인공지능

Project 1

학　　과: 컴퓨터정보공학부

담당교수: 박철수 교수님

학　　번: 2019202050

성　　명: 이강현

1. Introduction

이번 프로젝트의 전체적인 개요는 lfw dataset 을 이용하여 PCA 알고리즘의 성능을 KNN 모델을 이용하여 평가하고 추가적으로 이미지 분류 모델로 많이 사용되는 CNN 과 비교해 보는 것이다. 프로젝트를 통해 PCA 알고리즘의 원리와 효과를 깊이 학습하는 것뿐만 아니라 KNN,CNN 모델을 학습시켜보며 머신러닝, 딥러닝과정을 직접 코딩을 통해 학습해볼 수 있다.

2. Algorithm

PCA(주성분 분석): 고차원의 데이터를 저차원의 데이터로 환원시키는 기법

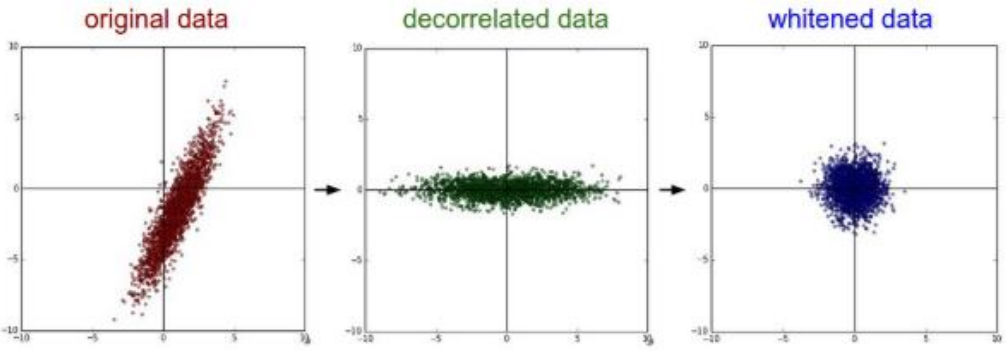
2 차원으로 변환되게 PCA 를 적용한다면 고차원의 데이터를 저차원으로 변환하기 위해 직교 변환을 사용하고 가장 분산이 큰 축을 첫번째 주성분, 두 번째로 큰 축을 두번째 주성분으로 놓이도록 새로운 좌표계를 만들고 데이터를 선형 변환한다.

PCA 는 다음과 같은 순서로 진행된다.

데이터에서 평균을 빼 데이터가 중앙에 위치하게 끔 하고 covariance 를 계산한다. Covariance matrix 에서 eigenvector 와 eigenvalue 를 계산하고 이렇게 구한 eigenvector 들이 새로운 좌표축으로서의 역할을 한다. 이 좌표축에 데이터를 projection 하면 데이터를 더 적은 feature 로 표현 가능해진다.

PCA whitening: feature 간 서로 uncorrelation 해지도록 하고 동일한 분산을 가지도록 하는 작업

데이터를 eigenvalue 에 루트를 씌운 역수로 스케일링 하여 각 주성분의 분산을 동일하게 만든다.



KNN: K-Nearest Neighbors 는 지도 학습으로 분류,회귀문제를 해결하는 머신러닝 알고리즘이다.

KNN 은 데이터의 특징과 레이블만으로 학습을 진행한다. 먼저 새로운 데이터가 주어지면 유클리드 거리가 가장 적은 k 개의 이웃을 찾아 데이터의 클래스를 예측한다.
이때 k 값은 너무 작게되면 노이즈에 민감해질 수 있고 너무 크게되면 모델을 둔하게 만들기에 적절한 파라미터를 설정하는 것이 중요하다.

CNN: Convolutional Neural Network 로 이미지 및 비디오 데이터와 관련된 인공 신경망 모델이다.

주로 세개의 계층을 가지며 이는 1. 합성곱 계층 2. 풀링 계층 3. 완전 연결 계층이다.

합성곱 계층(Convolutional Layer)는 필터를 이미지 위로 슬라이딩하여 합성곱을 진행한다. 이를 통해 이미지의 공간구조로 보존하면서 이미지의 패턴이나 특징을 보존한다.

풀링 계층(Pooling Layer)는 출력 데이터의 크기를 줄이고 계산량을 감소하기 위해 정보의 특징적인 부분을 뽑아낸다.

완전 연결 계층(Fully Connected Layer)는 CNN 계층의 마지막 부분으로 데이터를 평탄화하여 추출된 특징들을 기반으로 분류 및 예측을 수행하는 계층이다.

3. Result

<Dataset>

```
kang@2019202050:~/바탕화면/3-2$ python3 ./assignment1_PCA_KNN.py
people.images의 형태
(3023, 87, 65)
class 갯수
62
<사진 분포 확인>
Alejandro Toledo           39  Alvaro Uribe              35  Amelie Mauresmo           21  Andre Agassi              36
Angelina Jolie             20  Ariel Sharon              77  Arnold Schwarzenegger     42  Atal Bihari Vajpayee      24
Bill Clinton               29  Carlos Menem              21  Colin Powell             236  David Beckham             31
Donald Rumsfeld           121  George Robertson          22  George W Bush            530  Gerhard Schroeder         109
Gloria Macapagal Arroyo    44  Gray Davis                26  Guillermo Coria           30  Hamid Karzai              22
Hans Blix                  39  Hugo Chavez               71  Igor Ivanov               20  Jack Straw                28
Jacques Chirac             52  Jean Chretien             55  Jennifer Aniston          21  Jennifer Capriati         42
Jennifer Lopez             21  Jeremy Greenstock         24  Jiang Zemin               20  John Ashcroft             53
John Negroponte            31  Jose Maria Aznar          23  Juan Carlos Ferrero       28  Junichiro Koizumi         60
Kofi Annan                 32  Laura Bush                41  Lindsay Davenport         22  Lleyton Hewitt            41
Luiz Inacio Lula da Silva  48  Mahmoud Abbas             29  Megawati Sukarnoputri     33  Michael Bloomberg         20
Naomi Watts                22  Nestor Kirchner           37  Paul Bremer               20  Pete Sampras              22
Recep Tayyip Erdogan       30  Ricardo Lagos             27  Roh Moo-hyun              32  Rudolph Giuliani          26
Saddam Hussein             23  Serena Williams           52  Silvio Berlusconi         33  Tiger Woods               23
Tom Daschle                25  Tom Ridge                 33  Tony Blair               144  Vicente Fox               32
Vladimir Putin             49  Winona Ryder              24
20개만 뽑은 데이터의 형태
(1240, 5655)
```

