EX 2

written by:Roi Hezkiyahu

imports

In [6]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
from os import path,listdir
from pydiffmap import diffusion_map as dm
from pydiffmap.visualization import embedding_plot, data_plot
from sklearn.manifold import TSNE
from umap.umap_ import UMAP
from time import time
```

Q 1

Plot a simple example in 2D for which calculating the SVD of X is not equivalent to calculating the Principal components of X. Explain.

In [7]:

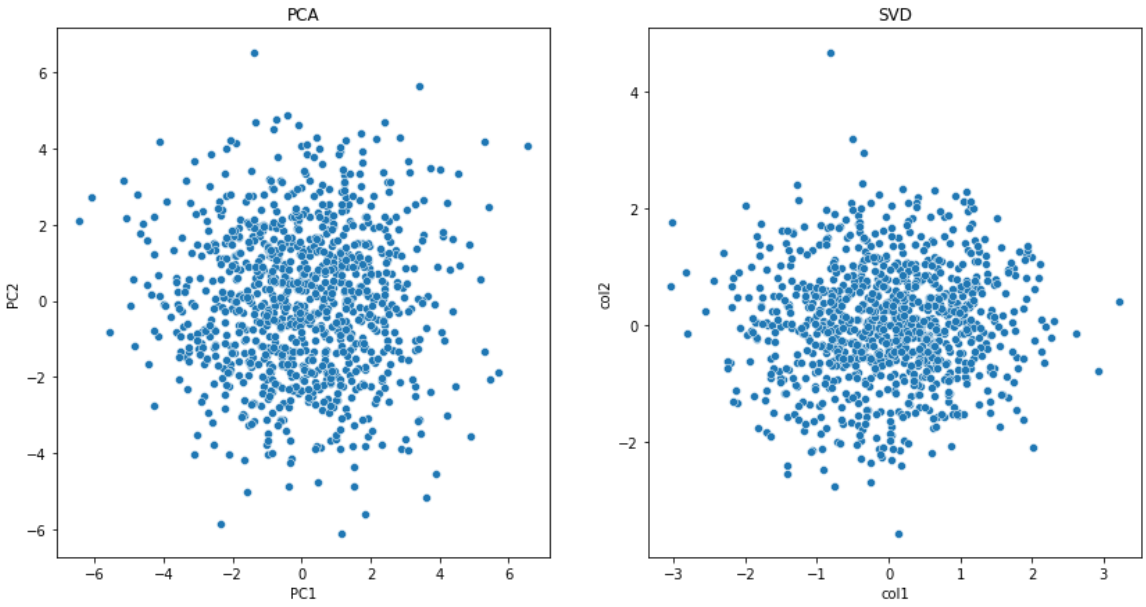
```
X = np.random.randn(1000,1000)
pca = PCA(2)
X_pca = pca.fit_transform(X)
U, Sigma, Vh = np.linalg.svd(X)
X_svd = U @ np.diag(Sigma) @ Vh[:,:2]
```

In [8]:

```
fig, (ax1, ax2) = plt.subplots(1, 2,figsize = (14,7))

sns.scatterplot(X_pca[:,0], X_pca[:,1], ax = ax1)
ax1.set_title("PCA")
ax1.set_ylabel("PC2")
ax1.set_xlabel("PC1")
sns.scatterplot(X_svd[:,0], X_svd[:,1], ax = ax2)
ax2.set_title("SVD")
ax2.set_ylabel("col2")
ax2.set_xlabel("col1")
plt.show()
```

```
C:\Users\Nir\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



we can see that the plots are different, this is beacuse SVD is applied to X, and in PCA we apply SVD to the centered X

Q 2

Use the two leading principle components to project a 1000 samples of the digit ’1’ into a 2 dimensional space. Use scatter plot to show the result.

