Image feature Extraction Transfer Learning

* Tapan Kumar Patro
* tapankumarpatro05@gmail.com

Work Items:

1.Image Load
2.Feature Extraction for all Image
3.Save into A pickel file
4.Load the pickel file
5.Extract feature from given image
6.Find the minimum distance and return the images

Importing Libraries

```
from __future__ import absolute_import, division, print_function, unicode_litera
# from google_images_download import google_images_download

try:
  # The %tensorflow_version magic only works in colab.
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf

import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
tf.__version__
```

```
'2.3.0'
```

```
import os
from tqdm import tqdm
from tensorflow.keras import models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import Dense,AveragePooling2D,BatchNormalization,Co
```

```
from time import time
from datetime import datetime
from tensorflow.python.keras.callbacks import TensorBoard
import cv2
```

Setup Google Colab for importing Dataset

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_

Enter your authorization code:
··········
Mounted at /content/drive
```

```
# Navigating to Dataset folder in my drive
path = 'drive/My Drive/PocketApps/Avantari/dataset'
os.chdir(path)

```

```
!pwd
```

```
/content/drive/My Drive/PocketApps/Avantari/dataset
```

Clustring

```
# # Navigating to Dataset folder in my drive
# path = 'drive/My Drive/PocketApps/Avantari'
# os.chdir(path)
# %cd ..
```

```
import pickle

```

```
# Loading pickle file

def load_stuff(filename):
    saved_stuff = open(filename, "rb")
    stuff = pickle.load(saved_stuff)
    saved_stuff.close()
    return stuff

precompute_features = load_stuff("precompute_img_features.pickle")
```

```
# Collecting all features for Clustering
```

```
z = pd.DataFrame(Y.tolist()) # a list
# Fit the model using t-SNE randomized algorithm
digits_proj = TSNE(random_state=25111993).fit_transform(all_features)
```

```
# An user defined function to create scatter plot of features
def scatter(x, colors):
    # We choose a color palette with seaborn.
    palette = np.array(sns.color_palette("hls", 18))

    # We create a scatter plot.
    f = plt.figure(figsize=(32, 32))
    ax = plt.subplot(aspect='equal')
    sc = ax.scatter(x[:,0], x[:,1], lw=0, s=120,
                    c=palette[colors.astype(np.int)])

    ax.axis('off')
    ax.axis('tight')

    # We add the labels for each cluster.
    txts = []
    for i in range(18):
        # Position of each label.
        xtext, ytext = np.median(x[colors == i, :], axis=0)
        txt = ax.text(xtext, ytext, str(i), fontsize=50)
        txt.set_path_effects([
            PathEffects.Stroke(linewidth=5, foreground="w"),
            PathEffects.Normal()])
        txts.append(txt)

    return f, ax, sc, txts
```

```
# Ploting the graph.
print(list(range(0,18)))
sns.palplot(np.array(sns.color_palette("hls", 18)))
scatter(digits_proj, Y)
plt.savefig('animal_cluster_.png', dpi=120)

```

```
# Collecting all features for Clustering
all_features = []

for each_image_data in precompute_features:
    image_feature = each_image_data.get("features")
    all_features.append(image_feature[0])

```

```
import numpy as np
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.manifold import TSNE
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.patheffects as PathEffects
import shutil
from tqdm import tqdm
```

```

```

```
# Clustering with K Means with 10 clusters
kmeans = KMeans(n_clusters=10, random_state=0).fit(np.array(all_features))

```

