

Module 3 (generating Image embedding)

In []:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7" --header="Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kaggle-data-sets/930393/1613771/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20210408%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210408T071356Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-Signature=543bf16d4aacd031137a27ce91ab974dfc2883d1180c78f2a0a7c7eb3a912101c9445b3d93ca648e33533d76a1a0e6b27c03c6ad0898605125f0616fd4d3aaff7219951948e3b319af1f87436130bc6a2bbcdf25c2337c7960bf33ef7dea86a0175e9eac39d59e76d9c4281b0a0e8f569d942487493762bb239bd6da3eb1322a481bcff60e0b9efeffdce022fcc9dbf9cd7d1c01bb8041c27fbdcae2a6a6c891159156bcc3531a18c2076a0ae790682c3c95345d9b6dae742c78ba0515d6f4525a2e7284917779b993e8a8a2619cfbe17516caa69c9abbd3e739f526a63359f7aac0e382f6c86171057c2d82c8c83b0847b75ad2d1e20f4b19a673a31b42966" -c -O 'archive.zip'
```

```
--2021-04-08 07:15:32-- https://storage.googleapis.com/kaggle-data-sets/930393/1613771/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20210408%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210408T071356Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-Signature=543bf16d4aacd031137a27ce91ab974dfc2883d1180c78f2a0a7c7eb3a912101c9445b3d93ca648e33533d76a1a0e6b27c03c6ad0898605125f0616fd4d3aaff7219951948e3b319af1f87436130bc6a2bbcdf25c2337c7960bf33ef7dea86a0175e9eac39d59e76d9c4281b0a0e8f569d942487493762bb239bd6da3eb1322a481bcff60e0b9efeffdce022fcc9dbf9cd7d1c01bb8041c27fbdcae2a6a6c891159156bcc3531a18c2076a0ae790682c3c95345d9b6dae742c78ba0515d6f4525a2e7284917779b993e8a8a2619cfbe17516caa69c9abbd3e739f526a63359f7aac0e382f6c86171057c2d82c8c83b0847b75ad2d1e20f4b19a673a31b42966
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.164.176, 172.217.164.144, 172.217.12.240, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.164.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11894512126 (11G) [application/zip]
Saving to: 'archive.zip'
```

```
archive.zip          100%[=====>]  11.08G  82.1MB/s   in 4m 24s
```

```
2021-04-08 07:19:57 (42.9 MB/s) - 'archive.zip' saved [11894512126/11894512126]
```

```
time: 4min 25s (started: 2021-04-08 07:15:32 +00:00)
```

Applications link : https://www.tensorflow.org/api_docs/python/tf/keras/applications

In []:

```
!mkdir data
!unzip -q '/content/archive.zip' -d '/content/data'
!rm -rf '/content/archive.zip'
```

```
time: 5min 39s (started: 2021-04-08 07:21:21 +00:00)
```

In []:

```
import tensorflow as tf
import cv2
import numpy as np
```

```
time: 7.74 s (started: 2021-04-08 07:32:24 +00:00)
```

DenseNet121

In []:

```

tf.keras.backend.clear_session()

model_embedding = tf.keras.applications.DenseNet121(include_top=False, weights='imagenet',
input_tensor=None, input_shape=(520,520,3),pooling=None,)
def load_img(path):
    # Reading an image
    image = cv2.imread(path)
    # resizing because pre-trained model image shape is 520x520
    image = cv2.resize(image, (520,520),interpolation=cv2.INTER_AREA)
    # Converting to RGB because it will be saved as a correct image even if it is saved after being converted to a PIL
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    # Preprocessed numpy.array or a tf.Tensor with type float32.
    image = tf.image.convert_image_dtype(image,tf.float32)[tf.newaxis, ...]
    return image

def get_embeddings(path):
    image = load_img(path)
    # Returns 1024 dimension array/ vector with predicted values
    img_embedding = model_embedding.predict(image,steps=1)
    # Removes dimensions of size 1 from the shape of a tensor.
    img_embedding = tf.squeeze(img_embedding, axis=None, name=None)
    # Computes the mean of elements across dimensions of a tensor. [ Normalize ]
    img_embedding = tf.reduce_mean(img_embedding, axis=(0,1), keepdims=False, name=None).numpy()
    # Converting to List
    img_embedding = img_embedding.tolist()
    return img_embedding