In [9]:

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train.reshape(-1,784)
```

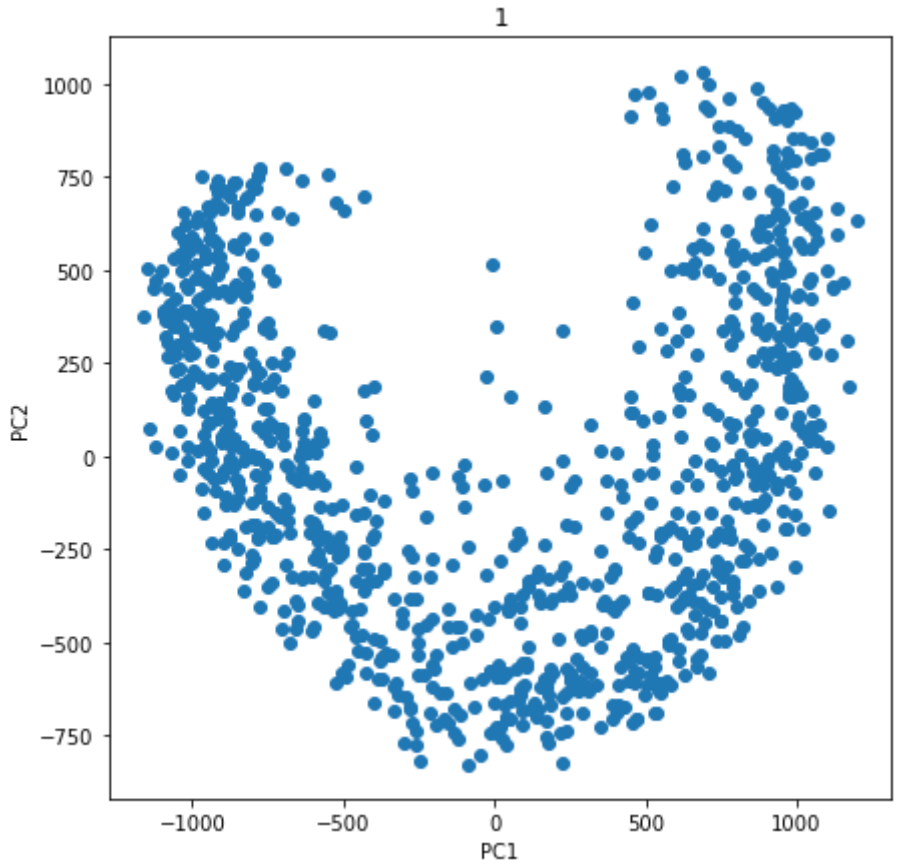
In [10]:

```
def plot_2_pcs(x_pca, title = "1"):
    plt.figure(figsize=(7,7))
    plt.scatter(x_pca[:,0],x_pca[:,1])
    plt.title(title)
    plt.ylabel("PC2")
    plt.xlabel("PC1")
    plt.show()

def pca_by_y_label(x_train,y_train,y_label, num_pcs=2, n_obs=1000, plot=True, title =
"1"):
    label_subset = x_train[np.isin(y_train, y_label)][:n_obs]
    pca = PCA(num_pcs)
    x_pca = pca.fit_transform(label_subset)
    if plot:
        plot_2_pcs(x_pca, title)
    return label_subset, x_pca
```

In [11]:

```
label_subset_1, x_pca = pca_by_y_label(x_train,y_train,y_label = [1], num_pcs=2, n_obs=
1000)
```

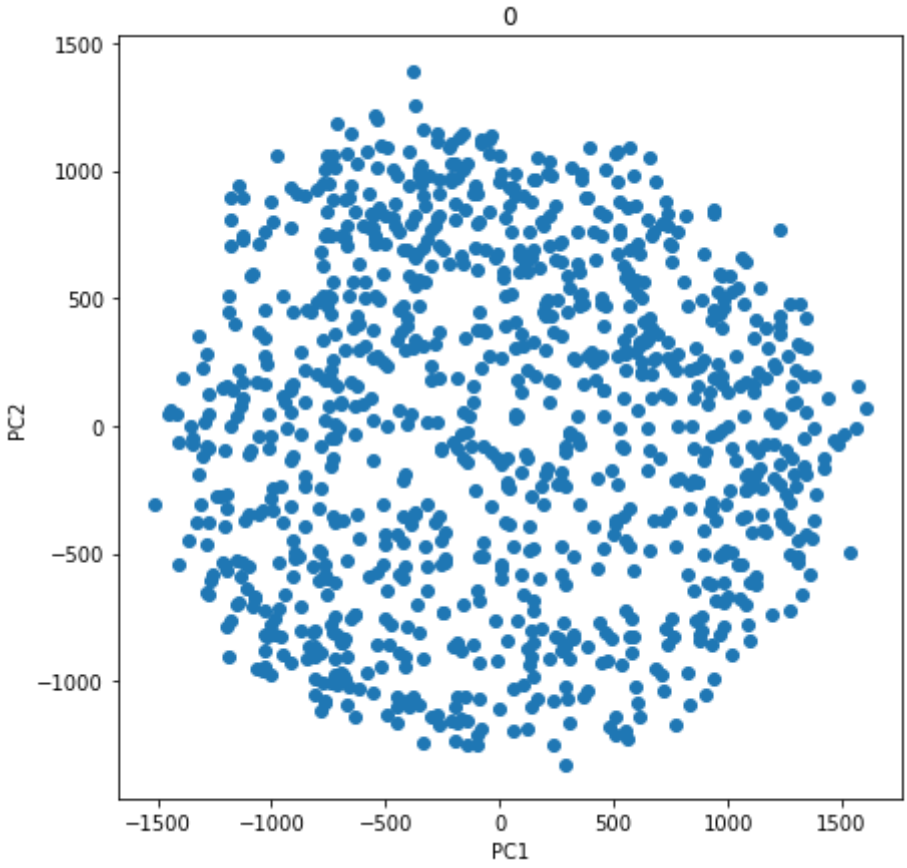


Q 3

Use the two leading principle components to project a 1000 samples of the digit ’0’. Use scatter plot to show the result.

In [12]:

