1. 구현

1) sift100000~sift101000의 1000개의 파일을 읽는다.

2) 각 파일마다 평균값을 계산하여 초기 center로 한다. Center의 개수는 1000개가 된다.

3) 1000개의 이미지의 feature의 개수는 총 760512개이고 각 feature마다 어떤 center와 가장 가까운지 계산한다.

4) 이 때 거리의 계산은 np.linalg.norm을 이용한다.

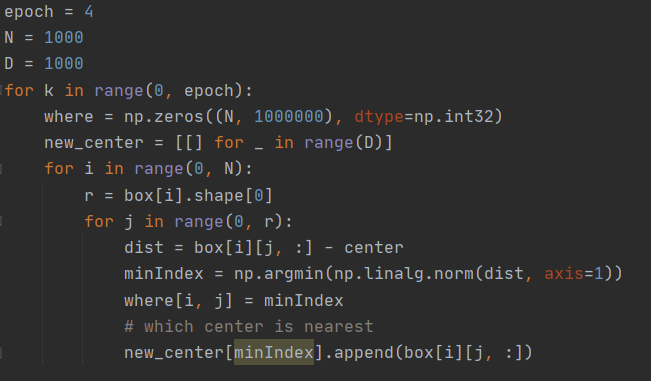
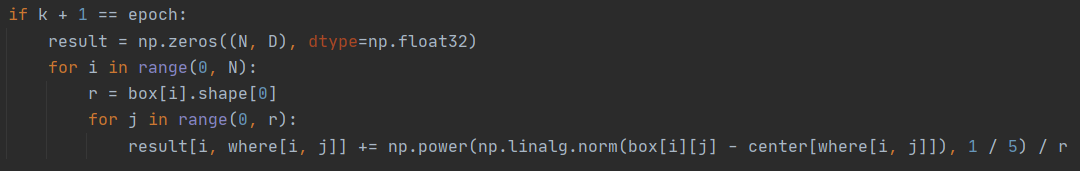
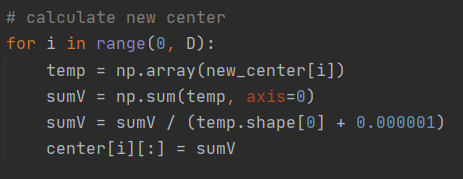
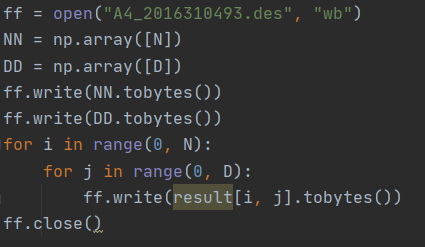
5) 각 feature 별로 가까운 center가 정해지면 center마다 자신에게 속하는 feature들의 평균값으로 center값을 갱신한다.

6) 3~5의 과정을 반복할수록 center값이 조금씩 변화하는데 이 과정을 1 epoch라고 한다.

7) k번의 epoch를 수행한다.

8) 모든 과정이 끝나면 des 파일을 만든다.

2. 코드



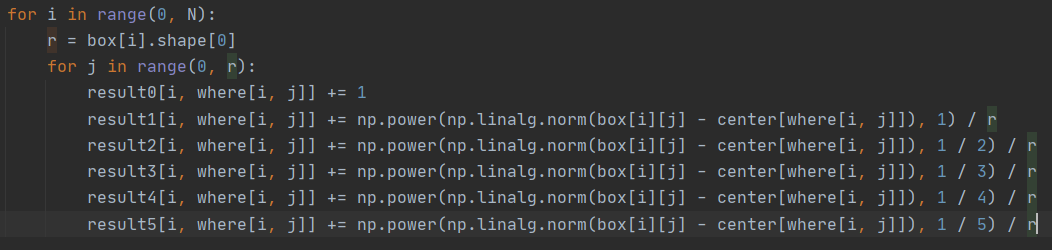
3. 시행착오

1) center의 개수 변화

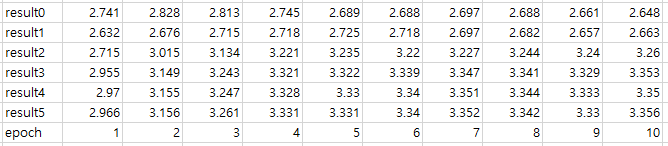
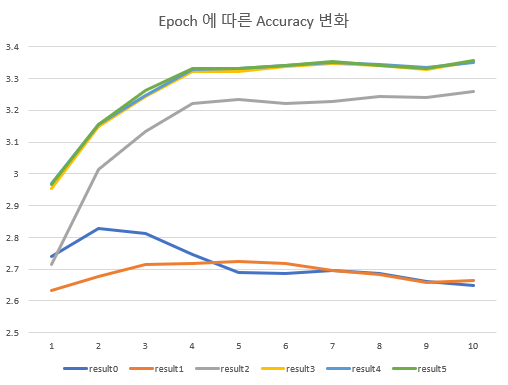
: center 개수를 500개, 750개, 1000개로 테스트 해본 결과 1000개일 때 Accuracy가 가장 높았다.

따라서, D의 값을 1000으로 하였다.

2) descriptor



위의 6가지 케이스로 나누어 테스트해 보았다. Result0의 경우가 예상대로 가장 낮은 Accuracy가 나왔다. 처음에는 result1처럼 center와의 거리만을 더해주었는데 Accuracy가 크게 높지 않았고, 거리 값에 weight를 주었더니 Accuracy가 증가하였다. 얼마만큼의 weight를 주어야 하나에 대한 테스트가 result2~5이다. 지수함수가 가장 간편하게 사용할 수 있을 거라 생각하여 제곱근, 세제곱근, 네제곱근, 다섯제곱근까지 테스트해 본 결과 세제곱근, 네제곱근, 다섯제곱근의 Accuracy가 비슷하게 나왔지만 다섯제곱근이 미세하게 가장 높게 나와 최종 출력으로 result5를 선택하였다.

 3) epoch

Epoch 4회이상부터는 크게 Accuracy가 변화하지 않는 모습을 볼 수 있다.

실행시간을 고려하여 epoch 횟수를 4회로 적용하여 최종 출력을 하도록 하였다.