Assignment 1 - Understanding a dataset

March 31, 2023

1 Assignment 1

Note: This notebook file for the assignment has deviations from the course guide with respect to the structure, sentence framing, question framing and numbering. Please consider this notebook file structure as the final structure and follow this.

In this assignment, you will explore the CIFAR10 dataset.

You have to download the dataset from Pytorch.

Comment your code and indicate what the different instructions are doing and what you are showing and printing. When printing figures do not forget about the title, x and y labels. The font size should be matching the text size of the text in your report. Do not forget to add legends to the plots.

---

[1]:
```
# Load all the needed packages for this assignment here
import numpy as np
# include packages you will be using
import torchvision
%matplotlib inline
import matplotlib.pyplot as plt
```

```
/Users/hungnguyen/miniforge3/envs/DS-CVassignment/lib/python3.10/site-
packages/torchvision/io/image.py:13: UserWarning: Failed to load image Python
extension: dlopen(/Users/hungnguyen/miniforge3/envs/DS-
CVassignment/lib/python3.10/site-packages/torchvision/image.so, 0x0006): symbol
not found in flat namespace '__ZN2at4_ops19empty_memory_format4callEN3c108ArrayR
efINS2_6SymIntEEENS2_8optionalINS2_10ScalarTypeEEENS6_INS2_6LayoutEEENS6_INS2_6D
eviceEEENS6_IbEENS6_INS2_12MemoryFormatEEE'
  warn(f"Failed to load image Python extension: {e}")
```

Exercise 1.1 - Load data **a)** Load the CIFAR10 dataset.

b) Print the number of samples and the number of classes present in the dataset.

c) Also print the shape of an image in the dataset.

[7]:
```
# Ex.1.1a,b & c
DATASET_LOCATION = "cifar10"
```

```
cifar10_train = torchvision.datasets.CIFAR10(DATASET_LOCATION, train=True,
 ↪download=False)
cifar10_test = torchvision.datasets.CIFAR10(DATASET_LOCATION, train=False,
 ↪download=False)
```

[8]:
```
print(cifar10_train.__len__())
print(cifar10_test.__len__())
print(cifar10_train[0][0].size)
```

```
50000
10000
(32, 32)
```

Exercise 1.2 - Quantify dataset **a)** Print the number of samples per category.

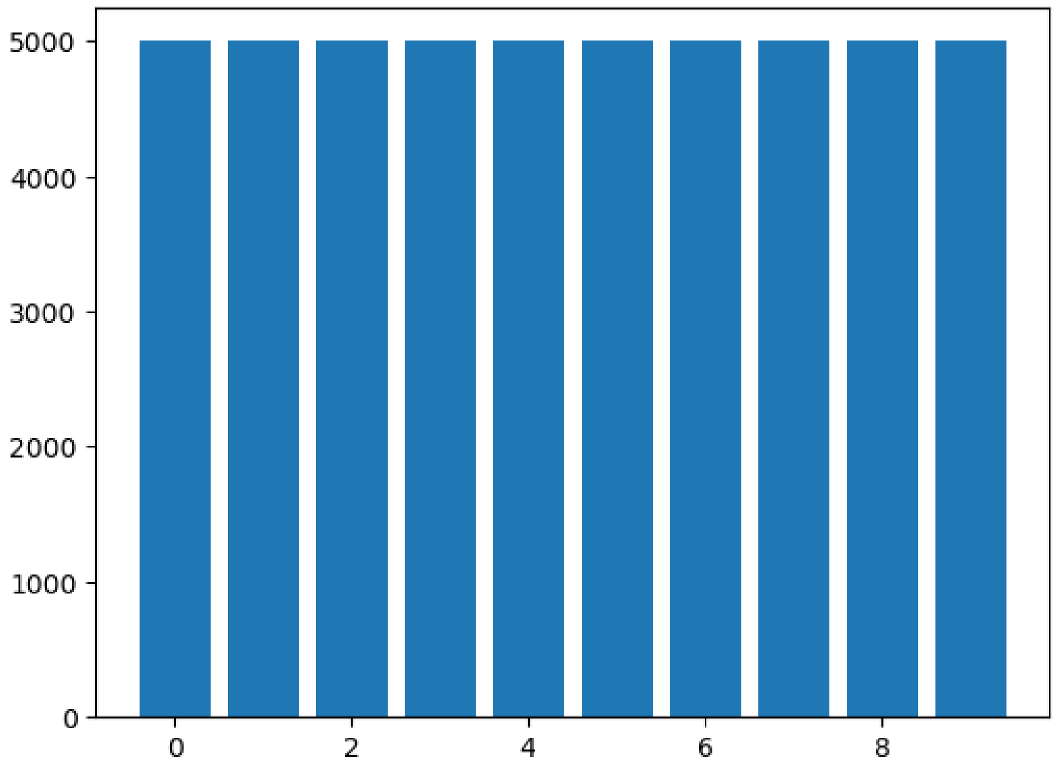
[9]:
```
# Ex.1.2a your code here
from collections import Counter
samples_per_cat = Counter(cifar10_train.targets)
print(samples_per_cat)
```

```
Counter({6: 5000, 9: 5000, 4: 5000, 1: 5000, 2: 5000, 7: 5000, 8: 5000, 3: 5000,
5: 5000, 0: 5000})
```

