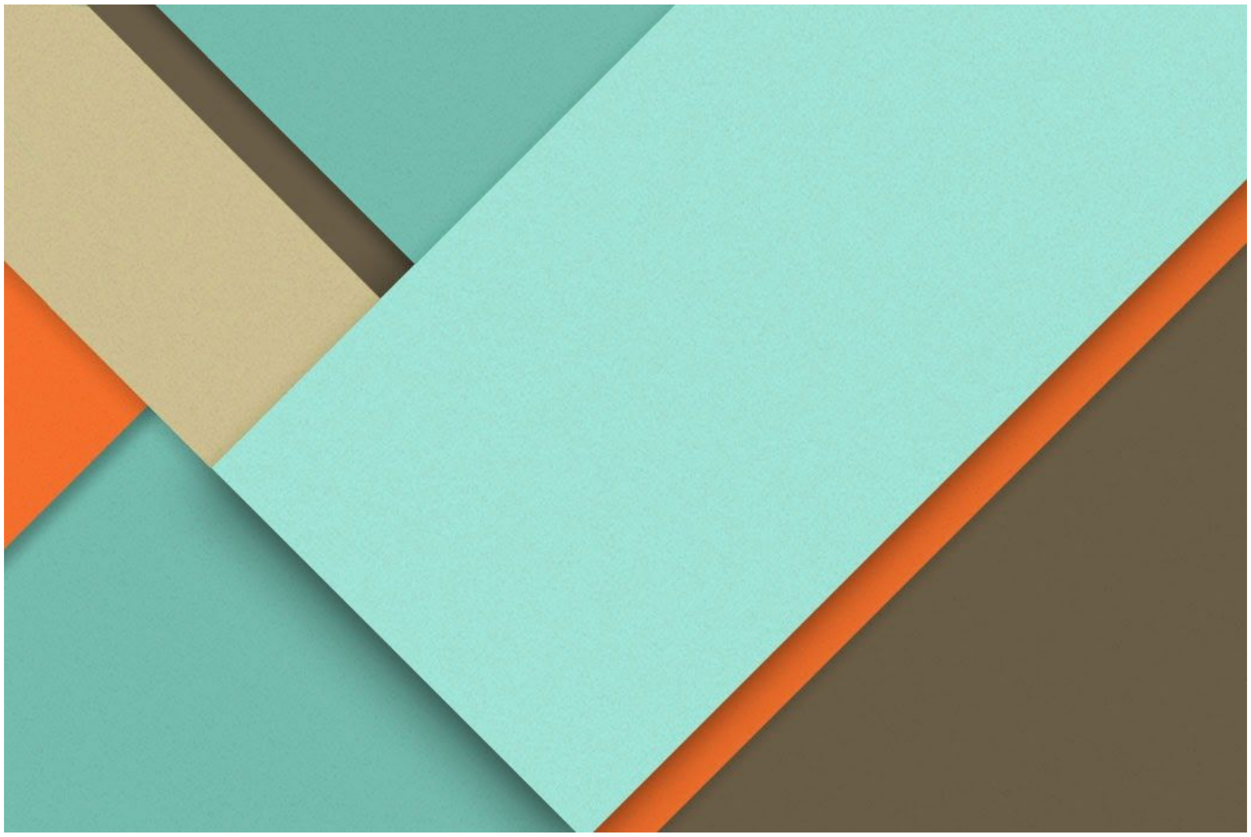


Divide and Conquer the Embedding Space for Metric Learning

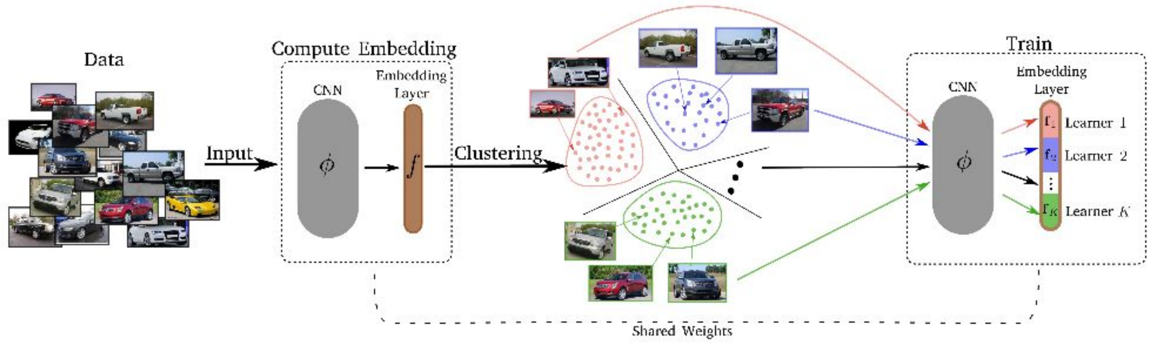
Team Strawhat

Ajay Shrihari (20171097), Aniket Mohanty (20171150), Amarthya Sasi Kiran (20171110)

Overview

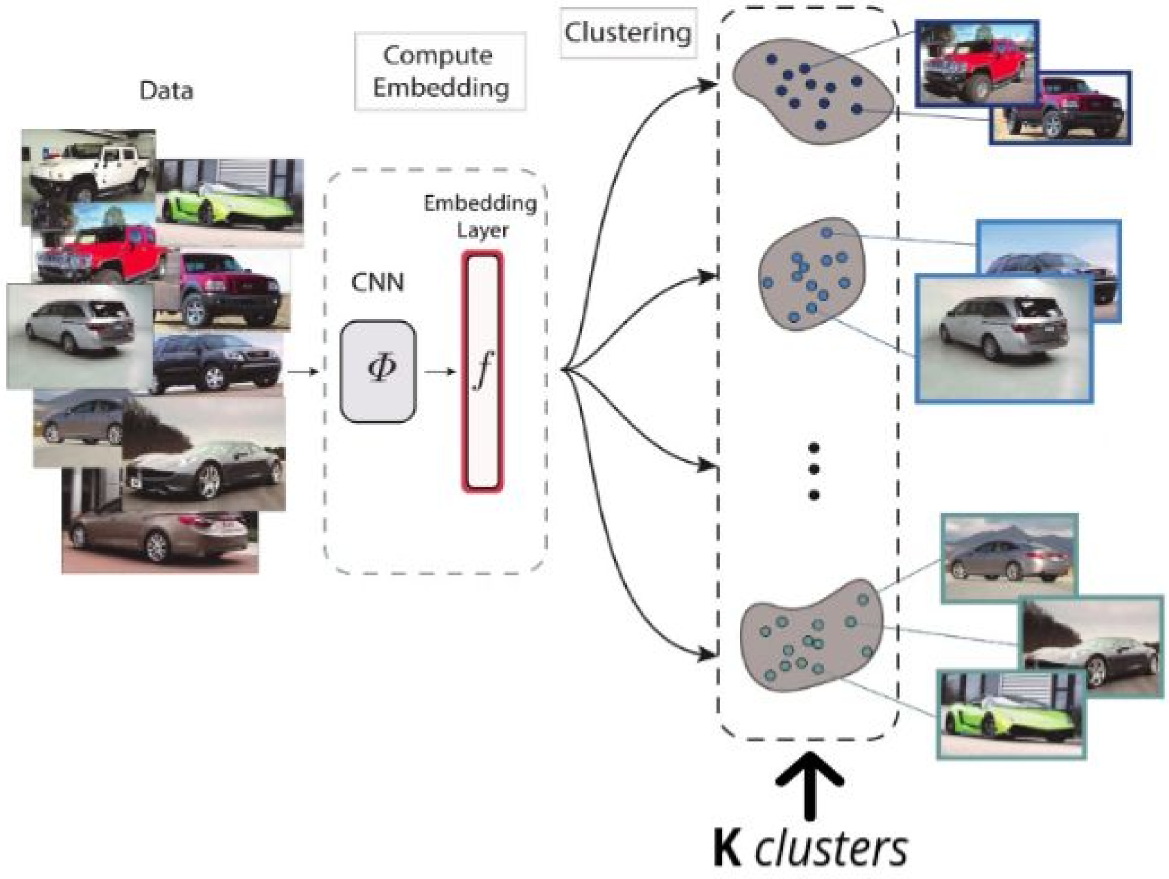
In an embedding space, we generally try to learn one distance metric and use that to train the given data. The issue with this is that it results in overfitting since the embedding space can be very large and based on a single distance metric, the model does not generalize well. ‘Divide and Conquer the Embedding Space for Metric Learning’ (CVPR 2019), aims to solve this problem by dividing the metric space into non-overlapping subsets of data and applying separate learning for each by an ensemble method.

