_sampler_copy.shape)
```

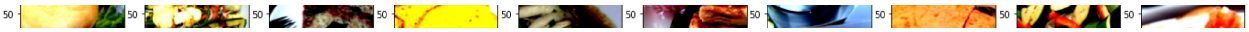
```
Query no. 1
ids:  [1570 1740  823 1054 2206 2202 1040 1422 2167 1634 1038  703 1630  911
  180 1906 1066 1541 2101  971 1364  461  727 1955 2093 1502  680 1530
  304 1394  755  134  873 1902 1648   35  295 1753  404 1459 1984 1086
  175  671 2069  817 1368 1917   91 1501] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.7751       0.8500        0.7139  18.2129
      2        0.7568       0.8667        0.7033  18.0054
      3        0.7445       0.8667        0.6932  18.1492
      4        0.7326       0.8667        0.6831  18.1145
      5        0.7207       0.8667        0.6731  17.8871
      6        0.7090       0.8667        0.6635  17.7211
      7        0.6974       0.8667        0.6541  17.5877
      8        0.6860       0.8667        0.6447  17.7243
      9        0.6748       0.8667        0.6354  17.6982
     10        0.6638       0.8667        0.6262  17.7216
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.95      0.48      0.64       368
           1       0.69      0.94      0.80       432
           2       0.82      0.92      0.87       231

    accuracy                           0.77      1031
   macro avg       0.82      0.78      0.77      1031
weighted avg       0.81      0.77      0.76      1031

(2200, 3, 128, 128)
Query no. 2
ids:  [1335 1385  758 1866  295   78  660  305 1844 1835 1828 1883   68  324
  639   66  253 1200  948  110  570  899  239  545 2039 1921  222 1929
 1556  157 1356  721 1683 2156 1138  709 1566 1032 1045  855 1965 1332
 1153 1680 1047 1380 1338  256 1716 1767] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.7352       0.7857        0.6078  21.3312
      2        0.7327       0.7571        0.6076  21.2800
      3        0.7303       0.7714        0.5981  21.0662
      4        0.7200       0.7714        0.5867  21.4384
      5        0.7080       0.7714        0.5769  21.4085
      6        0.6967       0.7857        0.5666  21.4945
      7        0.6847       0.8000        0.5573  21.4028
      8        0.6733       0.8000        0.5485  21.4899
```

```
      9        0.6622       0.8000        0.5397  21.6291
     10        0.6508       0.8143        0.5303  21.6469
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.68      0.96      0.80       368
           1       0.95      0.63      0.76       432
           2       0.91      0.87      0.89       231

    accuracy                           0.81      1031
   macro avg       0.85      0.82      0.82      1031
weighted avg       0.85      0.81      0.80      1031

(2150, 3, 128, 128)
Query no. 3
ids:  [1703  493 1082 1745 1519 1261  890  541  251 1404 1952  275 1718  294
  379  524 1176 1165   85 1485  827  331 1807 1904 1697 1701 1540   89
 1730  984  262  897  730 1468  447 1325 1262  539  394 1645  315 1480
  361  592  903  745 1208 2115 1251  274] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.7042       0.8750        0.5047  24.2768
      2        0.6597       0.8625        0.5049  24.3597
      3        0.6496       0.8500        0.4978  24.3875
      4        0.6366       0.8625        0.4900  24.1614
      5        0.6236       0.8750        0.4825  24.1209
      6        0.6106       0.8750        0.4755  24.3828
      7        0.5979       0.8750        0.4691  24.2283
      8        0.5856       0.8875        0.4630  24.2762
      9        0.5735       0.8875        0.4572  24.1237
     10        0.5617       0.8875        0.4516  24.1282
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.95      0.59      0.73       368
           1       0.72      0.95      0.82       432
           2       0.91      0.91      0.91       231

    accuracy                           0.81      1031
   macro avg       0.86      0.82      0.82      1031
weighted avg       0.84      0.81      0.81      1031

(2100, 3, 128, 128)
Query no. 4
ids:  [ 847 1883  398 1712 1140 1704  154 2005  823  905  672   74 1779 1531
 1472 1319 1309 1969 1014 1880 1012   44 1557 1299   29 1639 1396  464
  161 1967 1907  551  945 1002  626 1157  842 1863 1208  173 1943 1163
  996  277  437 1868 2038 1511 1677  617] Instances:  50
Printing top and bottom Queried images:

None
```

```
None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6228       0.8667        0.4205  28.0542
      2        0.6063       0.8889        0.4133  27.9640
      3        0.5964       0.8778        0.4068  28.0384
      4        0.5867       0.8778        0.4009  27.8876
      5        0.5772       0.8778        0.3954  27.8228
      6        0.5679       0.8778        0.3902  27.8697
      7        0.5587       0.8778        0.3852  27.7362
      8        0.5497       0.8778        0.3804  28.0648
      9        0.5409       0.8778        0.3759  27.7973
     10        0.5323       0.8667        0.3716  27.8280
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.91      0.78      0.84       368
           1       0.83      0.90      0.86       432
           2       0.89      0.94      0.91       231

    accuracy                           0.87      1031
   macro avg       0.87      0.87      0.87      1031
weighted avg       0.87      0.87      0.87      1031

(2050, 3, 128, 128)
Query no. 5
ids:  [1402 1809  708 1798 1812  715 1516 2040  204  179   10 1013 1950 1245
  735 1870  311 1261  406 1912  688 1163  819  895  420  672 1308 1907
 1354  935 1655  135  261 1364  614  604 1693 1416 1692  101 1565  443
  770  153  527 1440  938 1390   33  578] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6084       0.8800        0.3710  29.7610
      2        0.6013       0.8800        0.3668  29.8029
      3        0.5897       0.8800        0.3636  29.8406
      4        0.5787       0.8800        0.3607  29.8407
      5        0.5684       0.8900        0.3576  29.7868
      6        0.5585       0.8900        0.3549  29.8369
      7        0.5491       0.8800        0.3523  29.6138
      8        0.5400       0.8800        0.3493  29.3628
      9        0.5312       0.8800        0.3465  29.6267
     10        0.5227       0.8800        0.3438  29.5287
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.94      0.74      0.83       368
           1       0.79      0.95      0.86       432
           2       0.96      0.90      0.93       231
```

```
              accuracy                           0.87      1031
             macro avg       0.90      0.87      0.87      1031
          weighted avg       0.88      0.87      0.87      1031

(2000, 3, 128, 128)
```

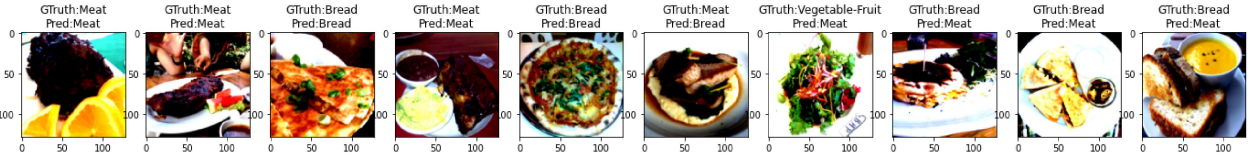
Uncertainty Sampling Performance



```
pred2 = learner1.predict(X_test)
print(classification_report(y_test, pred2))
```

```
                  precision    recall  f1-score   support

               0       0.94      0.74      0.83       368
               1       0.79      0.95      0.86       432
               2       0.96      0.90      0.93       231

        accuracy                           0.87      1031
       macro avg       0.90      0.87      0.87      1031
    weighted avg       0.88      0.87      0.87      1031
```

```
# Display test data predictions
display_predictions(5, X_test, y_test, pred2, class_map)
```



EXPERIMENT 3

Initialize modAL for Entropy Sampling

```
# Initialize ActiveLearner 2 using pre-trained RESNET50 original classifier
learner2 = ActiveLearner(
```

```
    estimator = classifier_entropy,
    X_training = X, y_training = y
    , query_strategy=entropy_sampling
)
```

epoch	train_loss	valid_acc	valid_loss	dur
1	0.9375	0.5800	0.8269	14.5356
2	0.8771	0.7600	0.7890	14.9374
3	0.8390	0.9000	0.7700	14.7878
4	0.8119	0.8800	0.7600	14.7829
5	0.7903	0.8600	0.7521	14.6104
6	0.7713	0.8000	0.7451	14.7384
7	0.7531	0.7800	0.7384	14.6705
8	0.7355	0.7800	0.7314	14.8638
9	0.7181	0.8000	0.7235	14.6466
10	0.7011	0.8400	0.7150	14.7210

Entropy Sampling on Resnet50 pre-trained