b) Plot the number of samples per category using a bar plot.

[10]:
```
# Ex.1.2b your code here
plt.bar(samples_per_cat.keys(), samples_per_cat.values())
```

[10]:
```
<BarContainer object of 10 artists>
```



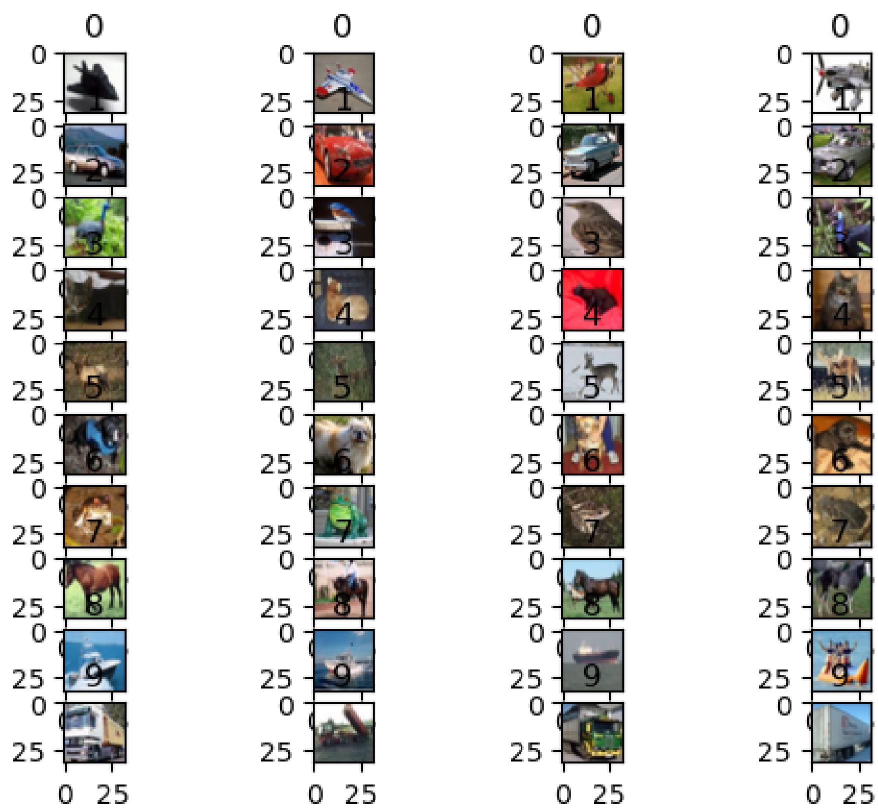
Reflection: Answer the below question

Are you working with a balanced or an unbalanced dataset? Are there majoritarian classes? Do you think this will affect the later analysis and training of your models?

Your Answer (Double click to edit): A very balanced dataset.

Exercise 1.3 - Visualize images in your dataset Create a figure with n x 4 sub-plots. The value of 'n' depends on the number of categories present in the dataset. As the title of each row in your figure, indicate the category it belongs to.

[11]:
```
# Ex.1.3 your code here
fig, axs = plt.subplots(10, 4)
for key in sorted(samples_per_cat.keys()):
    i = 0
    x = 0
    while i < 4:
        if cifar10_train[x][1] == key:
            axs[key, i].imshow(cifar10_train[x][0])
            axs[key, i].set_title(cifar10_train[x][1])
            i += 1
        x += 1
```



Exercise 1.4 - RGB feature extraction Extract RGB values from each image in your dataset as three seperate lists(one per channel). Each list should have 8 values. To do so, you can compute the histogram of each channel with 8 bins. Then you have to concatenate the values of all the three channels together resulting in a feature vector of size 24. This feature vector is the descriptor of an image in your dataset. You will have to do this for all the images present in your dataset in order to get the overall RGB descriptor which will be of size (n,24). Here 'n' depends on the number of samples present in the dataset.

