sprint3_autoencoder_features

December 23, 2022

1 Using features to cluster

[1]:
```
import tensorflow as tf
from tensorflow import keras
import os
import numpy as np
import pickle
```

```
2022-12-22 09:33:02.984474: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-12-22 09:33:03.372935: E tensorflow/stream_executor/cuda/cuda_blas.cc:2981]
Unable to register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
2022-12-22 09:33:04.911476: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot open shared
object file: No such file or directory
2022-12-22 09:33:04.911730: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_plugin.so.7:
cannot open shared object file: No such file or directory
2022-12-22 09:33:04.911757: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
```

First we load in the features that were created from the encoder of the autoencoder

[2]:
```
with open('features_encoder_pictures_with_no_buildings_last_version.pkl','rb')␣
↪as f:
    features = pickle.load(f)
features.shape
```

```
[2]: (10881, 16, 16, 128)
```

We flatten the feature shape (16, 16, 128) to 1D because clustering expects a 2D input.

```
[3]: features = features.reshape(features.shape[0], features.shape[1]* features.
     ↪shape[2]* features.shape[3])
     features.shape
```

```
[3]: (10881, 32768)
```

1.1 Clustering features

We use birch to cluster our images. The reason we choose birch is because in sprint2 we have determined that birch gave the best results.

We use 100 clusters, this would mean that we try to differentiate between 100 dishes. We now don't have to worry about having buildings or non food related images because most of them should be filtered out in our previous notebook.

```
[4]: from sklearn.cluster import Birch
     birch = Birch(n_clusters=100)
     predictions = birch.fit_predict(features)
     predictions
```

```
[4]: array([82, 32, 35, …, 21, 61, 40])
```

```
[5]: # with open('predictions.pkl','wb') as f:
     #     pickle.dump(predictions, f)
```

```
[6]: img_folder = "../tripadvisor_dataset/tripadvisor_images_small/"
     image_files=os.listdir(img_folder)
```

1.2 Clustering input image

We read in the image from the user

```
[7]: from fastai.vision.all import PILImage
     img = PILImage.create('../pasta.png')
     img
```

```
[7]:
```

```
[8]: IMG_HEIGHT = 128
IMG_WIDTH = 128

img_resized = img.resize((IMG_HEIGHT,IMG_WIDTH))
img_resized = np.array(img_resized)
img_resized.shape
```

```
[8]: (128, 128, 3)
```

```
[9]: img_resized_model = img_resized.reshape(1, IMG_HEIGHT, IMG_WIDTH, 3)
```

We extract the features of that image and reshape it to 2D

```
[11]: model = keras.models.load_model("./model_last_version")
results = model.predict(img_resized_model)
results = results.reshape(results.shape[0], results.shape[1]* results.shape[2]*
  ↪results.shape[3])
results.shape
```

```
2022-12-22 10:31:49.542424: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
```

```
compiler flags.
2022-12-22 10:31:50.627345: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 10405 MB memory:  -> device:
0, name: NVIDIA GeForce GTX 1080 Ti, pci bus id: 0000:b0:00.0, compute
capability: 6.1
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
was *not* compiled. Compile it manually.
```

```
2022-12-22 10:31:53.226560: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384]
Loaded cuDNN version 8500
```

```
1/1 [==============================] - 3s 3s/step
```

[11]:
```
(1, 32768)
```

Prediciting in which cluster that image belongs

[12]:
```python
pasta_cluster = birch.predict(results)
pasta_cluster
```

[12]:
```
array([67])
```

Getting the restaurants ids from all the restaurants in that cluster

[13]:
```python
indices = np.where(predictions == pasta_cluster)[0]
print(len(indices))
file_names = [image_files[i] for i in list(indices)]
unique_restaurants = set()
for file in file_names:
    unique_restaurants.add(int(file.split("_")[0]))
unique_restaurants
```

```
88
```

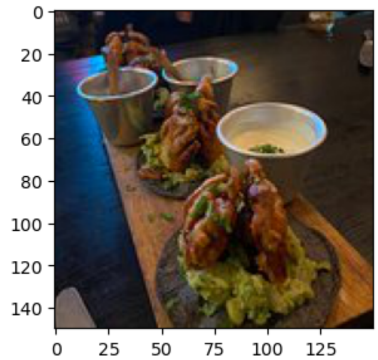
[13]:
```
{740645,
 740664,
 740695,
 740708,
 740802,
 784380,
 896603,
 1143981,
 1179020,
 1179069,
 1592412,
 1895959,
 2163268,
 2232741,
```