Method



1. The embeddings for the training images are computed and then clustered into K-clusters (disjoint subsets).
2. The d-dimensional embedding spaces are then split into K subspaces of d/K dimensions each.
3. Next, for each of the subspaces, a separate loss is assigned and trained. Different weights are learned for different subspaces, and we get K different distance metrics for the disjoint learners.
4. This process is repeated every T-epoch.

Progress



Currently, we have used a pretrained ResNet 50 model along with extra fully connected layers in order to get embeddings. We have implemented custom modules for random sampling and semi hard negative mining, and have implemented regular metric learning on the Stanford Online Products Dataset.

In random sampling, we observed a low triplet loss while training, suggesting the following:

- In the acquired sample, the negative examples are very close to the margin.

- The positive examples are very close to the anchor.

Hard negative mining considers the negative examples very close to the anchor. But experiments have shown that hard negative mining converges to local minima in the first few iterations.

Hence, we implemented Semi hard negative mining, which does the following:

- Considers the negative examples, closer than the furthest positive example in the sample.
- Considers the positive examples, far away from the nearest negative example in the sample.

The Triplet loss expression is given below, which tries to pull positive examples closer to the anchor, and negative examples away from the anchor. This has been used as our loss function, based on which we have trained our model.

Triplet Loss = max(0 , [$d_+ - d_- + \alpha$])

We have implemented a Normalized Mutual Information (NMI) score, which scales results between 0 (no mutual information) and 1 (perfect correlation) to evaluate clustering.

Challenges faced

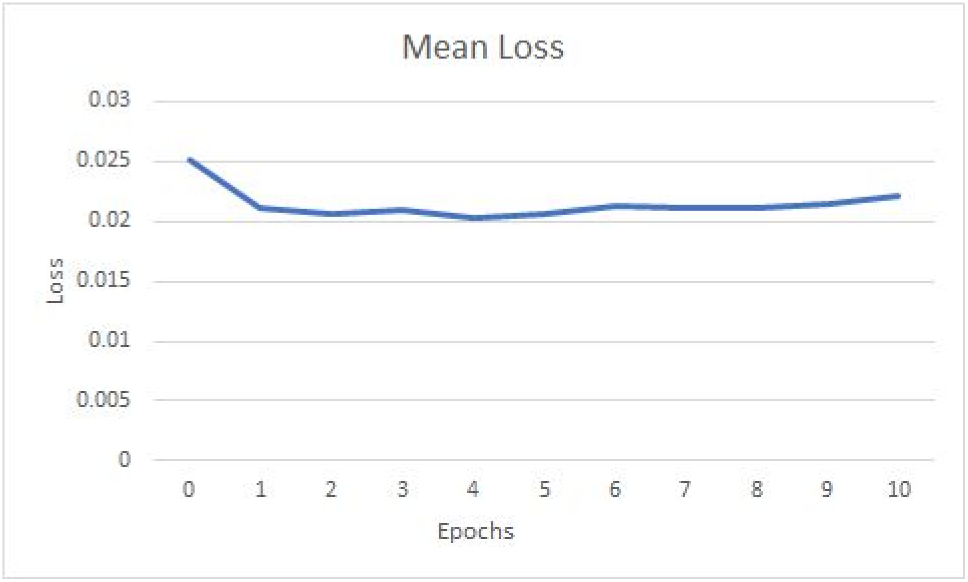
During the testing period, we first started out with the KMeans Clustering in the Sklearn package. We found that this is very slow for an entire pass on the test data used. Hence, we implemented the clustering using the GPU accelerated library ‘FAISS’ by Facebook Engineering.