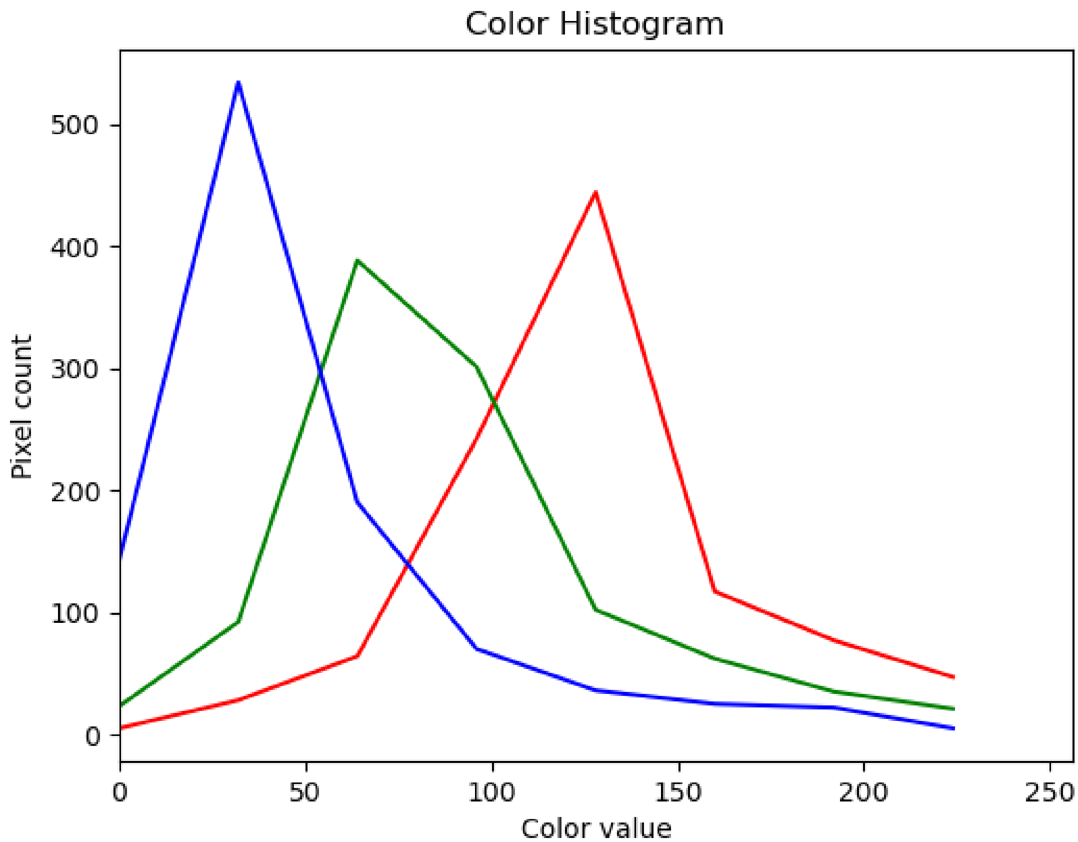
[12]:
```
# Ex.1.4 your code here
colors = ("red", "green", "blue")
image = np.array(cifar10_train[0][0])

plt.figure()
plt.xlim([0, 256])
for channel_id, color in enumerate(colors):
    histogram, bin_edges = np.histogram(
        image[:, :, channel_id], bins=8, range=(0, 256)
    )
    plt.plot(bin_edges[0:-1], histogram, color=color)

plt.title("Color Histogram")
```

```
plt.xlabel("Color value")
plt.ylabel("Pixel count")
```

[12]: Text(0, 0.5, 'Pixel count')



[23]:
```
feature_list = []
for image, label in cifar10_train:
    image_feature = []
    # image_feature_with_label = []
    colors = ("red", "green", "blue")
    print(image)
    image = np.array(image)
    # print(image.shape)
    # plt.imshow(image)
    # break
    # image_feature_with_label.append(label)
    for channel_id, color in enumerate(colors):
        histogram, bin_edges = np.histogram(
            image[:, :, channel_id], bins=8, range=(0, 256)
        )
```