2385392,
3174943,
3368355,
3443862,
3457241,
3589043,
3602096,
3785414,
3819823,
4026066,
4197484,
4371500,
4510013,
4542201,
5500931,
5813529,
6022514,
6380688,
6920678,
7045750,
7154403,
7240054,
7243474,
7246379,
7253996,
7695538,
7989928,
8025230,
8319700,
8462977,
8495013,
8594246,
10027323,
10041401,
10062100,
10173853,
10469486,
10644250,
10661664,
10730255,
10830257,
11797609,
12007215,
12300282,
12606150,
12850619,
12883299,

```
13432522,
13473197,
13479805,
13523455,
14013341,
14990372,
18153830,
18453187,
18455497,
18542716,
18543205,
19383824,
19853784,
20159380,
20938371,
20996099,
21022432,
21053410,
21058987,
21092655,
23222867,
23744059,
23938101,
23954909,
24889503}
```

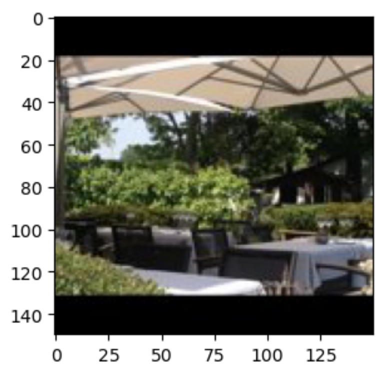
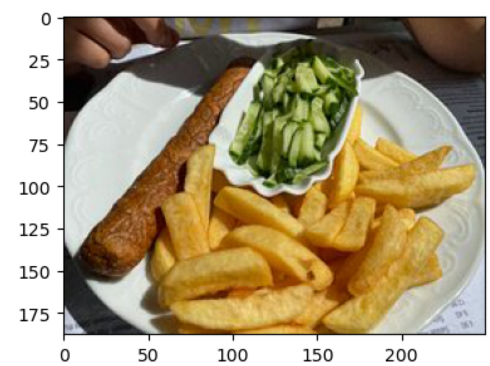
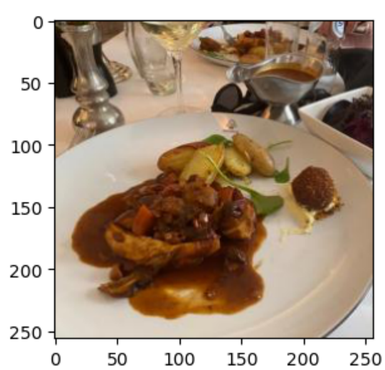
1.3 Plotting images that are in the cluster

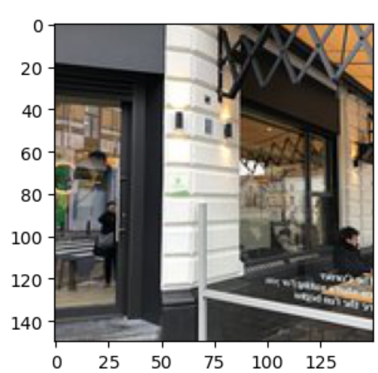
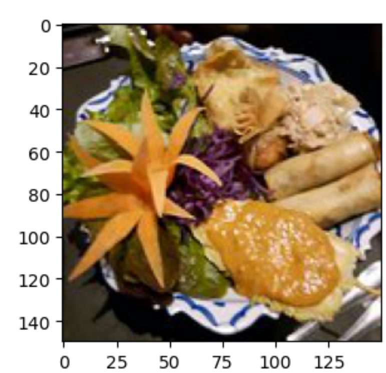
[14]:
```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(10,15))

for i in range(0,10):
    plt.subplot(5,2,i+1)
    img = PILImage.create(img_folder + '/' + file_names[i])
    # restaurant name get be obtained in file_names[i]
    plt.imshow(img)
fig.tight_layout()
plt.show()
```