```
# Copy sampler data
X_sampler_copy = X_sampler.copy()
y_sampler_copy = y_sampler.copy()

# Number of queries and instances
n_queries = 5
num_instances = 50

# Running Queries
for idx in range(n_queries):
    print('Query no. %d' % (idx + 1))
    # Entropy samping query
    query_idx, query_instance = learner2.query(X_sampler_copy, n_instances=num_instances)
    print("ids: ",query_idx, "Instances: ", len(query_instance))
    print("Printing top and bottom Queried images: \n")
    print(query_headtail(X_sampler_copy[query_idx], y_sampler_copy[query_idx], class_map, 5,
    # Teach base model
    learner2.teach(
        X=X_sampler_copy[query_idx], y=y_sampler_copy[query_idx], only_new=False,
    )
    y_prediction = learner2.predict(X_test)
    print("Report after ", num_instances, " new instances:\n", classification_report(y_test,
    # remove queried instance from pool
    X_sampler_copy = np.delete(X_sampler_copy, query_idx, axis=0)
    y_sampler_copy = np.delete(y_sampler_copy, query_idx, axis=0)
    print(X_sampler_copy.shape)
```

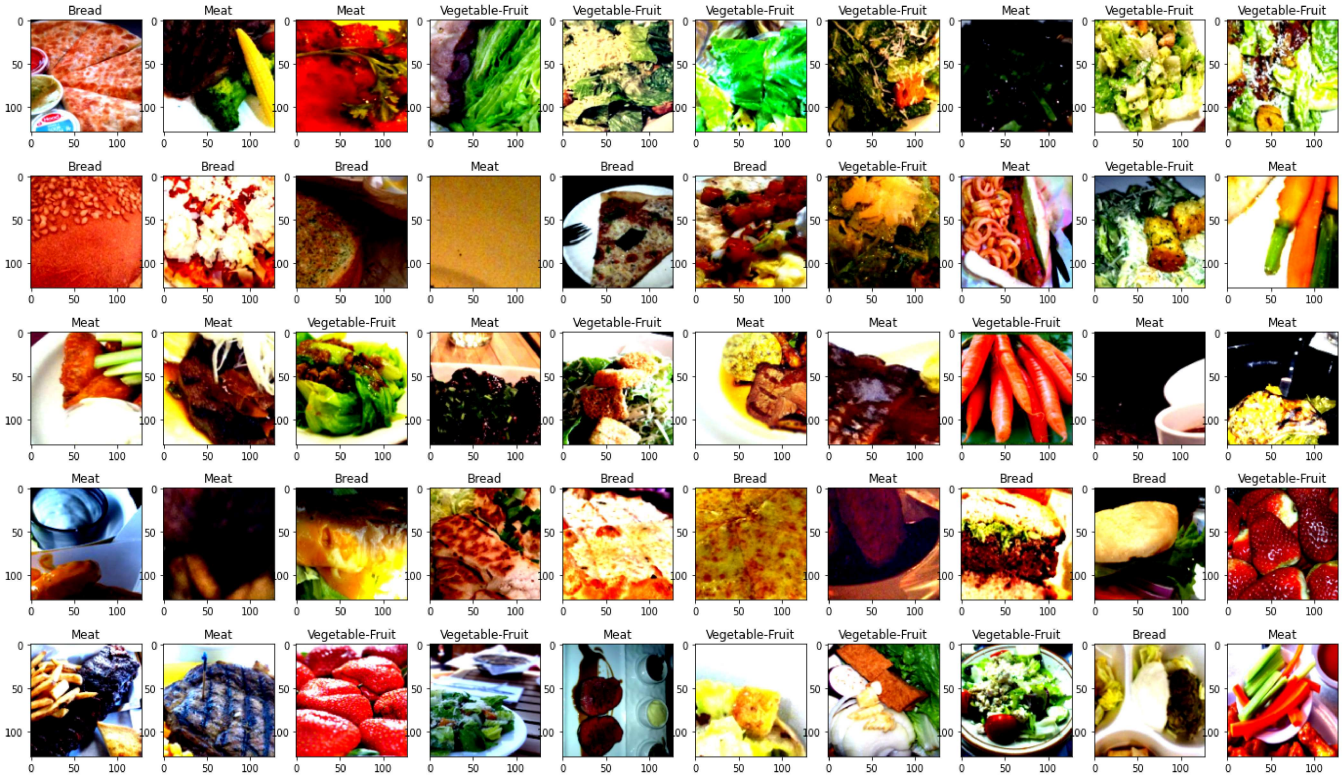
```
Query no. 1
ids:  [1634  727 1502 1459 1917 2167  911 1906 1902 1530 1394 1955  755 1038
 1823 1040   16 1541 2069  180 1054  703  876  971 1422 1570 2202  854
 1753 1066  823 1740 2007 2206 2093 1630 1718 1368  304  680 1633 1364
  298  295 2195 1524  404  886  945 1086] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.7769       0.8500        0.7144  17.8004
      2        0.7593       0.8667        0.7040  17.9130
      3        0.7472       0.8667        0.6941  17.7370
      4        0.7356       0.8667        0.6845  17.7336
      5        0.7240       0.8667        0.6749  17.8402
      6        0.7125       0.8667        0.6652  18.1309
      7        0.7011       0.8667        0.6557  17.8113
      8        0.6898       0.8667        0.6463  17.7133
      9        0.6787       0.8667        0.6371  17.9207
     10        0.6678       0.8667        0.6280  17.8575
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.94      0.49      0.64       368
           1       0.69      0.94      0.80       432
           2       0.82      0.91      0.86       231

