**Dino V2 code explanation**

1. Upload our data on our google drive and then link our google drive to our google collab notebook.
2. Extract all these images from the folder in the collab working directory.
3. The ‘data’ list will contain the file paths of all the PNG files present in the specified directory and its subdirectories.
4. We then create a symbolic link (soft link) to change the default CUDA version from CUDA 12 to CUDA 10.
5. Set up this CUDA on a GPU
6. Add the requirements.txt file to our collab directory, after this we can download all required packages necessary to use DINO embeddings. We can get this from <https://raw.githubusercontent.com/facebookresearch/dinov2/main/requirements.txt>
7. Call the dino model we need to use in order to make carry out the embeddings. In this project we use the “dinov2\_vitg14”, which refers to the Vision Transformer (ViT) version of the DINO v2 model.
8. Pre-process the dataset before feeding it into the model i) Create an instance to allow us to chain multiple transformations together. ii) Resize the images to 222x224 pixels. Iii) normalize the 3d images to pixel values of images. This pre-processing step ensures that the input images are properly resized, converted to tensors, and normalized according to the specified values.
9. A forward pass is performed through the pre-trained DINO v2 model for each image in the dataset. A new list called “embeddings” is made and will contain the output embeddings for each image based on the “model” variable from the previous step and contains the pre-trained DINO v2 model. This list also contains the image file path of each image.
10. The list of PyTorch tensors containing the embeddings is converted to a list of NumPy arrays representing the embeddings of the images.
11. We then save these embeddings to a folder on our collab notebook and have our embeddings which will be used later on to predict novelty in art.

**KMeans code explanation**

* Building the KMeans model

1. Upload our DINO embeddings data on google drive and then link our google drive to our google collab notebook.
2. Extract all these embedding files from the folder in the collab working directory.
3. The ‘data’ list will contain the file paths of all the .npy files present in the specified directory and its subdirectories.
4. Split the data into 80/20 for the training and test datasets. Apply the Kmeans model to the training set.
5. Use principal component analysis for visualization of the clustering results based on the training set.
6. Since DINO is high dimensional, using only the first 2 PCs for visualization did not give clear results. So we used elbow method to determine the optimal value of k.
7. Re-running the kmeans model to the training set based on the optimal value of k.
8. Apply the trained model to generate labels for the test dataset.
9. Plot the bar graph of ‘Kmeans distribution’ to see the distribution of clusters based on the test labels.

* Building the novelty metrics model

1. calculate the average distance of samples to their respective cluster centroid and add the values to a list named “average\_distance\_per\_cluster”. The list contains k values for k average distances.

2. for each test point that belongs to a certain cluster k, if the distance between the test point and its cluster centroid is larger than the value of average\_distance\_per\_cluster[k], it is considered novel. The novel samples are added to the kth element of the “novel\_samples” list.

* Building the surprise metrics model

1. Calculate pairwise distances between centroids in the Kmeans model.
2. Calculate a single value, the average of all the pairwise distances between centroids, named “avg\_dis\_centroid”.
3. Assign a new test sample to the cluster with the shortest distance to its centroid.
4. If the distance between the new sample and its cluster centroid is larger than “avg\_dis\_centroid”, it is essentially in a cluster of its own and is considered surprising. Then the sample is added to the “surprising\_samples” list.