Open in Colab

In [61]:
```
# Shubham Patel - Student ID : 916193675
```

In [62]:
```
!pip install timm
```

```
Requirement already satisfied: timm in /usr/local/lib/python3.10/dist-packag
es (1.0.12)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packa
ges (from timm) (2.5.1+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist
-packages (from timm) (0.20.1+cu121)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-pack
ages (from timm) (6.0.2)
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.10/
dist-packages (from timm) (0.26.5)
Requirement already satisfied: safetensors in /usr/local/lib/python3.10/dist
-packages (from timm) (0.4.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pa
ckages (from huggingface_hub->timm) (3.16.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
0/dist-packages (from huggingface_hub->timm) (2024.10.0)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/
dist-packages (from huggingface_hub->timm) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-pa
ckages (from huggingface_hub->timm) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dis
t-packages (from huggingface_hub->timm) (4.66.6)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/
python3.10/dist-packages (from huggingface_hub->timm) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-pa
ckages (from torch->timm) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pack
ages (from torch->timm) (3.1.4)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/di
st-packages (from torch->timm) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.
10/dist-packages (from sympy==1.13.1->torch->timm) (1.3.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packa
ges (from torchvision->timm) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/pytho
n3.10/dist-packages (from torchvision->timm) (11.0.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/
dist-packages (from jinja2->torch->timm) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/py
thon3.10/dist-packages (from requests->huggingface_hub->timm) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dis
t-packages (from requests->huggingface_hub->timm) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.
10/dist-packages (from requests->huggingface_hub->timm) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.
10/dist-packages (from requests->huggingface_hub->timm) (2024.8.30)
```

In [63]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

In [64]:
```
import pandas as pd
import cv2
import os
import numpy as np
import warnings
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, SpectralClustering, BisectingKMeans
from sklearn.cluster import DBSCAN, AgglomerativeClustering
from sklearn.metrics import fowlkes_mallows_score, silhouette_score


import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader


import timm

import albumentations as A
from albumentations.pytorch import ToTensorV2
```

In [65]:
```
warnings.filterwarnings('ignore')
```

1. Feature Extraction

In [66]:
```
path = "/content/drive/MyDrive/DM/updated"
```

In [67]:
```
class_names = ['Malamute','Kerry_blue_terrier','German_short-haired_pointer'

actual_classes = ['n02110063-malamute','n02093859-Kerry_blue_terrier','n0210
```

In [68]:
```
df = pd.DataFrame(columns = ['image_id','label'])
for class_ in actual_classes:
  class_path = os.path.join(path,class_)
  for filename in os.listdir(class_path):
    df.loc[len(df)] = [filename,actual_classes.index(class_)]
```

In [69]:
```
df.head()
```

Out[69]:

	image_id	label
0	n02110063_13152.jpg	0
1	n02110063_6276.jpg	0
2	n02110063_17073.jpg	0
3	n02110063_609.jpg	0
4	n02110063_15580.jpg	0

In [70]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

In [71]:
```
class ImageData(Dataset):


    def __init__(self, data, directory, transform,actual_classes):
        self.data = data
        self.directory = directory
        self.transform = transform
        self.actual_classes = actual_classes


    def __len__(self):
        return len(self.data)


    def __getitem__(self, idx):


        path = os.path.join(self.directory,actual_classes[self.data.iloc[idx
        image = cv2.imread(
            os.path.join(path, self.data.iloc[idx]["image_id"])
        )
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        image = self.transform(image=image)["image"]

        return image
```

In [72]:
```
transforms = A.Compose([A.Resize(height=128, width=128), A.Normalize(), ToTe

data_set = ImageData(
    data=df,
    directory=path,
    transform=transforms,
    actual_classes=actual_classes
)

data_loader = DataLoader(data_set, batch_size=32, shuffle=False, num_workers
```

In [73]:
```
model = timm.create_model(model_name="resnet18", pretrained=True)
model.fc = nn.Linear(512, 4)
```

```
model.to(device)
```