DenseNet121_embeddings = []
dir = r'/content/data/women_boots'
for filename in os.listdir(dir):
    if filename.endswith(".jpg"):
        DenseNet121_embeddings.append(get_embeddings(os.path.join(dir,filename)))

```

time: 8min 25s (started: 2021-04-08 07:59:28 +00:00)

ResNet50

In []:

```

tf.keras.backend.clear_session()

model_embedding = tf.keras.applications.ResNet50(include_top=False, weights='imagenet',
input_tensor=None, input_shape=(520,520,3), pooling=None,)
def load_img(path):
    # Reading an image
    image = cv2.imread(path)
    # resizing because pre-trained model image shape is 520x520
    image = cv2.resize(image, (520,520),interpolation=cv2.INTER_AREA)
    # Converting to RGB because it will be saved as a correct image even if it is saved after being converted to a PIL
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    # Preprocessed numpy.array or a tf.Tensor with type float32.
    image = tf.image.convert_image_dtype(image,tf.float32)[tf.newaxis, ...]
    return image

def get_embeddings(path):
    image = load_img(path)
    # Returns 1024 dimension array/ vector with predicted values
    img_embedding = model_embedding.predict(image,steps=1)
    # Removes dimensions of size 1 from the shape of a tensor.
    img_embedding = tf.squeeze(img_embedding, axis=None, name=None)
    # Computes the mean of elements across dimensions of a tensor. [ Normalize ]
    img_embedding = tf.reduce_mean(img_embedding, axis=(0,1), keepdims=False, name=None).numpy()
    # Converting to List
    img_embedding = img_embedding.tolist()
    return img_embedding

```

```

ResNet50_embeddings = []
dir = r'/content/data/women_boots'
for filename in os.listdir(dir):
    if filename.endswith(".jpg"):
        ResNet50_embeddings.append(get_embeddings(os.path.join(dir, filename)))

```

time: 8min 52s (started: 2021-04-08 08:08:09 +00:00)

ResNet101

In []:

```

tf.keras.backend.clear_session()

model_embedding = tf.keras.applications.ResNet101(include_top=False, weights='imagenet',
input_tensor=None, input_shape=(520,520,3), pooling=None,)
def load_img(path):
    # Reading an image
    image = cv2.imread(path)
    # resizing because pre-trained model image shape is 520x520
    image = cv2.resize(image, (520,520), interpolation=cv2.INTER_AREA)
    # Converting to RGB because it will be saved as a correct image even if it is saved after being converted to a PIL
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Preprocessed numpy.array or a tf.Tensor with type float32.
    image = tf.image.convert_image_dtype(image, tf.float32)[tf.newaxis, ...]
    return image

def get_embeddings(path):
    image = load_img(path)
    # Returns 1024 dimension array/ vector with predicted values
    img_embedding = model_embedding.predict(image, steps=1)
    # Removes dimensions of size 1 from the shape of a tensor.
    img_embedding = tf.squeeze(img_embedding, axis=None, name=None)
    # Computes the mean of elements across dimensions of a tensor. [ Normalize ]
    img_embedding = tf.reduce_mean(img_embedding, axis=(0,1), keepdims=False, name=None).numpy()
    # Converting to List
    img_embedding = img_embedding.tolist()
    return img_embedding

ResNet101_embeddings = []
dir = r'/content/data/women_boots'
for filename in os.listdir(dir):
    if filename.endswith(".jpg"):
        ResNet101_embeddings.append(get_embeddings(os.path.join(dir, filename)))

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet101_weights_tf_dim_ordering_tf_kernels_notop.h5
171450368/171446536 [=====] - 2s 0us/step
time: 9min 47s (started: 2021-04-08 08:17:48 +00:00)

