Notes :-

1. **The commented codes are the ones which I used somehow but not needed with the final run.**
2. **I didn't use CMakeLists.txt for building the CPP code. Used torch's cpp_extension for the same.**
3. **I didn't work on CUDA implementation of mean-reduction because reduction on CPU itself is very fast.**
4. **I used pylint and cpplint to improve my code to match google standards**
5. **Checked for GPU and CPU memory leaks**

- I started by writing the implementation in pytorch and get the similarity scores for each image against other 50 images including itself. The similarity scores were very high so I became cautious and had a look at images. Then I understood that images are actually very similar. All the values of the 50*50 similarity scores can be checked in the scores.npy file.

Some of the entries are the following -

[[1. 0.89286464 0.97714585 ... 0.90900588 0.97315782 0.95990586]

[0.89286464 1. 0.89924455 ... 0.94708997 0.91927212 0.93780541]

[0.97714585 0.89924455 1. ... 0.90903234 0.96538097 0.95814002]

...

[0.90900588 0.94708997 0.90903234 ... 1. 0.92812967 0.92051244]

[0.97315782 0.91927212 0.96538097 ... 0.92812967 1. 0.96845007]

[0.95990586 0.93780541 0.95814002 ... 0.92051244 0.96845007 1.]]

- Now the custom ONNX reduction operation had to be written. I wrote the implementation following the tutorial shared in the challenge document. I matched the similarity scores with custom reduction vs Python-based reduction operation. The scores matched, which ensure that implementation is right. Now I have to look for the computation time.

- I calculated time to run 3 **repeatInterleave** and mean-reduction parts separately, and found that **repeatInterleave** was very relatively slow (0.01 secs for each image). Then I used float array instead of torch tensor in the for loop and now the time taken is 1e-4 range.

- The reduction operation in python on GPU is actually a little slower than custom reduction on CPU.

Doubts - I had some doubts for which I tried reaching you (Niranjan) but somehow you were not available.

1. I calculated time to run 3 repeatInterleave and mean-reduction parts separately for python and CPP code. The mean-reduction was already very fast and was taking time in 1e-5 range. But in the challenge document, it is commented that mean-reduction part is expensive on CPU.

2. Why outputs of repeatInterleave operations need to be concatenated (mentioned in the paper), instead we can simply take the mean?