Programming Assignment 2
Locality Sensitive Hashing

*Tentative Deadline: **13th October**

In this assignment, we explore two variants of locality-sensitive hashing that you've studied in class. In the first variant, we rely on randomly generated hyperplanes to develop buckets into which similar items can be put. In the second variant, we try to overcome the limitations of random projection hashing with data-driven neural hashing.

Task: We will be developing an image retrieval system. Given a set of images, split into train and test, we will use the trainset (AKA the gallery set) to develop hash buckets, and images from the test set will form our queries. The goal is to retrieve images from the gallery set that are similar to the queries. We will use the usual ranking metrics: mean average precision, precision@10, and precision@50. To rank results returned by a method, you can compare query image and result image embeddings using cosine similarity (decreasing order) or Hamming distance (increasing order, if vectors are **binary**). Relevance of a result must be defined based on the information you have in the dataset.

Tutorial Link on teams: Tutorial Video for Assignment 2.
Clarification about ranking results in neural LSH (from a student's query): You have L hashtables and you test your query in the buckets of each of them. Your query may hash to a bucket in each of the tables. You take the union of results across these buckets - these constitute your candidate set - and then you can perform ranking as usual.

Datasets: You may use the feature descriptors provided in the datasets, if any, as input feature embeddings for the hashing algorithms. You are also allowed to use pre-trained ResNet like models to extract feature embeddings from images, and compare performance with given features. Note that this assignment is *not* about *training* with ResNet.
The datasets to be used are as follows -:

1. CIFAR-10
2. CIFAR-100
3. ImageNet-1K (ILSVRC 2012 competition version): The original dataset can be downloaded from here (see here if you run into issues). If the original size is too big for you (150 GB), use this version from HuggingFace Hub (it is much smaller in size).

All of these datasets can be loaded using torchvision/HuggingFace Hub. ImageNet-1K is the largest among them, with 1.2 million images.

Methods: We will explore the following methods -:

1. **K-means clustering**: As a start, perform K-means clustering on the gallery set to form clusters analogous to the buckets you would get using LSH. Query this set of clusters using the test set. How does the number of clusters change performance? Does feature dimensionality play a role - try reducing ResNet feature dimensions using PCA or similar and see if K-means does better than before?
2. **Random projection hashing**: In this method, you develop hash buckets by random hyperplane splitting. How well does this method do against K-means? Can it perform on higher feature dimensions like in the features obtained from ResNet? Does the number of hyperplanes affect the result, and if so, how?

3. **Neural LSH #1**: In this method, we learn how to produce hash codes based on the given data. How does this method perform against random projection hashing - is there any advantage that it might have? See sections 4.1 and 4.2 of the paper for an overview of the training method, and appendix section B on the overall LSH strategy; also see the codebase for how appendix B has been implemented. **Hints**: To adapt this method for this assignment (as it is originally used on graphs), think about what each term of the loss function would mean for images. The first two terms of the loss directly work out for image embeddings in the train-set; for the third term, you need to sample negatives for each image in the train-set (how would this be done?).
4. **Neural LSH #2** (**extra credit**): From this paper, pages 6 and 7, try to adapt the training strategy provided here. Note that this is a two-stage training procedure - equations 11 and 12 show this. Think about what the relevance labels mean in the case of images (eqn 11). There is no need to use the Fourier transformations described in the paper - you can directly use the input image features that you were using already.

In each of the cases, report time taken to retrieve objects as well as the performance metrics.

General Hints:
You may use either Google Collab or Kaggle. However Kaggle has a more generous 30 hr per week GPU quota, so please make use of it. Don't forget to terminate GPU usage when you don't need it.

Submission Protocol:
Moodle will be used for the submission of the assignment.
- All the findings need to be put down into a report file.
- Zip all the files used for building/training/testing of the models along with the report into a single file and then submit it on Moodle. Name the compressed file as [Roll1_roll2.(zip/tar.gz)]

Note - Only one member of the team should submit the assignment on Moodle.