MobileNet

In []:

```

tf.keras.backend.clear_session()

model_embedding = tf.keras.applications.MobileNet(input_shape=(520,520,3), alpha=1.0,
                                                    depth_multiplier=1, dropout=0.001, include_top=False, weights='imagenet', input_tensor=None, pooling=None,)
def load_img(path):
    # Reading an image
    image = cv2.imread(path)
    # resizing because pre-trained model image shape is 520x520
    image = cv2.resize(image, (520,520), interpolation=cv2.INTER_AREA)
    # Converting to RGB because it will be saved as a correct image even if it is saved after being converted to a PIL

```

```

er being converted to a PIL
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
# Preprocessed numpy.array or a tf.Tensor with type float32.
image = tf.image.convert_image_dtype(image, tf.float32) [tf.newaxis, ...]
return image

def get_embeddings(path):
    image = load_img(path)
    # Returns 1024 dimension array/ vector with predicted values
    img_embedding = model_embedding.predict(image, steps=1)
    # Removes dimensions of size 1 from the shape of a tensor.
    img_embedding = tf.squeeze(img_embedding, axis=None, name=None)
    # Computes the mean of elements across dimensions of a tensor. [ Normalize ]
    img_embedding = tf.reduce_mean(img_embedding, axis=(0,1), keepdims=False, name=None).numpy()
    # Converting to List
    img_embedding = img_embedding.tolist()
    return img_embedding

MobileNet_embeddings = []
dir = r'/content/data/women_boots'
for filename in os.listdir(dir):
    if filename.endswith(".jpg"):
        MobileNet_embeddings.append(get_embeddings(os.path.join(dir, filename)))

```

WARNING:tensorflow: `input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
 Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobile_net/mobilenet_1_0_224_tf_no_top.h5
 17227776/17225924 [=====] - 0s 0us/step
 time: 6min 27s (started: 2021-04-08 08:37:24 +00:00)

InceptionV3

In []:

```

tf.keras.backend.clear_session()

model_embedding = tf.keras.applications.InceptionV3(include_top=False, weights='imagenet',
input_tensor=None, input_shape=(520, 520, 3), pooling=None,)
def load_img(path):
    # Reading an image
    image = cv2.imread(path)
    # resizing because pre-trained model image shape is 520x520
    image = cv2.resize(image, (520, 520), interpolation=cv2.INTER_AREA)
    # Converting to RGB because it will be saved as a correct image even if it is saved after being converted to a PIL
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Preprocessed numpy.array or a tf.Tensor with type float32.
    image = tf.image.convert_image_dtype(image, tf.float32) [tf.newaxis, ...]
    return image

def get_embeddings(path):
    image = load_img(path)
    # Returns 1024 dimension array/ vector with predicted values
    img_embedding = model_embedding.predict(image, steps=1)
    # Removes dimensions of size 1 from the shape of a tensor.
    img_embedding = tf.squeeze(img_embedding, axis=None, name=None)
    # Computes the mean of elements across dimensions of a tensor. [ Normalize ]
    img_embedding = tf.reduce_mean(img_embedding, axis=(0,1), keepdims=False, name=None).numpy()
    # Converting to List
    img_embedding = img_embedding.tolist()
    return img_embedding

InceptionV3_embeddings = []
dir = r'/content/data/women_boots'
for filename in os.listdir(dir):
    if filename.endswith(".jpg"):
        InceptionV3_embeddings.append(get_embeddings(os.path.join(dir, filename)))

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [=====] - 0s 0us/step
time: 8min 59s (started: 2021-04-08 08:43:52 +00:00)

In []:

```
print(len(DenseNet121_embeddings))  
print(len(InceptionV3_embeddings))  
print(len(MobileNet_embeddings))  
print(len(ResNet101_embeddings))  
print(len(ResNet50_embeddings))
```

5229
5229
5229
5229
5229
time: 2.36 ms (started: 2021-04-08 08:53:24 +00:00)

In []:

