CSE 673 Assignment 2- Part 1: Recognition and Retrieval

The objective of this part of the assignment is to get started with a Deep Learning framework and to use it to create object recognition and retrieval systems. The primary goal of the assignment is to do as well as possible on the image classification problem (on the Tiny ImageNet dataset) and on the image retrieval problem (on the cars 196 dataset) using convolutional neural networks.

Datasets:

- TinyImagenet:
  - Tiny Imagenet has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. All images are 64 X 64. You can download the dataset from the link below Link

- Cars196:
  - The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. You can download the dataset from the link below link

Deep Learning Framework and Infrastructure:

You are allowed to use any of the following Deep Learning frameworks (The associated link will have starter codes available)

- Pytorch, mnist example
- Keras, mnist example
- Tensorflow, mnist example

Since the objective is to get familiarized with any one of the frameworks, we would want you to write all the functions yourself. We won't allow you to use codebases from GitHub., Machine learning mastery, Kaggle or other similar websites. If you have used any small helper functions available online (like stack overflow), please ensure you cite it. You will require a GPU compute to complete the assignment. So make sure you have Google Colab or similar compute facilities.

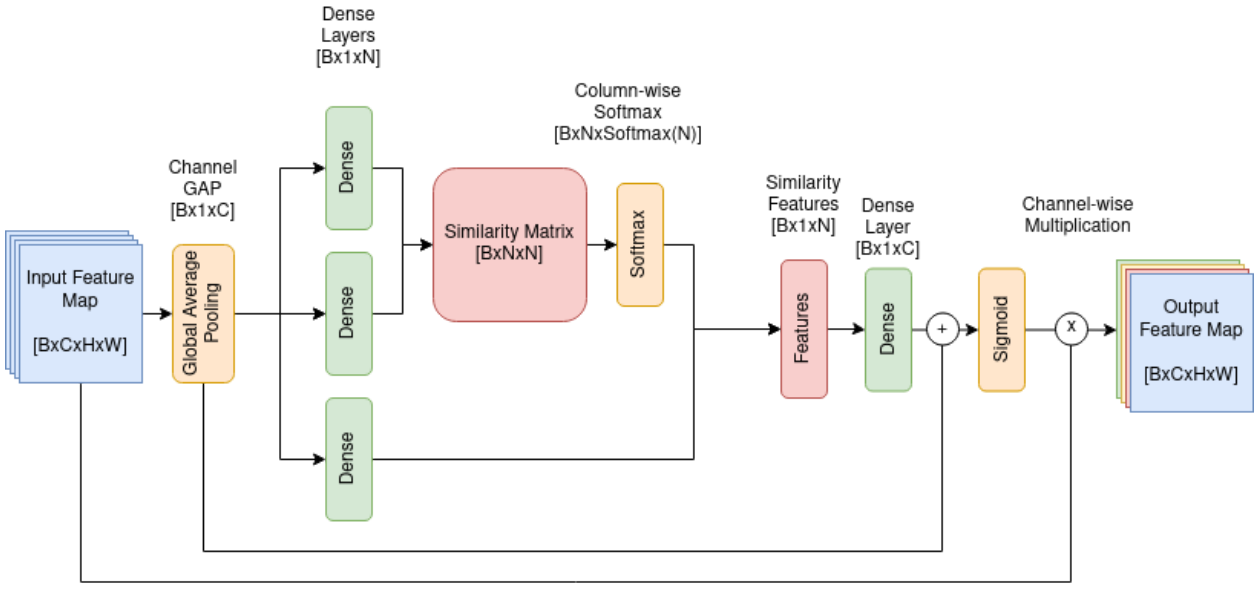
Tasks and Evaluation:

- Task 1 (T1): Train a custom CNN network of your choice on the Tiny Imagenet dataset and report the accuracy.
  - The choice of network is up to you.[Don't use any predefined architecture like Resnet or VGG etc. for this task, **Design your own**]