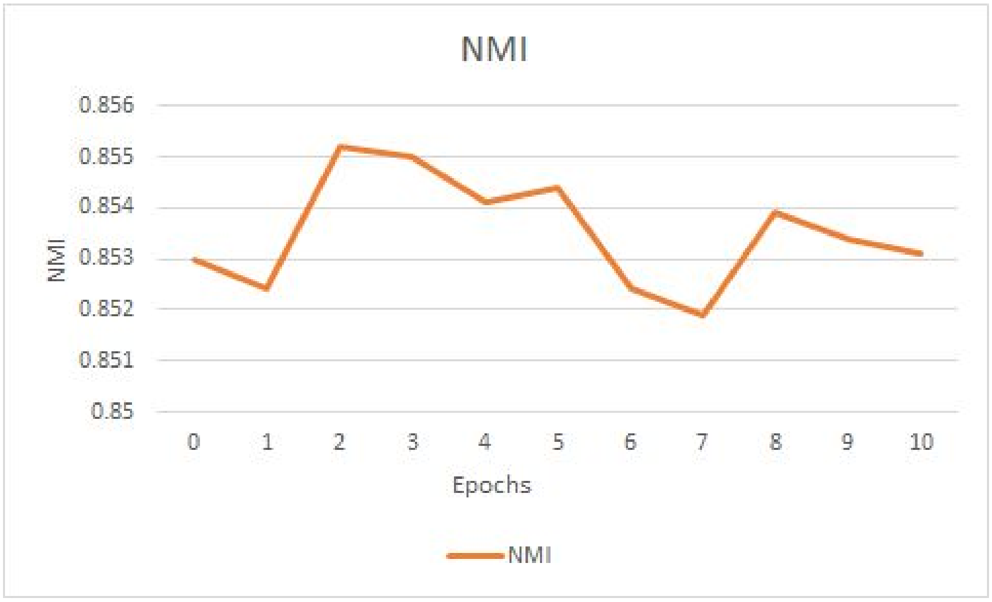
Link to an explanation of FAISS
https://engineering.fb.com/data-infrastructure/faiss-a-library-for-efficient-similarity-search/