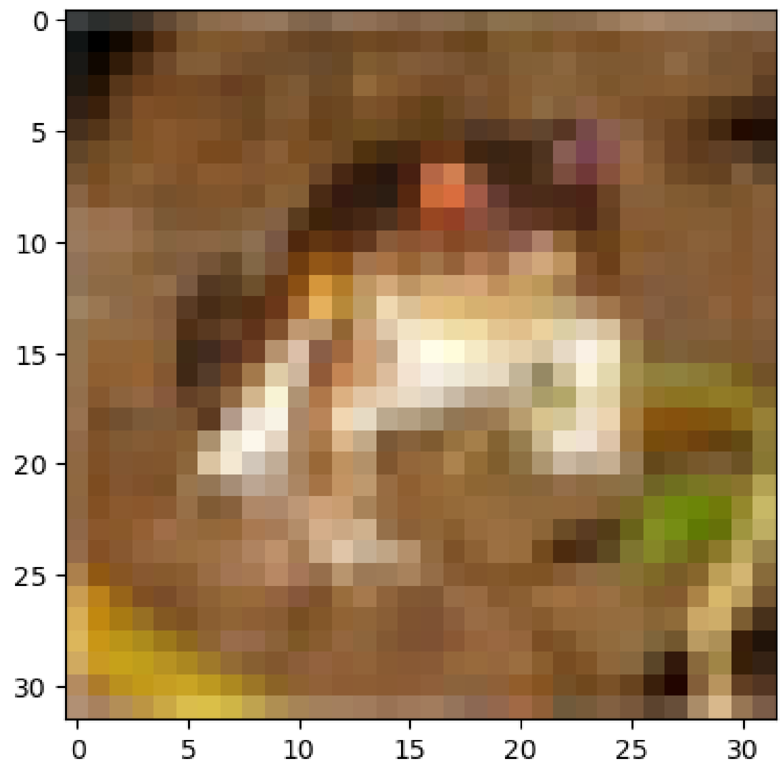
```
        image_feature = np.concatenate((image_feature, histogram))
    # print(image_feature)
    image_feature = image_feature / 784
    image_feature = np.append(image_feature,label)
    feature_list.append(image_feature)



feature_list
```

```
<PIL.Image.Image image mode=RGB size=32x32 at 0x2A3D674C0>
(32, 32, 3)
```

[23]: []



Exercise 1.5 - Correlation among samples of the different categories After extracting the overall RGB descriptor from previous exercise, concatenate the labels(each category represents a label) to it.

a) Compute the intra-class variability of your dataset.

Intra-class correlation aims at understanding the compactness of a class/group/category. This is done basically through the computation of a score of similarity among samples. In this assignment, the purpose of doing intra-class correlation is basically to check the similarity between the samples of each class, so that we would have an idea of how similar the dataset is, for each class.

For this, you can use the implementation from the pingouin package- https://pingouin-stats.org/build/html/generated/pingouin.intraclass_corr.html

Here is an example: https://www.statology.org/intraclass-correlation-coefficient-python/. You are not obliged to follow it. You can implement your own function or another one you may implement.

[15]:
```
import pandas as pd

features_df = pd.DataFrame(feature_list)
features_df = features_df.rename({24: 'label'}, axis='columns')
features_df = features_df.rename({0: 'rating'}, axis='columns')
features_df = features_df.sort_values(by='label')
array_index = np.arange(1, 5001, 1, dtype=int)
for i in range(9):
    newData = np.arange(1, 5001, 1, dtype=int)
    array_index = np.concatenate([array_index, newData])
#concatonate

features_df['index'] = array_index
# features_df = pd.concat([features_df, pd.DataFrame(array_index)], axis=1)
features_df
```