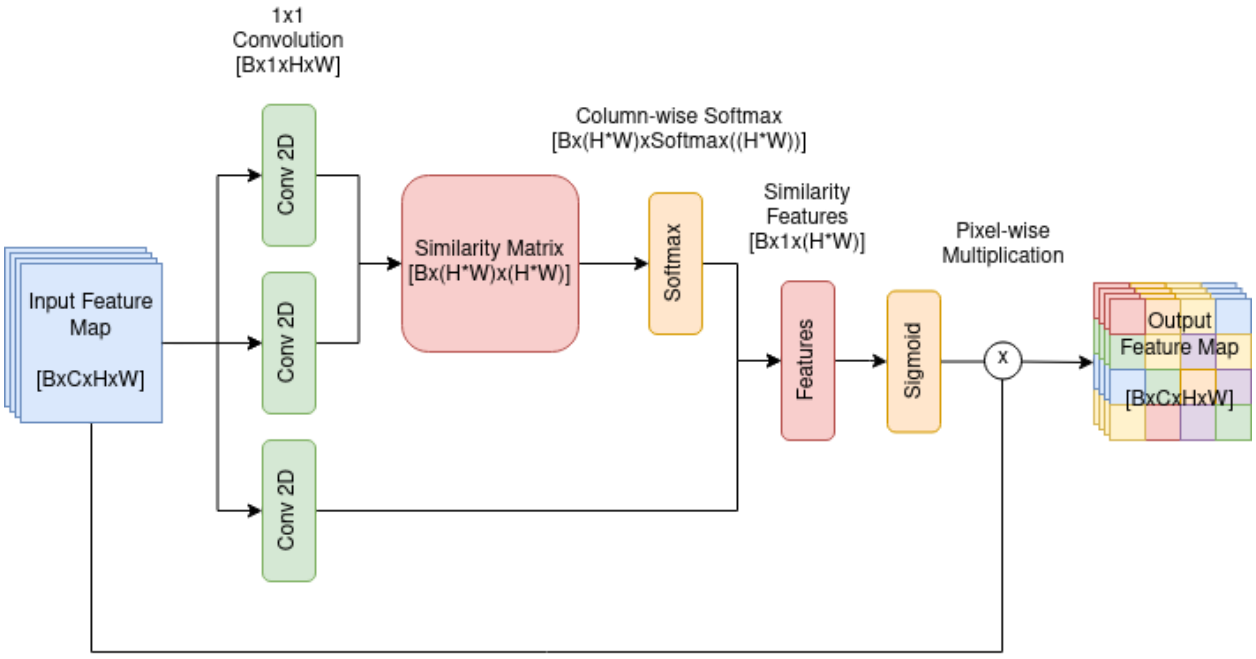
- Make sure your model trains and produces more than 40% accuracy on the Tiny Imagenet test set.
- Proper use of dataset generators/data loaders, to only load images batch by batch is a must (Don't use predefined API like the flow from the directory, etc.)
- Use of data augmentation, Early stopping, Learning decay (or schedule), and model checkpointing (you can use available callbacks or APIs for this) is a must.
- Train the model using cross-entropy loss.

- T2: Train Resnet 18 model on Tiny Imagenet dataset.
  - Make sure that you don't take the pre-trained model, but initialize it from random.
  - If T1 is done properly, the only change is to use the Resnet18 model instead of your custom model
  - The Resnet 18 implementation is here link .

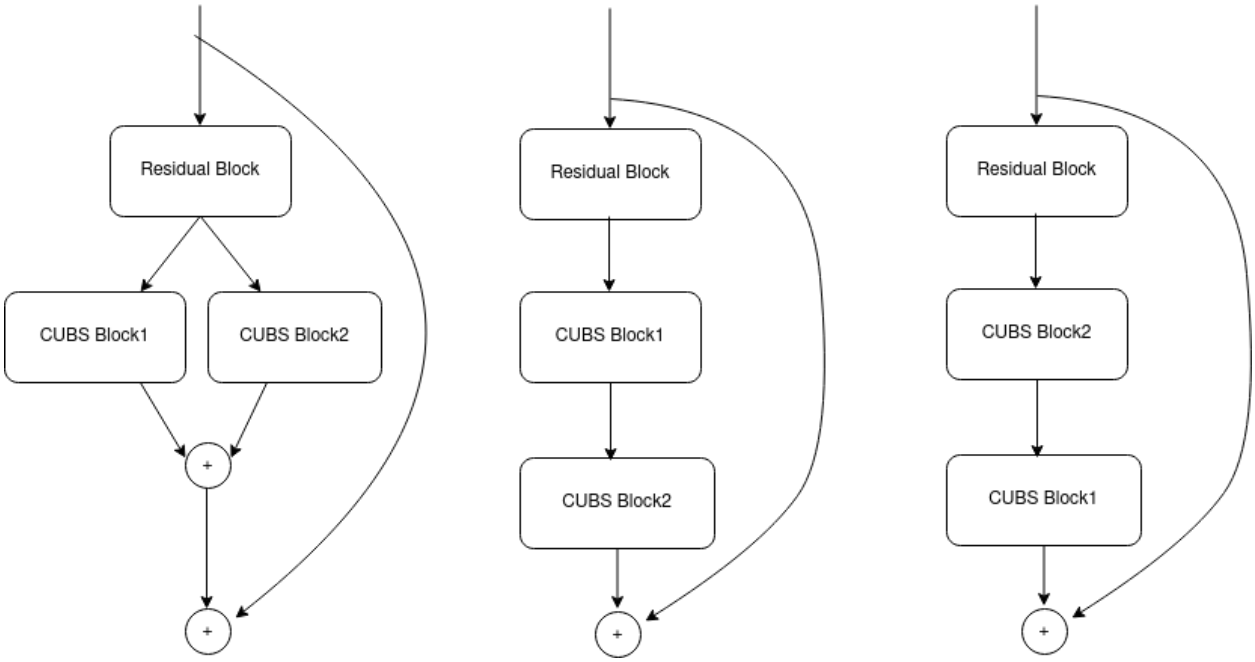
- T3: Augment resnet 18 with the following modules:
  - CUBS Block 1