```
DenseNet121_embeddings_ =numpy.array([numpy.array(xi) for xi in DenseNet121_embeddings])  
InceptionV3_embeddings_ =numpy.array([numpy.array(xi) for xi in InceptionV3_embeddings])  
MobileNet_embeddings_ =numpy.array([numpy.array(xi) for xi in MobileNet_embeddings])  
ResNet101_embeddings_ =numpy.array([numpy.array(xi) for xi in ResNet101_embeddings])  
ResNet50_embeddings_ =numpy.array([numpy.array(xi) for xi in ResNet50_embeddings])
```

time: 2.11 s (started: 2021-04-08 09:11:04 +00:00)

In []:

```
def Average(lst):  
    return sum(lst) / len(lst)
```

time: 1 ms (started: 2021-04-08 09:08:53 +00:00)

In []:

```
ResNet50_embeddings_sparsity = []  
for i in range(len(ResNet50_embeddings_)):  
    non_zero = np.count_nonzero(ResNet50_embeddings_[i])  
    total_val = np.product(ResNet50_embeddings_[i].shape)  
    sparsity = (total_val - non_zero) / total_val  
    ResNet50_embeddings_sparsity.append(sparsity)
```

time: 76.5 ms (started: 2021-04-08 09:14:27 +00:00)

In []:

```
ResNet50_embeddings_sparsity = Average(ResNet50_embeddings_sparsity)  
print(ResNet50_embeddings_sparsity)
```

0.28845375923336203
time: 4.11 ms (started: 2021-04-08 09:14:29 +00:00)

In []:

```
import pandas as pd  
Application = ["DenseNet121", "InceptionV3", "MobileNet", "ResNet101", "ResNet50"]  
Sparsity = [DenseNet121_embeddings_sparsity, DenseNet121_embeddings_sparsity, MobileNet_embeddings_sparsity, ResNet101_embeddings_sparsity, ResNet50_embeddings_sparsity]  
  
App_Spar = pd.DataFrame(list(zip(Application, Sparsity)), columns = ['Application', 'Average Sparsity'])  
  
App_Spar.head()
```

Out []:

Application	Average Sparsity
-------------	------------------

0	Application	Average Sparsity
	DenseNet121	0.000173
1	InceptionV3	0.000173
2	MobileNet	0.048810
3	ResNet101	0.118761
4	ResNet50	0.288454

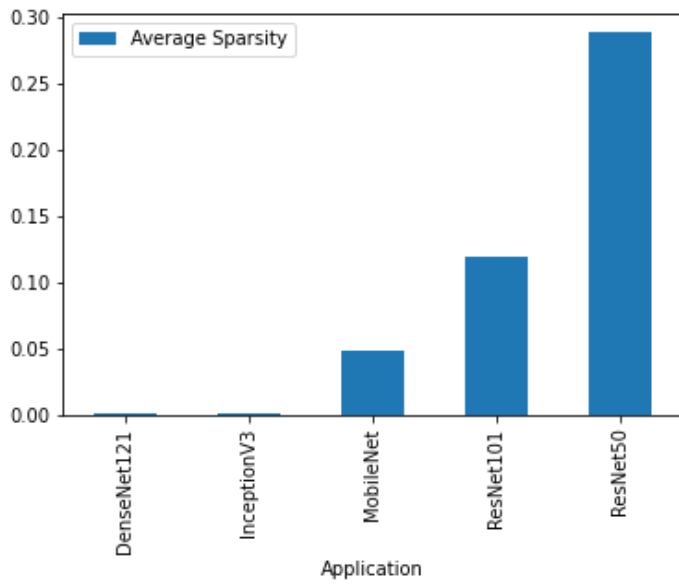
time: 59.1 ms (started: 2021-04-08 09:19:48 +00:00)

In []:

```
App_Spar.plot.bar(x = 'Application', y = "Average Sparsity", rot = 90)
```

Out[]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f59dbfe2dd0>



time: 211 ms (started: 2021-04-08 09:20:55 +00:00)