```
!pwd
```

```
all_images = os.listdir()
```

```
all_images[:3]
```

```
['4490.jpg', '1642.jpg', '1390.jpg']
```

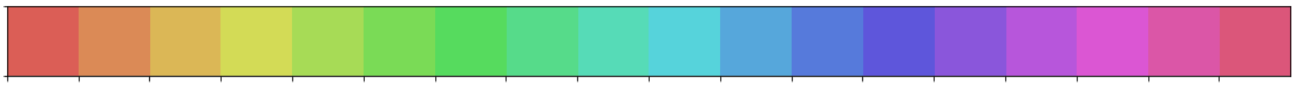
```
# categorizing images into clusters

print("\n")
for i, m in tqdm(enumerate(kmeans.labels_)):
    print("    Copy: %s / %s" %(i, len(kmeans.labels_)), end="\r")
    shutil.copy(all_images[i], str(m) + "_" + str(i) + ".jpg")
```

```
6.37it/s]28it [00:03,  5.52it/s]30it [00:03,  4.98it/s]32it [00:04,  4.82it/s]
.80it/s]39it [00:05,  4.67it/s]
.40it/s]43it [00:06,  4.28it/s]
```

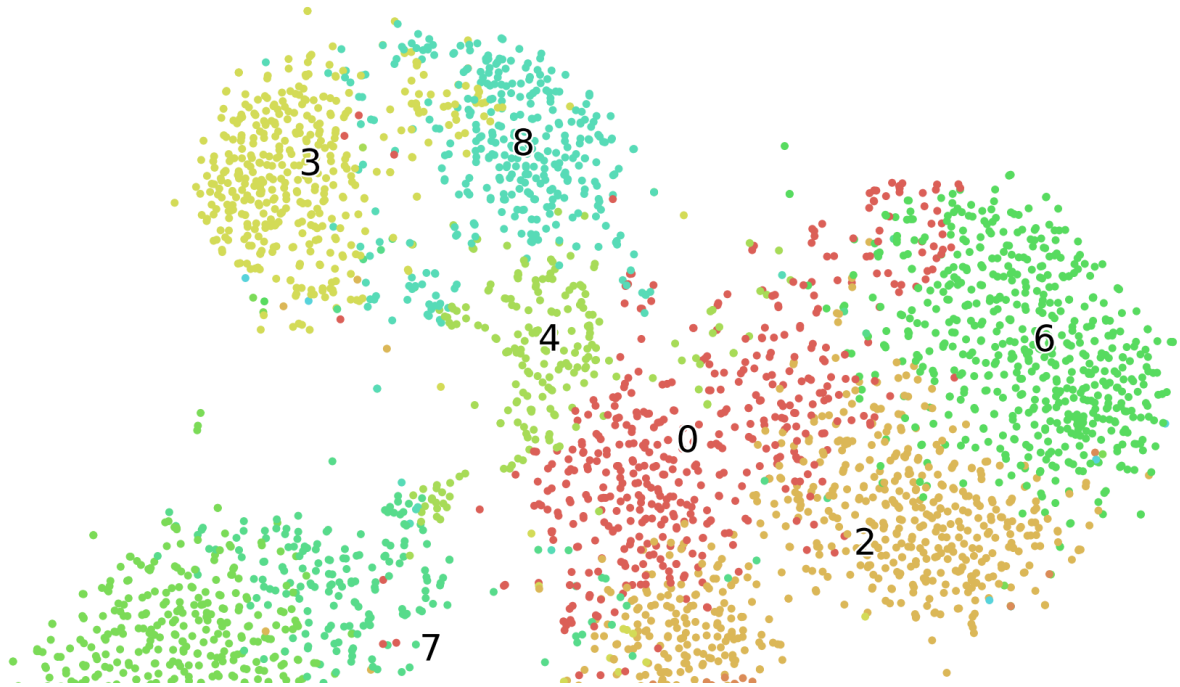
```

```

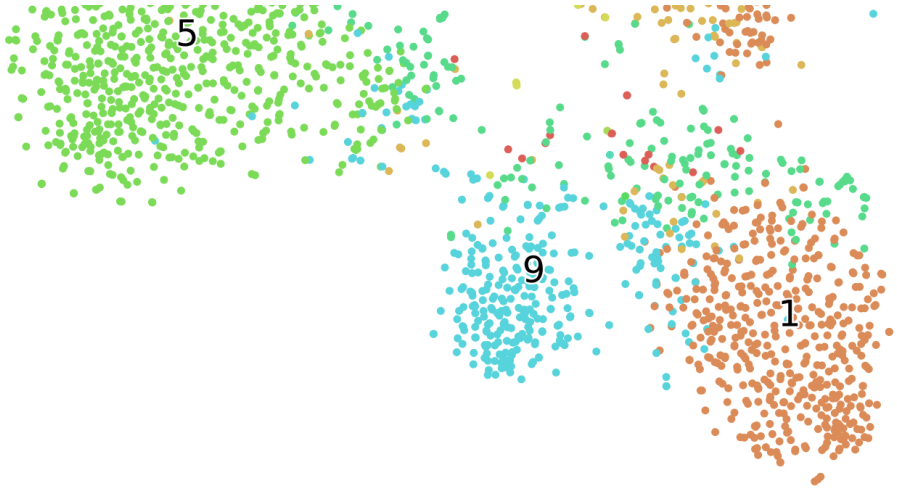
```
# Labes
Y=kmeans.labels_
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
/usr/local/lib/python3.6/dist-packages/numpy/core/fromnumeric.py:3335: Runtime
  out=out, **kwargs)
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:154: RuntimeWarn
  ret, rcount, out=ret, casting='unsafe', subok=False)
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```



```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```