0 20 40 60 80 100 120 140
0 25 50 75 100 125

0 25 50 75 100 125 150 175
0 50 100 150 200

0 25 50 75 100 125 150 175
0 50 100 150 200 250

0 20 40 60 80 100 120 140 160
0 50 100 150 200

0 20 40 60 80 100 120 140
0 25 50 75 100 125

0 25 50 75 100 125 150 175
0 50 100 150 200

0 50 100 150 200 250
0 50 100 150 200 250

0 20 40 60 80 100 120 140 160
0 50 100 150 200

0 20 40 60 80 100 120 140
0 25 50 75 100 125

0 20 40 60 80 100 120 140
0 25 50 75 100 125

1.4 Conclusion:

We are sadly not seeing any pasta related images, so it hasn't worked well. It's performing worse than HOG because now we don't see a relation between the images.

We also see some non food related images even after we filtered them out.

We don't know why this performs so bad. Our autoencoder has high accuracy, so the problem is not that we have a bad autoencoder. The problem must be our cluster method. To perform clustering we had to flatten our feature shape, we partially lose spacial information of neighboring pixels.

As last attempt we try to do another clustering method, namely k-means.

1.5 K-means

[15]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=100)
predictions = kmeans.fit_predict(features)
predictions
```

[15]:
```
array([30, 85, 41, …, 52, 33, 30], dtype=int32)
```

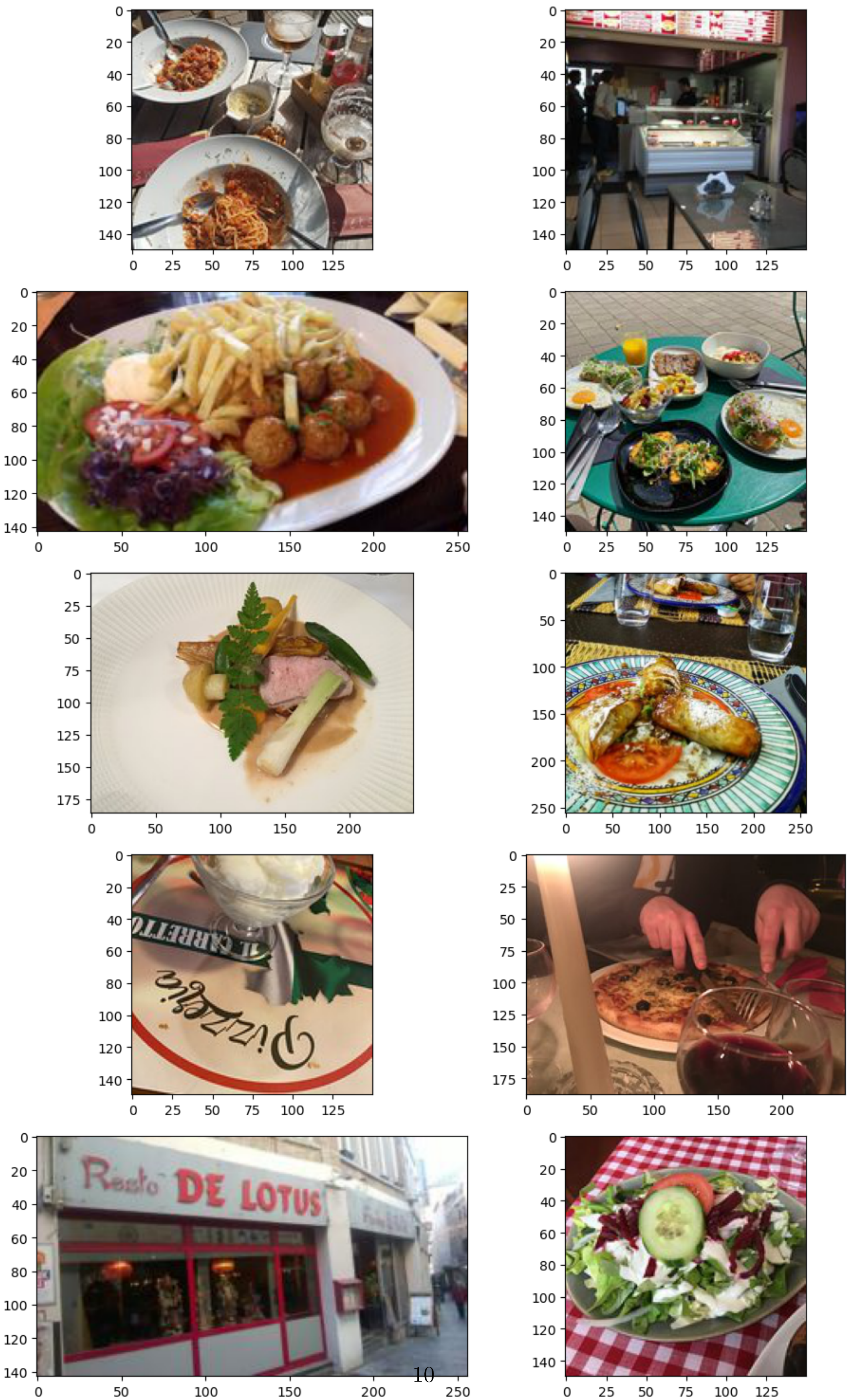
The method `process_image()` does the same as the code cells above applied on Birch but in a compact manner.

[16]:
```
def process_image(predictions, cluster_model):
    img_folder = "../tripadvisor_dataset/tripadvisor_images_small/"
    image_files=os.listdir(img_folder)
    img = PILImage.create('../pasta.png')
    IMG_HEIGHT = 128
    IMG_WIDTH = 128