[15]:
```
         rating         1         2         3         4         5         6  \
29513  0.022959  0.137755  0.322704  0.539541  0.232143  0.029337  0.021684
16836  0.073980  0.082908  0.080357  0.056122  0.065051  0.505102  0.442602
32316  0.000000  0.003827  0.015306  0.125000  0.200255  0.225765  0.735969
32318  0.058673  0.043367  0.058673  0.344388  0.515306  0.238520  0.047194
32326  0.627551  0.088010  0.047194  0.073980  0.114796  0.207908  0.131378
…           …         …         …         …         …         …         …
13795  0.077806  0.070153  0.246173  0.118622  0.193878  0.420918  0.105867
25994  0.005102  0.307398  0.200255  0.142857  0.116071  0.133929  0.213010
36910  0.170918  0.168367  0.274235  0.252551  0.227041  0.061224  0.133929
21518  0.173469  0.126276  0.093112  0.146684  0.141582  0.140306  0.094388
25648  0.174745  0.261480  0.141582  0.107143  0.213010  0.211735  0.154337

              7         8         9  …        16        17        18  \
29513  0.000000  0.025510  0.144133  …  0.015306  0.109694  0.127551
16836  0.000000  0.045918  0.048469  …  0.072704  0.080357  0.131378
32316  0.000000  0.000000  0.002551  …  0.000000  0.003827  0.110969
32318  0.000000  0.039541  0.047194  …  0.024235  0.051020  0.048469
32326  0.015306  0.001276  0.019133  …  0.000000  0.000000  0.016582
…           …         …         …  …  …         …         …
13795  0.072704  0.079082  0.068878  …  0.077806  0.056122  0.284439
25994  0.187500  0.002551  0.292092  …  0.003827  0.329082  0.281888
36910  0.017857  0.204082  0.172194  …  0.169643  0.176020  0.158163
```

```
21518  0.390306  0.116071  0.122449  …  0.096939  0.154337  0.168367
25648  0.042092  0.329082  0.262755  …  0.262755  0.312500  0.198980

             19        20        21        22        23  label  index
29513  0.077806  0.065051  0.082908  0.793367  0.034439    0.0      1
16836  0.522959  0.484694  0.012755  0.001276  0.000000    0.0      2
32316  0.125000  0.163265  0.123724  0.779337  0.000000    0.0      3
32318  0.070153  0.112245  0.318878  0.553571  0.127551    0.0      4
32326  0.015306  0.075255  0.696429  0.271684  0.230867    0.0      5
…           …         …         …         …         …      …    …
13795  0.075255  0.118622  0.118622  0.526786  0.048469    9.0   4996
25994  0.178571  0.137755  0.103316  0.107143  0.164541    9.0   4997
36910  0.253827  0.209184  0.227041  0.096939  0.015306    9.0   4998
21518  0.184949  0.136480  0.099490  0.108418  0.357143    9.0   4999
25648  0.089286  0.104592  0.116071  0.121173  0.100765    9.0   5000

[50000 rows x 26 columns]
```

[11]:
```python
# Ex.1.5a your code here
from pingouin import intraclass_corr
# print(intraclass_corr(features_df, targets= label, rating = column, rater =␣
↪class_index))
# label them from 1 to 5000, class index.
#raters are the images themselves. We could create a collumn with all numbers␣
↪for 1 to 5000. We want to create an index, sort. First sort so you get all␣
↪images for class n then n+1, and then create index for 1-5000.
# last collumn is label

icc = intraclass_corr(data=features_df, targets='label', raters='index',␣
↪ratings='rating').round(3)

icc
```

b) Compute the inter-class variability of your dataset.

Inter-class correlation aims at understanding the relationship/correlation among the classes/categories present in your dataset. For this, you could compute a measure (for example mean, std etc.) collectively for all the samples belonging to each and every class of the dataset. Then you could make use of this measure to find the correlation among the classes/categories using the standard pandas dataframe correlation function. Link: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

[12]:
```python
# Ex.1.5b your code here
features_df = pd.DataFrame(feature_list)
features_df = features_df.rename({24: 'label'}, axis='columns')
mean_df = pd.DataFrame()
for i in range(10):
```

```
    x = features_df.sort_values(by='label').loc[features_df['label'] == i].
↪mean()
    x = x.iloc[::-1]
    mean_df = pd.concat([mean_df, x.to_frame()], axis=1)

mean_df.columns = mean_df.iloc[0]
mean_df = mean_df[1:]
mean_df.corr()
```

