**Instructions**

In this problem, you will use the Rock dataset located here: [https://osf.io/d6b9y/Links to an external site.](https://osf.io/d6b9y/).

**Extra tips:**

* If you are using Colab, make sure to select a GPU (even if you're not using a GPU via Edit->Notebook settings, it will give you more memory; no need for the paid version of Colab).
* You can convert images to grayscale from RGB and complete the assignment using only grayscale images.
* Feel free to downsize images if you are facing issues with the memory/running time.

[Links to an external site.](https://osf.io/cvwu9/wiki/Data%20File%20Descriptions/)

**Answer the questions below directly in your Jupyter Notebook. Please answer questions in order and use Markdown cells to clearly indicate which question you are answering.**

1. Apply PCA to the images from folder '360 Rocks'. How many components do you need to preserve 90% of the variance? **[3 points]**
2. Plot 10 images of your choice in the original form (without PCA) and then plot their reconstruction (projection in the original space) after you kept 90% of variance using PCA. **[3 points]**
3. Each of the images belongs to one of three rock categories. The category is indicated by the first letter in the filename (I, M and S). We will now try to see if the visualization can help us identify different clusters.
   1. Use PCA to reduce dimensionality to only 2 dimensions. How much of the variance is explained with the first two principal components? **[2 points].**
   2. Plot a 2D scatter plot of the images spanned by the first two principal components. Each image will be represented with a dot. Make the color of the dot correspond to the image category (so you will have three different colors). Then add some rock images to the visualization to better understand what features in the images are accounting for the majority of variance in the data (your visualization should look similar to the one after line 71 in this file [https://github.com/ageron/handson-ml3/blob/main/08\_dimensionality\_reduction.ipynbLinks to an external site.](https://github.com/ageron/handson-ml3/blob/main/08_dimensionality_reduction.ipynb) but with images of rocks instead of MNIST digits). Repeat the process and create the same type of plots for t-SNE, LLE and MDS. **[6 points]**
   3. Discuss your observations. **[1 point]**
4. Now let's see if these dimensionality reduction techniques can give us similar features to those that humans use to judge the images. File mds\_360.txt contains 8 features for each of the images (rankings are in the same order as the images in  '360 Rocks' folder. Run PCA, t-SNE, LLE and MDS to reduce the dimensionality of the images to 8. Then, compare those image embeddings with the ones from humans that are in the mds\_360.txt file. Use Procrustes analysis to do the comparison (here is one example of how to do that mtx1, mtx2, disparity = procrustes(matrix\_with\_human\_data, matrix\_with\_pca\_embeddings\_data). Here matrix\_with\_human\_data and matrix\_with\_pca\_embeddings\_data should be 360 by 8. disparity will tell you the difference in the data. Report disparity for each of the four dimensionality reduction methods. Compute the correlation coefficient between each dimension of mtx1 and mtx2 for each of the four methods - display results in a table. **[7 points]**
5. Cluster the 360 images using K-Means.
   1. You can reduce the dimensionality using PCA if you wish, but keep at least 90% of the variance. Determine the number of clusters using one of the techniques we discussed in class. **[4 points]**
   2. Set the number of clusters to 3 and report clustering accuracy. **[4 points]**
6. Cluster the 360 images using EM.
   1. You can again reduce the dimensionality using PCA if you wish, but keep at least 90% of the variance. Determine the number of clusters using one of the techniques we discussed in class. **[4 points]**
   2. Set the number of clusters to 3 and report clustering accuracy. **[4 points]**
   3. Use the model to generate 20 new rocks (using the sample() method), and visualize them in the original image space (since you used PCA, you will need to use its inverse\_transform() method).  **[4 points]**
7. Build a feedforward neural network (using dense and/or CNN layers) with a few hidden layers (we suggest using Keras (within Tensorflow) or Pytorch). Train the network to classify on 360 rock images using rock name as the label - the category is indicated by the first letter in the filename (I, M and S). Use images from '120 Rocks' folder as your validation data. Choose the number of neurons you find appropriate and efficient (so you have enough time to run it), but make the last layer before the softmax should consist of 8 neurons. The hidden layers should have ReLU activation function. Train the network for multiple epochs until it converges (if the process is too slow, tweak the learning rate and consider simplifying your network). We will not deduct points based on the simplicity of your network, but we expect you to have performance that is above chance performance that could be obtained with an untrained network - in other words, we expect to see train and validation loss decrease and accuracy increase throughout the training. We recommend using Colab (the free version should be totally fine), but make sure to run it with a GPU to speed up the training - to add a GPU on Colab go to Edit->Notebook settings).
   1. Report the training time (use code to do this). **[1 point]**
   2. Plot training and validation loss and accuracy as a function of training epochs. **[13 points]**
   3. How many parameters does the network have? How many of those parameters are bias parameters? **[1 points]**
   4. Compare the activity of neurons in the next to the last layer (the one with 8 neurons) with the human data. (to get human data use mds\_360.txt and mds\_120.txt files). Similar to before, use Procrustes analysis to do the comparison.  For training and validation data (separately), report disparity and compute the correlation coefficient between each dimension of mtx1 and mtx2. Display results in a table. **[3 points]**