이미지들의 Descriptor를 추출하기 위한 저의 알고리즘은 다음과 같습니다.

1. 각 image들의 Descriptor 200개를 랜덤으로 추출한다. =>총 200,000개의 Descriptor

해당 내용의 코드는 아래와 같습니다. (119번째 Line)



2. 200,000개의 Descriptor들에 대해서 K\_means Clustering을 진행합니다. Cluster의 개수는 8개입니다.

해당 내용의 코드는 아래와 같습니다. (127번째 Line)



3. VLAD 알고리즘을 활용하여 각 이미지의 descriptor들과 clustering center와의 차이를 해당 이미지의 Descriptor로 저장합니다.

해당 내용의 코드는 아래와 같습니다. (134번째 Line)



그 후에는 ‘my\_Des’를 eval 디렉토리에 binary file 형태로 저장합니다.

위의 알고리즘은 수업시간에 교수님께서 설명해주신 알고리즘과 동일하고, 제가 구현해본 알고리즘 중에 성능이 가장 우수했습니다.



성능을 개선해보고자 제 나름대로 구현해본 알고리즘들을 설명하겠습니다.

제가 생각한 위 알고리즘의 한계점은, 각 이미지에서 200개의 Descriptor를 추출할 때, 랜덤하게 추출한다는 점입니다. 즉, 이미지의 Descriptor들의 분포를 고려하지 않고 임의로 추출하기에, 해당 이미지를 잘 표현하지 못하는 Descriptor가 포함되어 있을 확률이 크다고 생각했습니다. 이 점을 보완하기 위해, 각 이미지로부터 Descriptor들을 추출하기 전에 Clustering을 이용하여 각 image를 잘 표현하는 Feature extraction 방법에 대해 고안했고, 두 가지 알고리즘을 설계하고 구현했습니다.

제가 구현한 알고리즘의 종류는 이렇습니다.

1. 각 이미지 descriptor들의 Cluster를 그대로 해당 이미지의 Descriptor로 이용한다.

2. 각 이미지 descriptor들의 Cluster를 구하고, 구한 Cluster와 가까운 점들만 추출하여, 100,000개의 Descriptor들을 모으고, 위와 동일한 알고리즘을 적용한다.

각 알고리즘을 자세하게 설명하겠습니다.

**<1. 각 이미지 descriptor들의 Cluster를 그대로 해당 이미지의 Descriptor로 이용한다.>**

1. 각 image의 Descriptor들의 Clustering center를 구한다. K\_means를 사용하여 8개의 clustering center를 구한다.

2. 이 Cluster center를 그대로 각 image의 Descriptor로 저장한다.

각 이미지를 Descriptor들의 Cluster point가 각 image의 Descriptor들을 가장 잘 표현할 것이라고 생각했습니다. 그 결과는 다음과 같았습니다.



위의 알고리즘과 동일한 정확도를 얻었습니다. 정확하게 동일한 이유는 잘 모르겠지만, 해당 알고리즘도 image들을 잘 표현하는 것이라고 생각할 수 있습니다.

**<2. 각 이미지 descriptor들의 Cluster를 구하고, 구한 Cluster와 가까운 점들만 추출하여, 100,000개의 Descriptor들을 모으고, 위와 동일한 알고리즘을 적용한다.>**

1. 각 image의 Descriptor들의 Clustering center를 구한다. K\_means를 사용하여 10개의 clustering center를 구한다.

2. 1번에서 구한 cluster center와 가장 가까운 100개의 descriptor들을 추출한다.

3. 모든 이미지로부터 100개씩 추출하므로, 총 100,000개의 Descriptor를 추출하게 된다.

4. 그 후로는 가장 처음 설명했던 알고리즘과 동일한 알고리즘을 적용한다.

해당 알고리즘을 구현한 코드는 additional의 A3\_make\_Des\_cluster\_8.py 에 있습니다.

이미지로부터 Descriptor를 추출할 때 랜덤하게 추출하는 것이 아닌, cluster center와 가까운 Descriptor들을 추출함으로써, 이미지 Descriptor들의 분포를 잘 대표하도록 Feature extraction을 진행했습니다.

위의 알고리즘들보다 좋은 성능을 보여줄 것이라 기대했으나, 제가 설명한 알고리즘 중 가장 좋지 않은 성능을 보여줬습니다.



해당 알고리즘을 구현한 코드는 additional의 A3\_make\_Des\_Insiders.py 에 있습니다.

감사합니다.