Out[73]:
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
  (act1): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil
_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (act2): ReLU(inplace=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (act2): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (act2): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
```

```
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (act2): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (act2): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=Fals
e)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (act2): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=
```

```
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (act2): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=Fals
e)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (drop_block): Identity()
      (act1): ReLU(inplace=True)
      (aa): Identity()
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_r
unning_stats=True)
      (act2): ReLU(inplace=True)
    )
  )
  (global_pool): SelectAdaptivePool2d(pool_type=avg, flatten=Flatten(start_
dim=1, end_dim=-1))
  (fc): Linear(in_features=512, out_features=4, bias=True)
)
```

In [74]:
```
def get_features(name):
    def hook(model, input, output):
        features[name] = output.detach()

    return hook
model.global_pool.register_forward_hook(get_features("feats"))
```

Out[74]:
```
<torch.utils.hooks.RemovableHandle at 0x7fed881d7040>
```

In [75]:
```
PREDS = []
FEATS = []

features = {}

for idx, inputs in enumerate(data_loader):

    inputs = inputs.to(device)
```

```
    preds = model(inputs)

    PREDS.append(preds.detach().cpu().numpy())
    FEATS.append(features["feats"].cpu().numpy())
```

In [76]:
```
features_df = pd.DataFrame(columns=list(range(512)))
```

In [77]:
```
for i in range(len(FEATS)):
  for j in range(len(FEATS[i])):
    features_df.loc[len(features_df)] = list(FEATS[i][j])
```

In [78]:
```
features_df['label'] = df['label']
```

2. Dimension Reduction

In [79]:
```
pca = PCA(n_components=2)

transformed_data = pca.fit_transform(features_df.drop('label', axis=1))
```

In [80]:
```
transformed_data
```

Out[80]:
```
array([[-2.032571  ,  4.191373  ],
       [-1.6878549 ,  1.7816083 ],
       [-0.74498135, -1.8623611 ],
       ...,
       [-0.6238642 ,  1.2101009 ],
       [-0.00475764,  1.4298079 ],
       [-2.19      , -0.01429355]], dtype=float32)
```

3. Clustering Algorithm

In [81]:
```
# K-means clustering with 'random'
kmeans_random = KMeans(n_clusters=4, init='random', random_state=20)
kmeans_random.fit(transformed_data)
kmeans_random_labels = kmeans_random.labels_

# K-means clustering with 'k-means++'
kmeans_kmeans_pp = KMeans(n_clusters=4, init='k-means++', random_state=20)
kmeans_kmeans_pp.fit(transformed_data)
kmeans_kmeans_pp_labels = kmeans_kmeans_pp.labels_

# Bisecting K-means clustering with 'random'
bisecting_kmeans_random = BisectingKMeans(n_clusters=4, init='random', randc
bisecting_kmeans_random.fit(transformed_data)
bisecting_kmeans_random_labels = bisecting_kmeans_random.labels_

# Spectral clustering with default parameters
spectral_clustering = SpectralClustering(n_clusters=4, random_state=20)
spectral_clustering.fit(transformed_data)
spectral_clustering_labels = spectral_clustering.labels_
```

In [91]:
```
# DBSCAN
dbscan = DBSCAN(eps=0.4, min_samples=10)
dbscan.fit(transformed_data)
dbscan_labels = dbscan.labels_

# Agglomerative clustering with different linkage methods
agglomerative_single = AgglomerativeClustering(n_clusters=4, linkage='single
agglomerative_single.fit(transformed_data)
agglomerative_single_labels = agglomerative_single.labels_

agglomerative_complete = AgglomerativeClustering(n_clusters=4, linkage='comp
agglomerative_complete.fit(transformed_data)
agglomerative_complete_labels = agglomerative_complete.labels_

agglomerative_average = AgglomerativeClustering(n_clusters=4, linkage='avera
agglomerative_average.fit(transformed_data)
agglomerative_average_labels = agglomerative_average.labels_

agglomerative_ward = AgglomerativeClustering(n_clusters=4, linkage='ward')
agglomerative_ward.fit(transformed_data)
agglomerative_ward_labels = agglomerative_ward.labels_
```

In [93]:
```
np.unique(dbscan_labels)