```
cluster1 = []
cluster5 = []
```

Some Image Samples from Clusters

```
from PIL import Image
```

```
all_images_cluster = os.listdir()
```

```
for img_name in all_images_cluster:
    if "1_" in img_name:
        cluster1.append(img_name)
    elif "5_" in img_name:
        cluster5.append(img_name)
    else:
        pass
```

```
        pass
```

```
1 cluster1[:3]
```

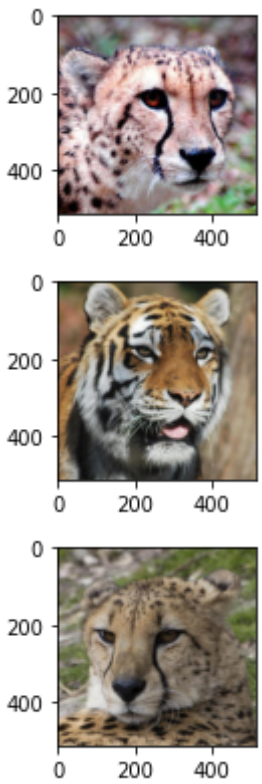
```
['1_42.jpg', '1_115.jpg', '1_123.jpg']
```

```
1 cluster5[:3]
```

```
['5_3.jpg', '5_5.jpg', '5_6.jpg']
```

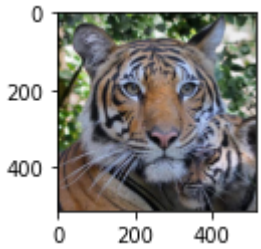
Ploting 3 images from Cluster 1

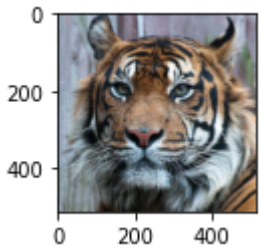
```
1 for i, val in enumerate(cluster1[:3]):
2     plt.subplot(1, 3, i+1)
3     image_data = Image.open(val)
4     plt.imshow(image_data)
5     plt.show()
```

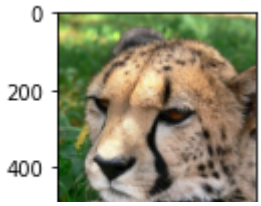


Ploting 3 images from Cluster 5

```
1 for i, val in enumerate(cluster5[7:10]):
2     plt.subplot(1, 3, i+1)
3     image_data = Image.open(val)
4     plt.imshow(image_data)
5     plt.show()
```







1

1

1

1

1