```
label_subset_0, x_pca = pca_by_y_label(x_train,y_train,y_label = [0], num_pcs=2, n_obs=
1000, title = 0)
```

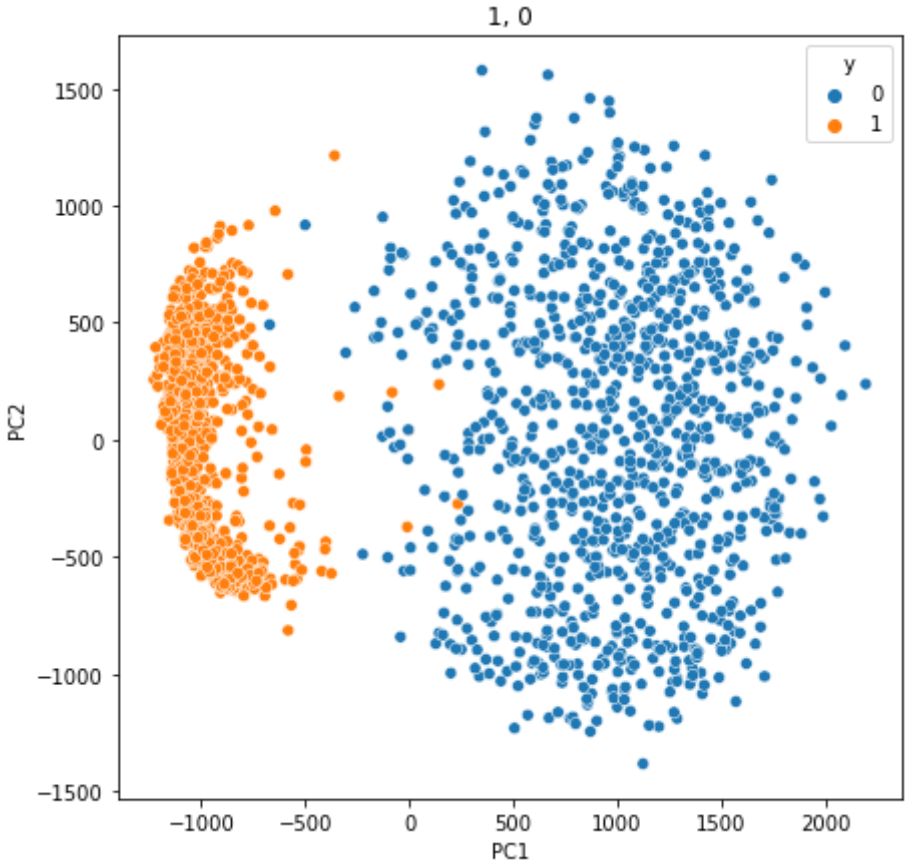


Q 4

Use the two leading principle components to project a 2000 combined samples from the last two sections. Use scatter plot to show the result, with color to indicate the samples label.

In [14]:

```
X_1_0 = np.concatenate([label_subset_1,label_subset_0 ])
X_1_0_pcaed = pd.DataFrame(pca.fit_transform(X_1_0), columns = ["PC1" ,"PC2"])
X_1_0_pcaed["y"] = np.array([1]*1000 + [0]*1000)
plt.figure(figsize=(7,7))
sns.scatterplot(data = X_1_0_pcaed, x="PC1", y="PC2",hue = "y")
plt.title("1, 0")
plt.ylabel("PC2")
plt.xlabel("PC1")
plt.show()
```



Q 5

How many components are required to capture 90% of the variance of the datasets from last three sections.

In [15]:

```
pca = PCA()
pca.fit_transform(x_train)
print("number of pcs needed for 90% variance capture:",(np.where(np.cumsum(pca.explaine
d_variance_ratio_) > 0.9))[0][0])
```

```
number of pcs needed for 90% variance capture: 86
```

Q 6

Select one object from the processed COIL20 dataset https://www.kaggle.com/datasets/codebreaker619/columbia-university-image-library (https://www.kaggle.com/datasets/codebreaker619/columbia-university-image-library) Embed the selected object into a two dimensional space using Diffusion Maps- show the results for several values of σ. What is considered a good value for σ in this example?

In [17]:

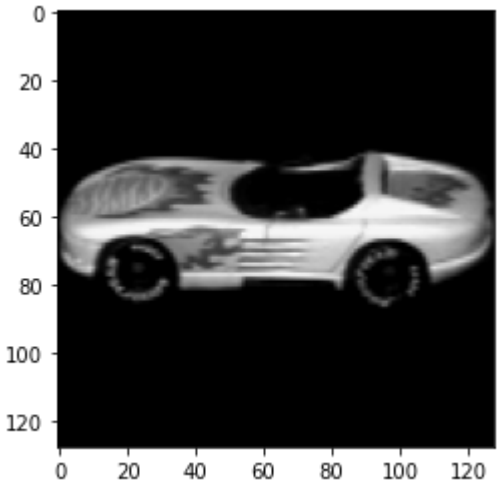
```
def plot_img(image, reshape = None, title = "", gray = True):
    if not isinstance(reshape, type(None)):
        image = image.reshape(reshape)
    if not gray:
        plt.imshow(image)
    else:
        plt.imshow(image, cmap = "gray")
    plt.title(title)
    plt.show()
```

In [18]:

```
coil_path = "coil-20/coil-20-proc"
object_chosen = "obj6_"
filtered_examples = np.array([cv2.imread(path.join(coil_path,obj))[:, :, 0]
                     for obj in listdir(coil_path) if object_chosen in obj])
filtered_examples_flat = filtered_examples.reshape(-1,filtered_examples.shape[1]*filter
ed_examples.shape[2])
```

In [19]:

```
plot_img(filtered_examples_flat[0],(128,128))
```

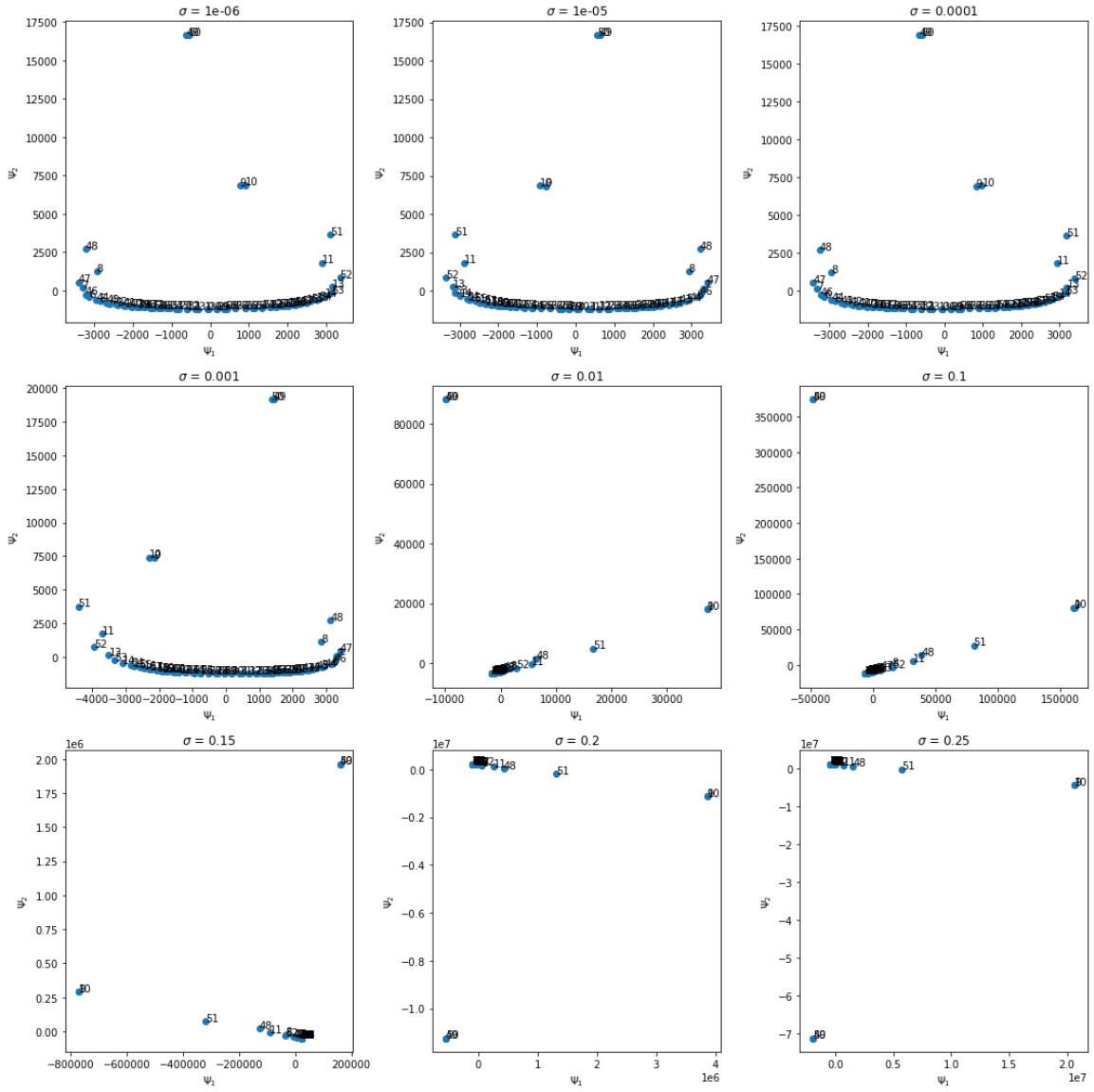


In [20]:

```
def fit_transform_dmap(sigma = 0.0001):
    dmap = dm.DiffusionMap.from_sklearn(n_evecs=2,k=15,bandwidth_type = sigma)
    filtered_examples_embed = dmap.fit_transform(filtered_examples_flat)
    return dmap, filtered_examples_embed
```

In [21]:

```
def plot_image_grid(images, rows, cols, titles = None):
    fig, axs = plt.subplots(rows, cols, figsize=(cols * 5, rows * 5))
    for i, ax in enumerate(axs.flat):
        psi_1, psi_2 = images[i][:,0],images[i][:,1]
        ax.scatter(psi_2, psi_1)
        if not isinstance(titles,type(None)):
            ax.set_title(titles[i])
        ax.set_xlabel('$\Psi_1$')
        ax.set_ylabel('$\Psi_2$')
        for x,y,j in zip(psi_1, psi_2,range(len(psi_1))):
            ax.text(y,x,j)
    plt.tight_layout()
    plt.show()
sigmas = [1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1.5e-1,2e-1,2.5e-1]
sigmas.sort()
images = [fit_transform_dmap(sigma)[1] for sigma in sigmas]
plot_image_grid(images, 3, 3, titles = [f"$\sigma$ = {sigma}" for sigma in sigmas])
```



i would go with a value somewhere between 0.000001 and 0.001 as they seem to capture the rotation, also this value seems to cahnge if we choose a different observation to work with

Q 7

Use TSN-E to embed all objects from COIL20. Run the algorithm twice and compare the results. Are they identical? How can you stabilize the results?

In [34]:

```
object_numbers = [obj.split("_")[0][3:] for obj in listdir(coil_path)]
all_examples = np.array([cv2.imread(path.join(coil_path,obj))[:, :, 0]
                     for obj in listdir(coil_path)])
all_examples_flat = all_examples.reshape(-1,all_examples.shape[1]*all_examples.shape
[2])
s_time = time()
tsne = TSNE()
tsne1 = tsne.fit_transform(all_examples_flat)
tnse_time = time() - s_time
tsne2 = tsne.fit_transform(all_examples_flat)
tsne1 = pd.DataFrame(tsne1, columns = ["t1","t2"])
tsne1["object"] = object_numbers
tsne2 = pd.DataFrame(tsne2, columns = ["t1","t2"])
tsne2["object"] = object_numbers
```

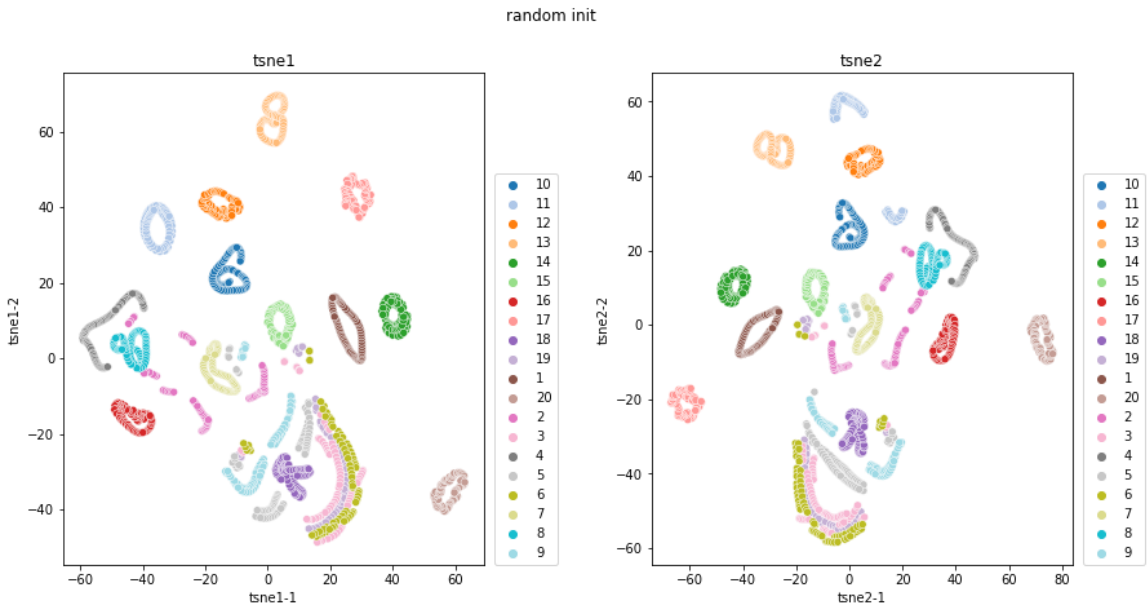
```
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780: F
utureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: F
utureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780: F
utureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: F
utureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
```

In [35]:

```
def plot_2_tsnes(tsne1,tsne2,title="random init",
                 title1="tsne1",xlabel_1= "tsne1-1",y_label_1 = "tsne1-2",
                 title2="tsne2",xlabel_2= "tsne2-1",y_label_2 = "tsne2-2"):
    fig, (ax1, ax2) = plt.subplots(1, 2,figsize = (14,7))

    sns.scatterplot(data = tsne1, x ="t1", y="t2", hue="object", ax = ax1,palette = "tab20")
    ax1.set_title(title1)
    ax1.set_ylabel(y_label_1)
    ax1.set_xlabel(xlabel_1)
    ax1.legend(bbox_to_anchor=(1.025, 0, 1, 1), loc=3,
               ncol=1, borderaxespad=0.)
    sns.scatterplot(data = tsne2, x ="t1", y="t2", hue="object", ax = ax2,palette = "tab20")
    ax2.set_title(title2)
    ax2.set_ylabel(y_label_2)
    ax2.set_xlabel(xlabel_2)
    ax2.legend(bbox_to_anchor=(1.025, 0, 1, 1), loc=3,
               ncol=1, borderaxespad=0.)
    fig.subplots_adjust(wspace=0.4)
    plt.suptitle(title)
    plt.show()