[12]:

```
label       0.0       1.0       2.0       3.0       4.0       5.0       6.0  \
label
0.0    1.000000  0.052321  0.470928  0.139518  0.322096  0.249280  0.123002
1.0    0.052321  1.000000  0.806705  0.916883  0.881101  0.863930  0.931807
2.0    0.470928  0.806705  1.000000  0.916491  0.979866  0.948224  0.907693
3.0    0.139518  0.916883  0.916491  1.000000  0.954946  0.984555  0.981524
4.0    0.322096  0.881101  0.979866  0.954946  1.000000  0.962833  0.959474
5.0    0.249280  0.863930  0.948224  0.984555  0.962833  1.000000  0.952901
6.0    0.123002  0.931807  0.907693  0.981524  0.959474  0.952901  1.000000
7.0    0.315648  0.866904  0.968539  0.962153  0.984896  0.968334  0.956356
8.0    0.887582  0.355914  0.674754  0.417146  0.564242  0.525305  0.374512
9.0    0.337923  0.890496  0.835521  0.837122  0.876925  0.806994  0.859290

label       7.0       8.0       9.0
label
0.0    0.315648  0.887582  0.337923
1.0    0.866904  0.355914  0.890496
2.0    0.968539  0.674754  0.835521
3.0    0.962153  0.417146  0.837122
4.0    0.984896  0.564242  0.876925
5.0    0.968334  0.525305  0.806994
6.0    0.956356  0.374512  0.859290
7.0    1.000000  0.530659  0.894939
8.0    0.530659  1.000000  0.504796
9.0    0.894939  0.504796  1.000000
```

c) Compute the Silhouette score.

The Silhouette score is used to assess the performance of using unsupervised machine learning (clustering). We can also use it here to assess the compactness of the extracted descriptors per category and for the group of categories as their mean.

You can use the function available in Sklearn - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[13]:

```
# Ex.1.5c your code here
from sklearn.metrics import silhouette_score
feature_df_without_label = features_df.drop(columns=['label'])
silhouette_score(feature_df_without_label, labels=features_df['label'])
```

[13]: -0.08492854865203084

Reflection: (Answer the below questions)

1. Does Intra-class correlation score/coefficient help you assess the degree of similarity among the samples of a category? > Your Answer (Double click to edit): > > The result of the average raters ICC is >0.9. This means the inter-rater agreement measures are excellent. This means that multiple raters have consistebt ratings for the same item, so there is a good degree of similarity. This is paired with a Confidence interval of 0.02, since this is smaller than 0.05 it is statistically accurate. > > The result of the single rater ICC is <0.4, so the inter-rater agreement measures are poor. The confidence interval is also 0.06, which is larger than 0.05, so it is not that statistically accurate.

2. What can you deduce from the Inter-class correlation and Silhouette score? > Your Answer (Double click to edit): > Inter-class correlation shows how similar the various classes. A high ICC signifies that the classes are very similar. A low one that they are easy to distinguish. This can be used to see how difficult it is to recognize different classes. > Silhouette score equals approx. 0, means that there are a lot of overlap between classes.

Exercise 1.6 - Dimensionality reduction for visualization We can visualize large datasets having higher dimensions or features in 2- or 3-dimensional spaces. For this, you need to reduce the dimensionality of the data.

In this exercise, you are asked to use PCA for reducing dimensionality.

Link to function to apply PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.htm