    accuracy                           0.77      1031
   macro avg       0.82      0.78      0.77      1031
weighted avg       0.81      0.77      0.76      1031

(2200, 3, 128, 128)
Query no. 2
ids:  [1828 1358 1965  111  762  307    6 1929 1140 1558   78  329  297  733
 1568 1588  773  902 1921 2156 1890  663 1883  256   66 1049 1684  548
  859  242 2104  492  665 1155   34  713  465 1866 1382  225 1047  642
 1844  170  174 1835 1334  739  163 1954] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.7349       0.8714        0.6084  20.7132
      2        0.7261       0.8571        0.6001  20.8109
      3        0.7182       0.8571        0.5887  21.9703
      4        0.7066       0.8857        0.5772  21.0422
      5        0.6942       0.9000        0.5664  20.9914
      6        0.6820       0.9143        0.5563  20.9370
      7        0.6700       0.9143        0.5461  20.9144
      8        0.6577       0.9000        0.5368  20.9972
```

```
      9        0.6462       0.9143        0.5274  20.9920
     10        0.6345       0.9143        0.5187  20.8456
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.74      0.91      0.82       368
           1       0.90      0.72      0.80       432
           2       0.90      0.90      0.90       231

    accuracy                           0.83      1031
   macro avg       0.85      0.85      0.84      1031
weighted avg       0.84      0.83      0.83      1031

(2150, 3, 128, 128)
Query no. 3
ids:  [ 894  880 1747  379 1723 2018  294 1597 1261 1373  540  252  538 1470
  226  766 1720 1487 1934  705   93  427  276 1319 1080 1705  154 1646
 1521  394  860  141   84  331  729 1952 1406 1906  744 1884 1924 1175
  621 1469 1124 1482 1490  696 1803 1326] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6935       0.8625        0.5058  23.5196
      2        0.6660       0.8250        0.5045  23.8849
      3        0.6567       0.8500        0.4967  23.7237
      4        0.6449       0.8500        0.4883  23.8123
      5        0.6329       0.8500        0.4803  23.7810
      6        0.6211       0.8500        0.4725  23.9005
      7        0.6094       0.8500        0.4653  23.6526
      8        0.5979       0.8500        0.4583  23.4743
      9        0.5867       0.8375        0.4517  23.7444
     10        0.5757       0.8375        0.4454  23.6516
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.96      0.60      0.74       368
           1       0.73      0.95      0.83       432
           2       0.90      0.93      0.91       231

    accuracy                           0.82      1031
   macro avg       0.86      0.83      0.82      1031
weighted avg       0.85      0.82      0.81      1031

(2100, 3, 128, 128)
Query no. 4
ids:  [1281  903  881 1907 1475  952  596  844 1881  996  779  399   12 1036
 1717 1399  414  420 1642 1140  675 1324   73 1164 1304 1190   42 1737
 1210  280  156  832 1688 1533 1330 1370 1368  620 1784   27 1884 1002
    9   24  109   43  542  305 1278 1698] Instances:  50
Printing top and bottom Queried images:

None
```

```
None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6232       0.8778        0.4178  26.7180
      2        0.6032       0.8778        0.4115  27.0241
      3        0.5928       0.8778        0.4058  26.9208
      4        0.5826       0.8778        0.4003  26.8576
      5        0.5725       0.8778        0.3951  26.8933
      6        0.5627       0.8778        0.3901  26.8066
      7        0.5530       0.8778        0.3852  26.8848
      8        0.5435       0.8778        0.3805  26.9293
      9        0.5341       0.8778        0.3760  26.8437
     10        0.5250       0.8778        0.3717  26.8632
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.91      0.78      0.84       368
           1       0.82      0.92      0.87       432
           2       0.92      0.93      0.92       231

    accuracy                           0.87      1031
   macro avg       0.88      0.88      0.88      1031
weighted avg       0.88      0.87      0.87      1031

(2050, 3, 128, 128)
Query no. 5
ids:  [ 613 1869 1362  915 1473 1829 1496 2047  920  402  934 1260  407  414
 1806  422 1512  161  257 1627 1988 1351  684 2040  951  480  174 1290
   28 1760  785 1920 1865 1161 1647 1562  776 1702 1904  293 1130 1128
 1599 1949 1025  731  578  593  986   71] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6008       0.8800        0.4181  29.4204
      2        0.6250       0.8800        0.4143  29.3450
      3        0.6135       0.8800        0.4101  29.4477
      4        0.6019       0.8800        0.4068  29.4248
      5        0.5908       0.8800        0.4027  29.3034
      6        0.5796       0.8800        0.3990  29.2125
      7        0.5689       0.8800        0.3949  29.3826
      8        0.5582       0.8600        0.3901  29.2755
      9        0.5474       0.8700        0.3862  29.5191
     10        0.5373       0.8700        0.3816  29.1666
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.94      0.71      0.81       368
           1       0.84      0.87      0.85       432
           2       0.73      0.97      0.83       231
```

```
    accuracy                           0.83      1031
   macro avg       0.84      0.85      0.83      1031
weighted avg       0.85      0.83      0.83      1031
```

```
(2000, 3, 128, 128)
```

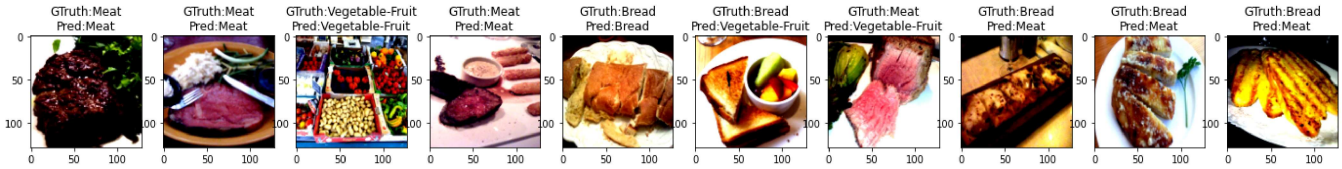


Entropy Sampling Performance

```
y_pred2_2 = learner2.predict(X_test)
print(classification_report(y_test, y_pred2_2))
```

	precision	recall	f1-score	support
0	0.94	0.71	0.81	368
1	0.84	0.87	0.85	432
2	0.73	0.97	0.83	231
accuracy			0.83	1031
macro avg	0.84	0.85	0.83	1031
weighted avg	0.85	0.83	0.83	1031

```
display_predictions(5, X_test, y_test, y_pred2_2, class_map)
```



EXPERIMENT 4

Initialize modAL for Margin Sampling

```
# initialize ActiveLearner 3 using pre-trained RESNET50 original classifier
learner3 = ActiveLearner(
    estimator = classifier_margin,
    X_training = X, y_training = y
    , query_strategy=margin_sampling
)
```

epoch	train_loss	valid_acc	valid_loss	dur
1	0.9375	0.5800	0.8269	14.9222
2	0.8771	0.7600	0.7890	14.7982
3	0.8390	0.9000	0.7700	14.8877
4	0.8119	0.8800	0.7600	14.9304
5	0.7903	0.8600	0.7521	14.9605
6	0.7713	0.8000	0.7451	14.9707
7	0.7531	0.7800	0.7384	14.9900
8	0.7355	0.7800	0.7314	14.9430
9	0.7181	0.8000	0.7235	14.9480
10	0.7011	0.8400	0.7150	14.8245

Margin Sampling on Resnet50 pre-trained

