CSE 673 Assignment 2- Part 1: Recognition and Retrieval

Name: **Karan Bali**
UBIT Number: **50381691**
UBIT Name: **kbali**

Datasets Assumptions:

1. Tiny-ImageNet-200:

For Tiny-ImageNet-200, I used “wget” command to download the dataset in the current working directory. Thus, my code assumes there’s a “`./tiny-imagenet-200/`” folder in the current directory with all the rest of required folders & files.

2. Cars-196:

For Cars-196, I download the dataset in a “./cars/” folder in the current working directory. Thus, my code assumes there’s a “`./cars/[Rest of files & folders]`” folder in the current directory with all the rest of required folders & files (especially training folder & annotation file).

NOTE 1)

As the training was taking a lot of time, I had to interrupt the training before it could be completed. Thus, there will be many “Key Interrupt” errors in the logs of Training cells.

But the logs will have the printed “Testing Dataset Accuracy” after each epoch before the training was interrupted.

As due to paucity of time & limitations of “GOOGLE COLAB”, the training has to be stopped. But in many cases the accuracy could be improved further by training the model for more time as the accuracy increment after each epoch hasn’t plateaued.

NOTE 2)

All files have 2 cells in the end to train the particular model. The 1st cell is used to train the model by resuming the training from a checkpoint. The 2nd cell is used to train the model from scratch.

Code files:

NOTE 1)

There are 2 folders (“Python”, “ipynb”) inside the main folder. Both folder contain same files given below, except that “Python” contains “.py” version & “ipynb” contains the “.ipynb” version

---

- Task_1.ipynb:

In this file I have trained a custom nn.module “MyModel” on “Tiny-ImageNet” dataset. My model is similar to a very simple “Hourglass” type model but in place of having series of convolution & Upsampling, “MyModel” uses multiple pairs of a single convolution layer & a single Upsampling layer attached in a serial manner. The number of skip connections are also modified & used only before the Upsampling layer.

Accuracy on Testing dataset: **`27.4%`**
Average Loss on Testing dataset: **`3.426925`**

- Task_2_Resnet.ipynb:

In this file I have trained an open-sourced implementation of ResNet-18 on the “Tiny-ImageNet” dataset.

Accuracy on Testing dataset: **`51.8%`**
Average Loss on Testing dataset: **`2.191711`**

- Task_3_Model_A.ipynb:

In this file I have trained a custom nn.module “Model_A” on “Tiny-ImageNet” dataset. My model uses “CUBS Block-1” & “CUBS Block-2” joined parallely inside the residual block of Resnet-18.

Model_C : CUBS Block-1 | parallely connected | CUBS Block-2

NOTE: Due to paucity of time, I had to stop training model earlier. In one of the earlier training loop, it had provided me with an accuracy of more than 45-50%. I forgot to save the weights that time. If the model is trained for longer duration, it will reach more than 45-50% accuracy.

Accuracy on Testing dataset: **`44.0%`**
Average Loss on Testing dataset: **`2.372956`**

- Task_3_Model_B.ipynb:

In this file I have trained a custom nn.module “Model_B” on “Tiny-ImageNet” dataset. My model uses “CUBS Block-1” & “CUBS Block-2” joined serially inside the residual block of Resnet-18.

Model_C : CUBS Block-1 ---> CUBS Block-2

NOTE: Due to paucity of time, I had to stop training models earlier. In one of the earlier training loop, it had provided me with an accuracy of more than 50%. I forgot to save the weights that time. If the model is trained for longer duration, it will reach more than 50% accuracy.

Accuracy on Testing dataset: **`48.1%`**
Average Loss on Testing dataset: **`2.216512`**

- Task_3_Model_C.ipynb:

In this file I have trained a custom nn.module “Model_C” on “Tiny-ImageNet” dataset. My model uses “CUBS Block-1” & “CUBS Block-2” joined serially inside the residual block of Resnet-18.

Model_C : CUBS Block-2 ---> CUBS Block-1

NOTE: Due to paucity of time, I had to stop training model earlier. In one of the earlier training loop, it had provided me with an accuracy of more than 50%. I forgot to save the weights that time. If the model is trained for longer duration, it will reach more than 50% accuracy.

Accuracy on Testing dataset: **49.3%**
Average Loss on Testing dataset: **2.182806**

Best Model from Task-3:

As I mentioned above, all of Task-3 Models couldn't reach their past-run accuracy goals due to paucity of time.

Both **Model B & C** were equally good (i.e. in the past-run, both provided **>50% accuracy**). But **Model_C** was slightly better.

Model A was in third and last place. It also provided around **45-50 %** accuracy in the past run.

In short all three models were equally good, but **Model C** was slightly better.

- Task_4.ipynb:

In this file I have trained a custom nn.module "Model_C" on the "Cars-196" dataset. My model uses "CUBS Block-1" & "CUBS Block-2" joined serially inside the residual block of Resnet-18.

Model_C : CUBS Block-2 ---> CUBS Block-1

Recall before using Custom Loss:

```
Recall at 1:  0.0747800586510264
Recall at 2:  0.060117302052785926
Recall at 4:  0.047653958944281524
Recall at 8:  0.038581378299120235
```

Recall after using Custom Loss:

```
Recall at 1:  0.08504398826979472
Recall at 2:  0.06867057673509286
Recall at 4:  0.05510752688172043
Recall at 8:  0.04536290322580645
```

Strategies Used:

- Training: At first, I trained on resolution of **64x64** for Task 1,2,3, after that re-trained them on resolution transformation of **224x224**. I also used the Early stopping mechanism & Learning rate scheduler. For normalization in transformers, I used the standard “ImageNet” Mean & standard deviation values.

- Hard Negative Mining: Firstly, I computed a similarity matrix for the whole test dataset. Then i sorted the similarity matrix along one dimension, so that each index first (0’th) dimension gives me a list of sorted similarity scores. Then I use a threshold (½ for the Task_4). The threshold filters out prospective hard negatives & positives from the sorted similarity list for each index. Thus to find “Hard positives”, I search in the lower part (based on threshold ‘k’) of the sorted (descending order) list. And to find “Hard negatives”, I search in the upper part (based on threshold ‘k’) of the sorted (descending order) list. Depending on the batch size, choose ‘k’ so that you can get both Hard positives & negatives.

****Note: "util" function inside Task_4.ipynb is used to mine "Hard Negative/Positive Samples"****