plot_2_tsnes(tsne1,tsne2,title="random init")
```

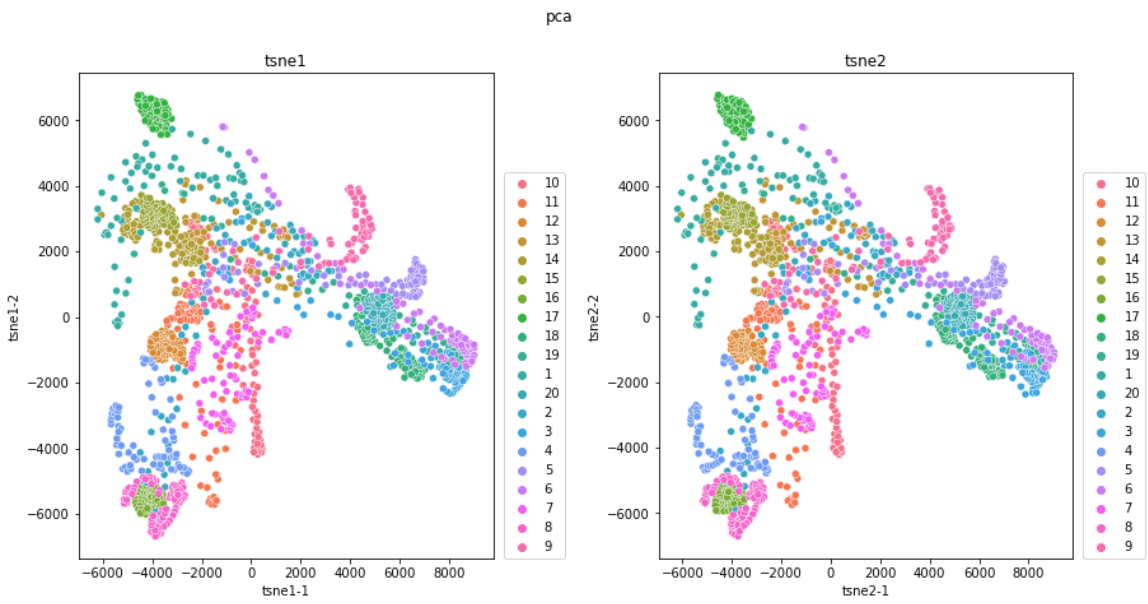


the results are not identical becuase tnse uses random initiation, we can stablize it by setting a seed also we can preform pca

In [24]:

```
tsne = TSNE(init = "pca")
tsne1 = tsne.fit_transform(all_examples_flat)
tsne2 = tsne.fit_transform(all_examples_flat)
tsne1 = pd.DataFrame(tsne1, columns = ["t1","t2"])
tsne1["object"] = object_numbers
tsne2 = pd.DataFrame(tsne2, columns = ["t1","t2"])
tsne2["object"] = object_numbers
plot_2_tsnes(tsne1,tsne2,title="pca")
```

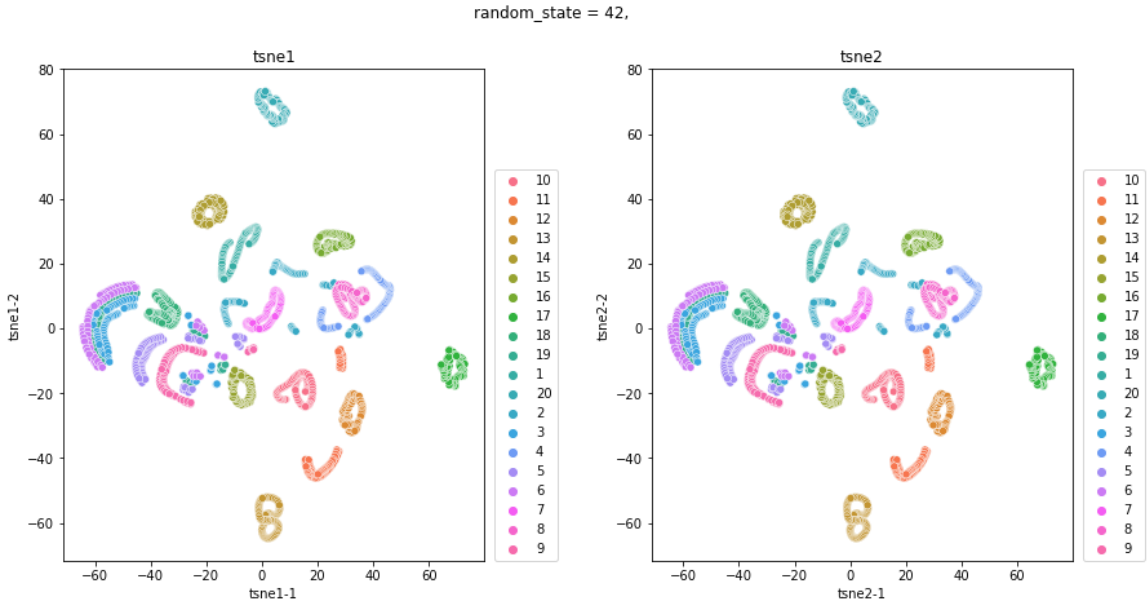
```
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: F
utureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:982: F
utureWarning: The PCA initialization in TSNE will change to have the stand
ard deviation of PC1 equal to 1e-4 in 1.2. This will ensure better converg
ence.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: F
utureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:982: F
utureWarning: The PCA initialization in TSNE will change to have the stand
ard deviation of PC1 equal to 1e-4 in 1.2. This will ensure better converg
ence.
  warnings.warn(
```



In [25]:

```
tsne = TSNE(random_state = 42)
tsne1 = tsne.fit_transform(all_examples_flat)
tsne2 = tsne.fit_transform(all_examples_flat)
tsne1 = pd.DataFrame(tsne1, columns = ["t1","t2"])
tsne1["object"] = object_numbers
tsne2 = pd.DataFrame(tsne2, columns = ["t1","t2"])
tsne2["object"] = object_numbers
plot_2_tsnes(tsne1,tsne2,title="random_state = 42")
```

```
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780: F
utureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: F
utureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780: F
utureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(
C:\Users\Nir\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: F
utureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
```

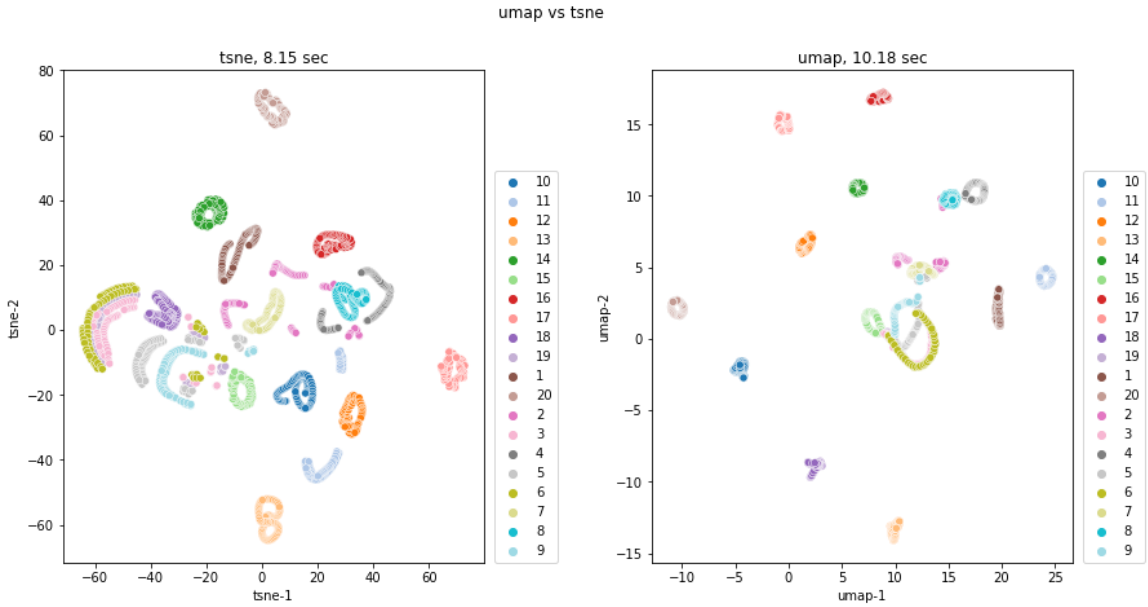


Q 8

Compare the results from the previous section to UMAP. Both in terms of run time and representation quality.

In [33]:

```
s_time = time()
umap_exmp = UMAP(n_components=2)
umap1 = umap_exmp.fit_transform(all_examples_flat)
umap_time = time() - s_time
umap1 = pd.DataFrame(umap1, columns = ["t1","t2"])
umap1["object"] = object_numbers
plot_2_tsnes(tsne1,umap1,title="umap vs tsne",
                title1=f"tsne, {tnse_time:.2f} sec",xlabel_1= "tsne-1",y_label_1 = "tsn
e-2",
                title2=f"umap, {umap_time:.2f} sec",xlabel_2= "umap-1",y_label_2 = "uma
p-2")
```



we can see that TSNE is faster by just a bit. its hard to say which one results in a better representiation, the TSNE embedding seems more dense (the different objects are rather close in the grid) while in the UMAP we can see that each observation has its own location

also if we take a look at the images we will see that UMAP "clusters" boxes with cars (objects 3,6 are cars,5,9 are boxes) and all other objects has thier own location in the embedding space while the TSNE is more spread for example object 2,3,5