```
# Copy sampler data
X_sampler_copy = X_sampler.copy()
y_sampler_copy = y_sampler.copy()

# Number of Queries and instances
n_queries = 5
num_instances = 50

# Running Queries
for idx in range(n_queries):
    print('Query no. %d' % (idx + 1))
    # Margin sampling Query
    query_idx, query_instance = learner3.query(X_sampler_copy, n_instances = num_instances)
    print("ids: ",query_idx, "Instances: ", len(query_instance))
    print("Printing top and bottom Queried images: \n")
    print(query_headtail(X_sampler_copy[query_idx], y_sampler_copy[query_idx], class_map, 5,
    # Teach base model
    learner3.teach(
        X=X_sampler_copy[query_idx], y=y_sampler_copy[query_idx], only_new=False,
    )
    y_prediction = learner3.predict(X_test)
    print("Report after ", num_instances, " new instances:\n", classification_report(y_test,
    # remove queried instance from pool
    X_sampler_copy = np.delete(X_sampler_copy, query_idx, axis=0)
    y_sampler_copy = np.delete(y_sampler_copy, query_idx, axis=0)
    print(X_sampler_copy.shape)
```

```
Query no. 1
ids:  [ 700 1878 1929 1038  200 1791 1597 1835 2220  484 1648 1194 2038  388
  369 1461  373 1333 2019  296 1972 1040  616 1321  643 1553  821  384
 1167 1515  285 1387  461  967   12 1643 1634  911 2101  428  763  180
 1688  554 1898 1564 1910 1984 1068  118] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.7374       0.8667        0.7349  17.5197
      2        0.7162       0.8667        0.7221  17.7790
      3        0.7013       0.8833        0.7112  17.7133
      4        0.6878       0.8833        0.7008  17.6944
      5        0.6746       0.8833        0.6907  17.5841
      6        0.6616       0.8833        0.6808  17.7433
      7        0.6489       0.8833        0.6710  17.6901
      8        0.6363       0.8833        0.6615  17.4559
      9        0.6239       0.8833        0.6521  17.6547
     10        0.6118       0.8833        0.6428  17.7647
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.84      0.85      0.84       368
           1       0.86      0.87      0.86       432
           2       0.91      0.88      0.89       231

    accuracy                           0.86      1031
   macro avg       0.87      0.87      0.87      1031
weighted avg       0.86      0.86      0.86      1031

(2200, 3, 128, 128)
Query no. 2
ids:  [ 569 1441 1417 1043 1061 2197  192 1874 1599 2021  719 1681 1864   41
 2166  636  608  576  114  886 1329 1772 1367  976 2190  783 1540  311
 1584  745  881  734 1379  916  611  623  631 2058  902  236  832  487
 1979  178 1025  248 2146 1647  710 2060] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6637       0.6571        0.6622  20.5377
      2        0.6821       0.6286        0.6759  20.5511
      3        0.6856       0.6286        0.6687  20.5884
      4        0.6761       0.6286        0.6589  20.4439
      5        0.6645       0.6571        0.6494  20.6948
      6        0.6530       0.6714        0.6405  20.6633
      7        0.6417       0.6714        0.6323  20.3725
      8        0.6307       0.6857        0.6245  20.5627
```

```
      9        0.6199       0.6857        0.6173  20.6097
     10        0.6095       0.7143        0.6104  20.5680
Report after  50  new instances:
              precision    recall  f1-score   support

           0       1.00      0.24      0.38       368
           1       0.55      1.00      0.71       432
           2       0.99      0.70      0.82       231

    accuracy                           0.66      1031
   macro avg       0.85      0.64      0.64      1031
weighted avg       0.81      0.66      0.62      1031