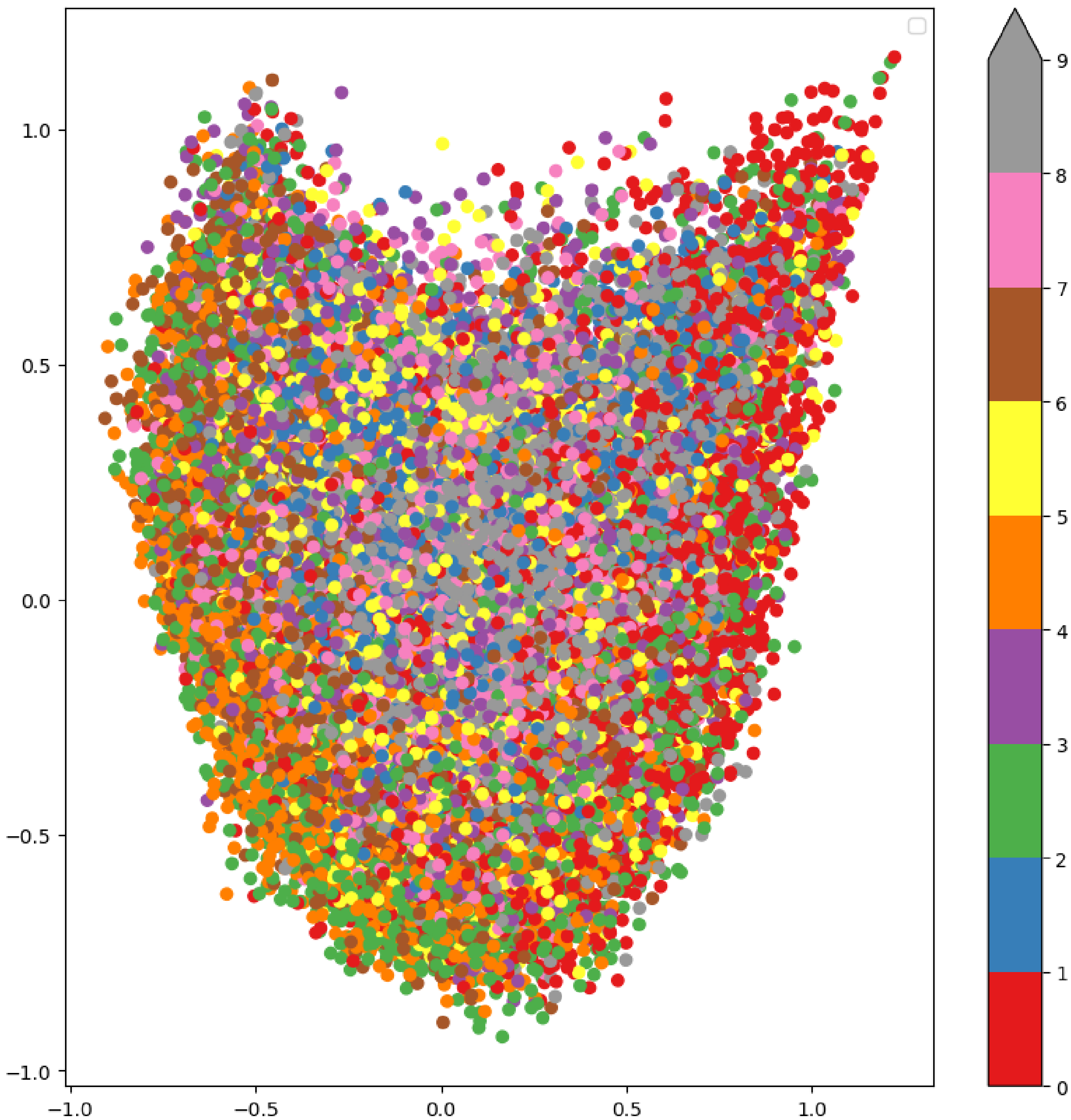
Create the following two figures:

a) Rely on the first 2 principal components to plot the samples of your dataset. Use one color per class.