# eps is equal to 0.4 and min_samples parameter is 10 to get 4 clusters
```

Out[93]:
```
array([-1,  0,  1,  2])
```

4. Clustering Evaluations

In [94]:
```
orginal_labels = features_df['label']
```

In [95]:
```
# Calculate Fowlkes-Mallows index
fowlkes_mallows_scores = {
    'K-means (Random)': fowlkes_mallows_score(orginal_labels, kmeans_random_
    'K-means (k-means++)': fowlkes_mallows_score(orginal_labels, kmeans_kmea
    'Bisecting K-means': fowlkes_mallows_score(orginal_labels, bisecting_kme
    'Spectral Clustering': fowlkes_mallows_score(orginal_labels, spectral_cl
    'DBSCAN': fowlkes_mallows_score(orginal_labels, dbscan_labels),
    'Agglomerative (Single link- MIN)': fowlkes_mallows_score(orginal_labels
    'Agglomerative (Complete link- MAX)': fowlkes_mallows_score(orginal_labe
    'Agglomerative (Group Average)': fowlkes_mallows_score(orginal_labels, a
    'Agglomerative (Ward)': fowlkes_mallows_score(orginal_labels, agglomerat
}
```

In [96]:
```
# Calculate Silhouette Coefficient
silhouette_scores = {
    'K-means (Random)': silhouette_score(transformed_data, kmeans_random_lab
    'K-means (k-means++)': silhouette_score(transformed_data, kmeans_kmeans_
    'Bisecting K-means': silhouette_score(transformed_data, bisecting_kmeans
    'Spectral Clustering': silhouette_score(transformed_data, spectral_clust
    'DBSCAN': silhouette_score(transformed_data, dbscan_labels),
```

```
    'Agglomerative (Single link-MIN)': silhouette_score(transformed_data, ag
    'Agglomerative (Complete link-MAX)': silhouette_score(transformed_data,
    'Agglomerative (Group Average)': silhouette_score(transformed_data, aggl
    'Agglomerative (Ward)': silhouette_score(transformed_data, agglomerative
}
```

In [97]:
```
# Rank methods based on Fowlkes-Mallows index
ranked_methods_fm = sorted(fowlkes_mallows_scores.items(), key=lambda x: x[1
print("Ranking based on Fowlkes-Mallows index:")
for method, score in ranked_methods_fm:
    print(f"{method}: {score}")
```

```
Ranking based on Fowlkes-Mallows index:
Agglomerative (Single link- MIN): 0.4982544375031002
Spectral Clustering: 0.3855319029938198
DBSCAN: 0.3755390768771918
Agglomerative (Complete link- MAX): 0.36929803127961575
Agglomerative (Group Average): 0.35085569553892687
K-means (k-means++): 0.26458420928708204
K-means (Random): 0.2598071716208636
Bisecting K-means: 0.2589423885177832
Agglomerative (Ward): 0.2571493780301932
```

In [98]:
```
# Rank methods based on Silhouette Coefficient
ranked_methods_silhouette = sorted(silhouette_scores.items(), key=lambda x:
print("\nRanking based on Silhouette Coefficient:")
for method, score in ranked_methods_silhouette:
    print(f"{method}: {score}")
```

```
Ranking based on Silhouette Coefficient:
Agglomerative (Single link-MIN): 0.39202287793159485
Spectral Clustering: 0.39007893204689026
K-means (k-means++): 0.36184126138687134
Agglomerative (Group Average): 0.3497146666049957
K-means (Random): 0.33136042952537537
Bisecting K-means: 0.3300669193267822
Agglomerative (Ward): 0.3030272424221039
Agglomerative (Complete link-MAX): 0.2649152874946594
DBSCAN: -0.045270971953868866
```