(2150, 3, 128, 128)
Query no. 3
ids:  [ 915   69 1666 1268 1897  671 1889 1696 1526  413  718 1966  720  324
  721 1882  742 1386  753 1036 1017 1465  814 1853  592 1005  846  146
  923 1816 2097 1226 1138  649 1451 1843  715 1800  941 1433  265 1224
 1625 1375 1579 1991 1825  843  377 1560] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.6054       0.8500        0.5303  23.6370
      2        0.5552       0.8375        0.5273  23.7972
      3        0.5501       0.8500        0.5163  23.8538
      4        0.5391       0.8625        0.5054  23.7607
      5        0.5278       0.8625        0.4949  23.4926
      6        0.5164       0.8750        0.4857  23.8866
      7        0.5056       0.8750        0.4770  23.6132
      8        0.4950       0.8875        0.4689  23.6835
      9        0.4847       0.8750        0.4611  23.8080
     10        0.4746       0.8750        0.4538  23.7852
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.80      0.94      0.86       368
           1       0.92      0.80      0.86       432
           2       0.95      0.90      0.92       231

    accuracy                           0.87      1031
   macro avg       0.89      0.88      0.88      1031
weighted avg       0.88      0.87      0.87      1031

(2100, 3, 128, 128)
Query no. 4
ids:  [1392 1841 1720  392 1872 1691  747  298 1238 1921 1289 1588  159  361
  893  457  665  911 1325 1552 1524  764  414 1655 1744    2 1611 1353
  739  535  459 1577  550 1907  306 1919 1966 1766   95 2059 1264   86
  612 1015  426 1539 1509 1881  630  829] Instances:  50
Printing top and bottom Queried images:

None
```

```
None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.5082       0.8778        0.4227  26.5618
      2        0.4756       0.8778        0.4165  27.0309
      3        0.4647       0.8778        0.4101  27.0845
      4        0.4548       0.8778        0.4040  26.8320
      5        0.4452       0.8778        0.3982  26.8383
      6        0.4359       0.8778        0.3926  27.2113
      7        0.4268       0.8778        0.3873  26.8073
      8        0.4180       0.8889        0.3822  26.8725
      9        0.4094       0.8889        0.3774  26.8150
     10        0.4011       0.8889        0.3727  26.8645
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.90      0.82      0.86       368
           1       0.84      0.92      0.88       432
           2       0.94      0.90      0.92       231

    accuracy                           0.88      1031
   macro avg       0.89      0.88      0.89      1031
weighted avg       0.88      0.88      0.88      1031

(2050, 3, 128, 128)
Query no. 5
ids:  [1131   96 1489  109  996 1503  135 1424  547 1921  977 1910  517  735
  738   79  743  755 1377 1105 1375 2031 1644 1208  773 1367 1653 1657
   76  229  239   64 1358   33 1357  421  419  281 1798 1270 1137  316
 1718  376  826 1297  658   50 1307 1913] Instances:  50
Printing top and bottom Queried images:

None
Re-initializing module.
Re-initializing criterion.
Re-initializing optimizer.
  epoch    train_loss    valid_acc    valid_loss      dur
-------  ------------  -----------  ------------  -------
      1        0.4597       0.8200        0.4370  30.1028
      2        0.4697       0.8200        0.4336  29.9555
      3        0.4598       0.8100        0.4294  30.1222
      4        0.4498       0.8000        0.4253  30.2353
      5        0.4401       0.8000        0.4216  30.2481
      6        0.4310       0.8000        0.4168  30.1675
      7        0.4220       0.8000        0.4129  30.0994
      8        0.4135       0.8000        0.4092  30.2309
      9        0.4054       0.8000        0.4057  30.1239
     10        0.3975       0.8000        0.4016  30.1725
Report after  50  new instances:
              precision    recall  f1-score   support

           0       0.97      0.61      0.75       368
           1       0.70      0.99      0.82       432
           2       0.98      0.82      0.89       231
```

```
    accuracy                           0.81      1031
   macro avg       0.89      0.80      0.82      1031
weighted avg       0.86      0.81      0.81      1031
```

```
(2000, 3, 128, 128)
```

Margin Sampling Performance

```
y_pred3_2 = learner3.predict(X_test)
print(classification_report(y_test, y_pred3_2))
```

```
              precision    recall  f1-score   support

           0       0.97      0.61      0.75       368
           1       0.70      0.99      0.82       432
           2       0.98      0.82      0.89       231

    accuracy                           0.81      1031
   macro avg       0.89      0.80      0.82      1031
weighted avg       0.86      0.81      0.81      1031
```

```
display_predictions(5, X_test, y_test, y_pred3_2, class_map)
```