[84]:
```
# Ex.1.6a your code here
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
features_pca_2 = pca.fit_transform(X=features_df.iloc[:, :24], y=features_df.
 ↪iloc[:, 24])
features_pca_2 = np.column_stack((features_pca_2, features_df.iloc[:, 24].
 ↪to_numpy().astype(int)))
plt.figure(figsize=(10,10))
plt.scatter(features_pca_2[:,0], features_pca_2[:,1], c=features_pca_2[:,2],␣
 ↪cmap='Set1')
plt.colorbar(extend="max")
plt.legend()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```

[84]: <matplotlib.legend.Legend at 0x179646ac0>



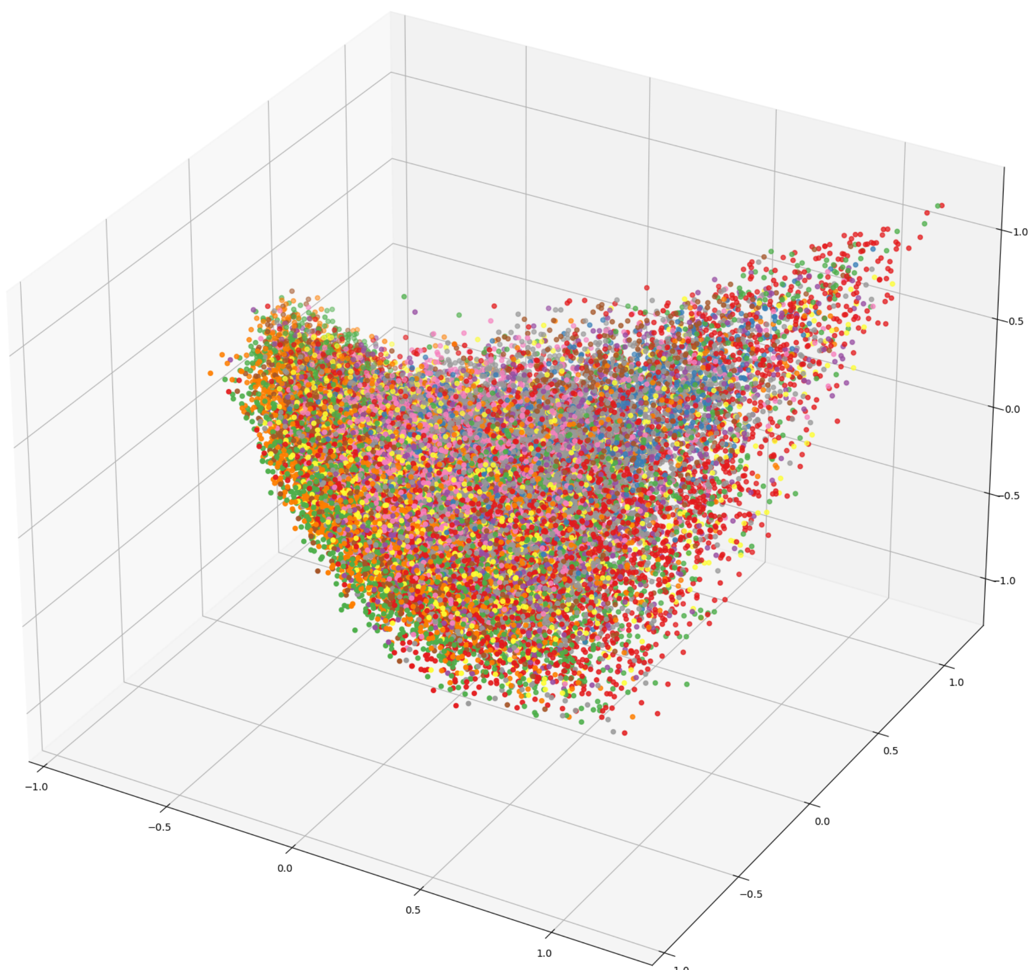
b) Rely on the first 3 principal components to create a 3D plot. Use one color per class.

[88]:
```
# Ex.1.6b your code here
pca = PCA(n_components=3)
fig = plt.figure(figsize=(20,20))
ax = fig.add_subplot(projection='3d')
features_pca_2 = pca.fit_transform(X=features_df.iloc[:, :24], y=features_df.
 ↪iloc[:, 24])
features_pca_2 = np.column_stack((features_pca_2, features_df.iloc[:, 24].
 ↪to_numpy().astype(int)))
ax.scatter(features_pca_2[:,0], features_pca_2[:,1], features_pca_2[:,2],␣
 ↪c=features_pca_2[:,3], cmap='Set1')
```

```
ax.legend()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[88], line 8
      6 features_pca_2 = np.column_stack((features_pca_2, features_df.iloc[:,␣
 ↪24].to_numpy().astype(int)))
      7 ax.scatter(features_pca_2[:,0], features_pca_2[:,1], features_pca_2[:
 ↪,2], c=features_pca_2[:,3], cmap='Set1')
----> 8 ax.colorbar(extend="max")
      9 ax.legend()

AttributeError: 'Axes3DSubplot' object has no attribute 'colorbar'
```



Exercise 1.7 - Reflection Reflect on the following questions.

a) Will you obtain the same visualisation in the feature space for different extracted features? > Your Answer (Double click to edit): I don't think so.

b) Are the classes distinguishable on the feature space when relying on PCA over RGB? > Your Answer (Double click to edit): Doesn't seem so. The PCA plot implies that there are no clear distinguish between each class (maybe increase PCA component could help, but relying only on one column to make the descriptors does not seems to be appropriate for a large image.

c) What other visualization could you include to better describe your data? > Your Answer (Double click to edit): maybe a t-SNE visualization, correlation matrix,…

[Optional] Exercise: Repeat experiments with different image descriptors e.g. - Harris Corner Detection

- Shi-Tomasi Corner Detector and Good Features to Track
- Scale-Invariant Feature Transform (SIFT)
- Speeded-up robust features (SURF)
- Features from Accelerated Segment Test (FAST)
- Blob Detectors With LoG, DoG, and DoH

If you have OpenCV installed you can follow this example, https://automaticaddison.com/image-feature-detection-description-and-matching-in-opencv/

When using Scikit-image, https://scikit-image.org/docs/dev/api/skimage.feature.html?highlight=hog

[ ]:

```

```