- CUBS Block 2



- Add these new modules after each residual block in one of the following fashions:



- Use the design which gives you better accuracy.
- The implementation of the module is really tricky, make sure that you understand the dimensions and create the module correctly
- With this module the architecture should perform at par or better with the vanilla resnet (max drop allowed -2%)
- Train this new architecture of tiny imagenet and report the accuracy.

- T4: Implement retrieval task using Augmented Architecture.
  - Using the new resnet architecture, train the model on the first 98 classes of the cars196 dataset. The rest 98 classes are the test set.
  - Create a function to compute recall (mentioned below) and report recall at 1,2,4,8
    - Recall @K = Number of relevant items in top K / Total number of items
  - After the recalls have been recorded, crop the classification layer and add a new single 512 dimensional embedding.
  - Train the whole network using the custom loss algorithm given below
    - For each batch find the similarity of samples
    - For each sample find other positive samples and negative samples
    - Compute hard positives and hard negatives (can innovate here!)
    - Use the below formulation to compute the loss

$$\mathcal{L} = \frac{1}{BatchSize} \sum_{i=1}^{BatchSize} \left\{ \frac{1}{\alpha} \log \left[1 + \sum_{positive \in \mathcal{P}ositives_i} e^{-\alpha(Positivesimilarity - \lambda)} \right] + \frac{1}{\beta} \log \left[1 + \sum_{negative \in \mathcal{N}egatives_i} e^{\beta(Negativesimilarity - \lambda)} \right] \right\}$$

- After training report the recall at 1,2,4 and 8

Evaluation:

We will be looking for the following while evaluating the code:

- Modularized and readable python code using one of the above mentioned deep learning frameworks.
- Following are the grade distribution
  - Task 1 - 10 Points
  - Task 2 - 10 Points
  - Task 3 - 40 Points
    - Implementing Block one - 10 Points
    - Implementing Block two - 10 Points
    - Overall training and getting performance numbers - 10 points
    - Selecting best design out of the three using accuracy numbers - 10 points
  - Task 4 - 30 Points
    - Implementing the loss function accurately - 15 points
    - Innovative ideas to find hard and semi-hard positive and negatives - 10 points
    - End to end training and reporting numbers - 5 points
  - Report - 10 Points
    - A two-page document explaining the network architecture and reporting performance measures like accuracy, loss, etc., and other strategies used in training like hard negative mining, etc.
  - Bonus - 20 points
    - An interesting modification to the module architecture creating a significant performance boost in performance (detailed in report) -10 points
    - If your group finishes with accuracy that is in the top 3 of the class, for the task of object recognition - 5points

    - If your group finished with R1 that is in the the top 3 of the class, for the task of image retrieval - 5 points

Submission Instructions

You will submit Assignment2.zip or Assigment2.tar.gz, a compressed archive file containing the following files:

- Python code for the tasks (training and evaluation code)
- Best model, that gave the performance that you have reported (for each of the tasks)
- Report

Submission is due 10/31/2021,Sunday, 11:59 PM EST. Please use the submit_cse673 script in Timberlake to submit your assignment.