    img_resized = img.resize((IMG_HEIGHT,IMG_WIDTH))
    img_resized = np.array(img_resized)
    img_resized_model = img_resized.reshape(1, IMG_HEIGHT, IMG_WIDTH, 3)

    model = keras.models.load_model("./model_last_version")
    results = model.predict(img_resized_model)
    results = results.reshape(results.shape[0], results.shape[1]* results.
 ↪shape[2]* results.shape[3])

    pasta_cluster = cluster_model.predict(results)
    indices = np.where(predictions == pasta_cluster)[0]
    print(len(indices))

    file_names = [image_files[i] for i in list(indices)]
```

```
    unique_restaurants = set()
    for file in file_names:
        unique_restaurants.add(int(file.split("_")[0]))
    print(unique_restaurants)

    fig=plt.figure(figsize=(10,15))
    for i in range(0,10):
        plt.subplot(5,2,i+1)
        img = PILImage.create(img_folder + '/' + file_names[i])
        # restaurant name get be obtained in file_names[i]
        plt.imshow(img)
    fig.tight_layout()
    plt.show()
    return unique_restaurants
```

[17]:
```
restaurants = process_image(predictions, kmeans)
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
was *not* compiled. Compile it manually.
1/1 [==============================] - 0s 125ms/step
49
{23819264, 740613, 2196877, 694671, 8451088, 3847454, 4557048, 7726497, 3589154,
12273827, 10131620, 7828141, 3602096, 740657, 11803061, 1179193, 8494267,
4553022, 7742785, 4971969, 4993603, 15058628, 7194566, 816070, 8733128,
15011401, 4371274, 13432522, 740682, 8744781, 20988878, 1223250, 13799895,
1536347, 12686684, 4225501, 12367843, 7154403, 10622693, 6351089, 12972530,
7285489, 12559476, 23641205, 9869041, 13847543, 1054072, 8486394, 10299899}
```


Pizzeria
Resto DE LOTUS

1.6 Conclusion

We finally see a pasta image, but that could be coincidence because the other images aren't pasta. We see more round shapes than when we cluster with birch which is a good sign.

We currently have around 10000 images that we try to cluster in 100 clusters. The amount of clusters to choose is a hard task, we've already explained this in depth in sprint 2 how to choose the amount of clusters. A high amount would mean more precision and a low amount would mean more recall.

We think that seperating a specific dish is a very hard task with clustering. To be able to cluster each dish to a seperate cluster would require knowledge about how many dishes there are. All the images that aren't food must be removed too. Knowing the amount of dishes and filtering out all the non food related images is a hard task.

A better approach might be to train a classifier to classify our input image as a dish, and from there recommending a restaurant that serves that dish. We've done that in this notebook: `transfer-learning-different-configurations.ipynb`