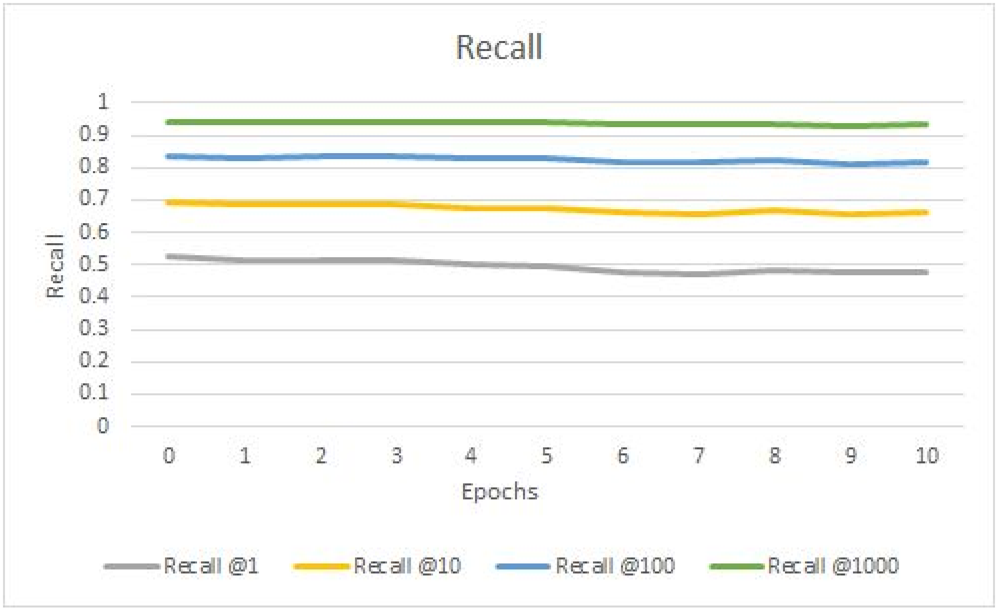
Results

a. We trained our model over 10 epochs using random sampling and triplet loss, and plotted the results: mean loss, NMI, and recall@k.

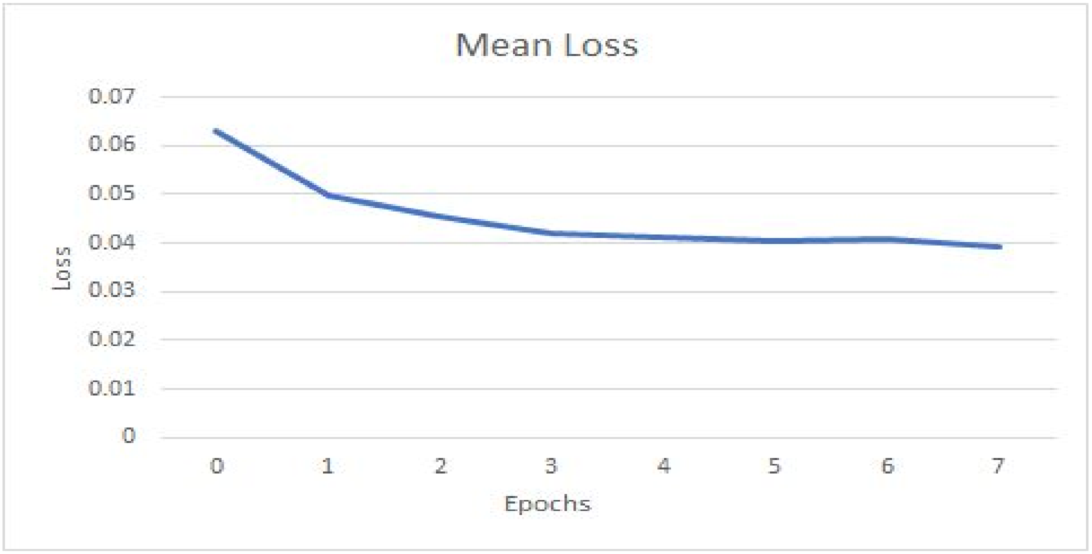
Epoch	Mean Loss	NMI	Recall @1	Recall @10	Recall @100	Recall @1000
0	0.0252	0.853	0.5236	0.6947	0.8366	0.9418
1	0.0212	0.8524	0.5123	0.6852	0.8294	0.9393
2	0.0206	0.8552	0.5158	0.6893	0.8341	0.9394
3	0.0209	0.855	0.5132	0.6866	0.8323	0.9391
4	0.0203	0.8541	0.4987	0.6755	0.8295	0.9424
5	0.0207	0.8544	0.4963	0.6768	0.8274	0.9388
6	0.0213	0.8524	0.4792	0.6598	0.8193	0.9329
7	0.0211	0.8519	0.4718	0.6554	0.8137	0.9331
8	0.0211	0.8539	0.4832	0.6678	0.8217	0.9354
9	0.0215	0.8534	0.4757	0.6568	0.8123	0.926
10	0.0221	0.8531	0.4772	0.6599	0.8175	0.9342

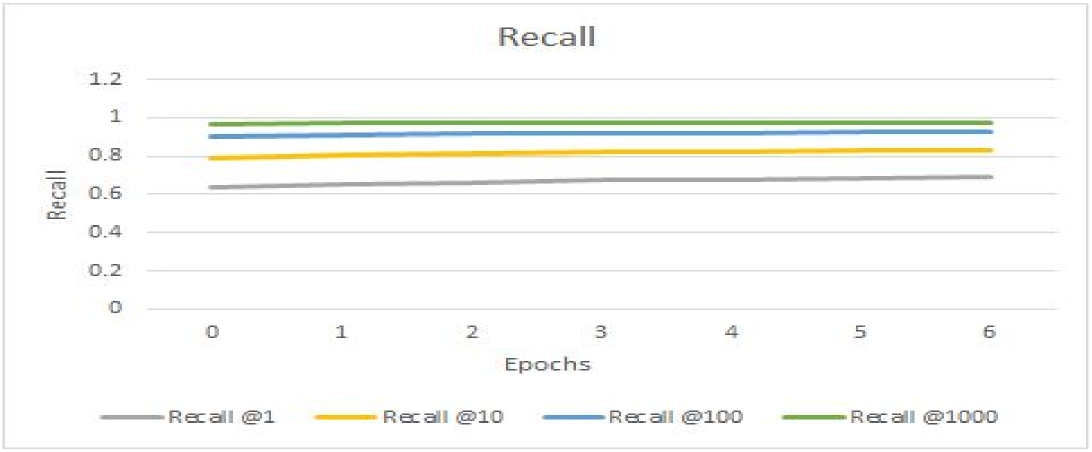


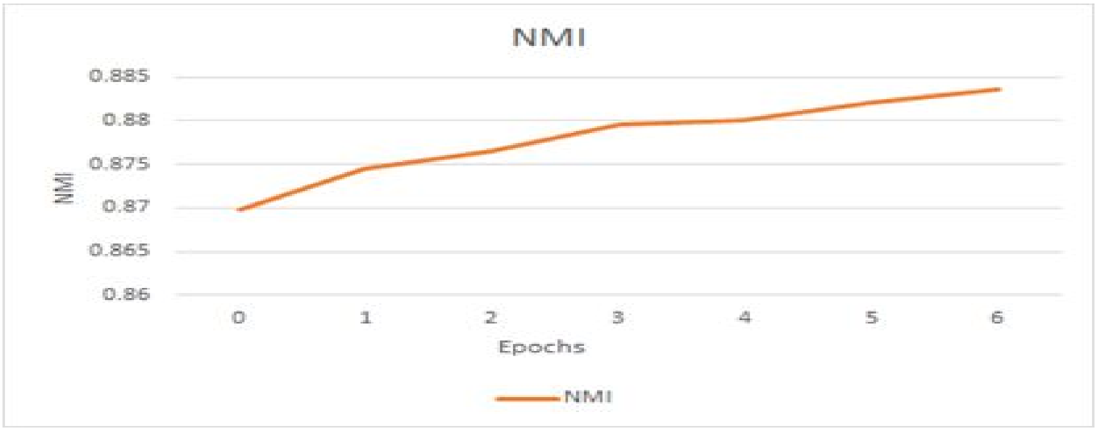

NMI
0.856
0.855
0.854
0.853
0.852
0.851
0.85
NMI
0
1
2
3
4
5
6
7
8
9
10
Epochs
NMI


Recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Recall
0
1
2
3
4
5
6
7
8
9
10
Epochs
Recall @1
Recall @10
Recall @100
Recall @1000

b. Trained our model over 7 epochs using semi hard negative mining and triplet loss, and plotted the results: mean loss, NMI, and recall@k.







Next Steps

The next step to be enacted in the project is dividing the embedding space into K learners, and training each learner in conjunction with the ResNet50 model separately. After a certain number of iterations, we resample the points and restart the process until the model converges to a good output.