전체 데이터셋은 3023 장의 87 x 65 크기의 사진들이다. 학습할 데이터셋은 전체 데이터셋에서 20 개 이상의 사진개수를 가진 클래스만 확인해보았을 때 20 개부터 530 개까지 클래스 별로 사진 수가 다양하다. 이대로 학습을 진행하면 클래스별로 학습이 편중되어 모델이 데이터가 많은 클래스를 더 과적합되게 학습할 가능성이 있다. 따라서 20 개를 가진 클래스의 데이터 개수만큼 전체 클래스에서 추출해내어 62 개의 클래스와 각각 20 개의 데이터셋으로 이루어진 (1240,5655(87 x 65))의 데이터를 얻었다. 또한 train_data 와 test_data 를 3:1 비율로 나누어서 학습 및 검증을 진행하였다.

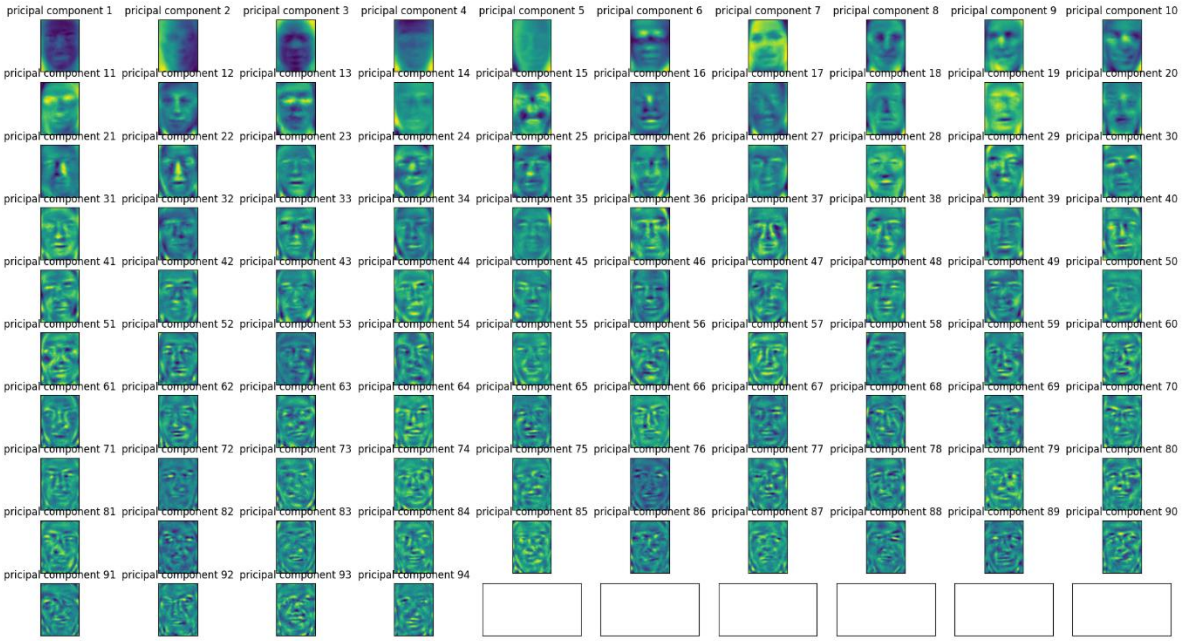
<PCA & PCA whitening>

PCA 만 적용하였을 때 변환 전과 변환 후의 성분이다.

```
pca 적용 전 shape:(930, 5655)
[[0.13333334 0.1254902  0.12287582 ... 0.53725487 0.53725487 0.53594774]
 [0.793464   0.75163394 0.70718956 ... 0.8104575  0.8300654  0.8718955 ]
 [0.19346406 0.22745098 0.19607843 ... 0.772549   0.75555557 0.74509805]
 ...
 [0.15686275 0.1882353  0.22614379 ... 0.4862745  0.5111111  0.52156866]
 [0.20915033 0.1764706  0.13202615 ... 0.6509804  0.72549015 0.7503268 ]
 [0.08235294 0.09281046 0.10980392 ... 0.7372549  0.75163394 0.76339865]]
pca 적용 후 shape:(930, 94)
[[-8.1313047e+00  1.5277666e+00 -2.6671484e+00 ... -7.3610707e-03
   8.3416289e-01  1.4408964e-01]
 [-1.6747576e+00 -6.7672956e-01  4.2199383e+00 ... -5.7442117e-01
   1.9156487e-01 -7.1636111e-01]
 [ 1.3040959e+01  2.2798035e+00  6.1366463e-01 ... -6.1058095e-03
  -1.6175050e-01  8.9065099e-01]
 ...
 [-5.8107696e+00 -1.5321264e+00 -1.4629927e+00 ... -5.9682775e-02
  -4.3929911e-01 -1.7847969e-01]
 [-7.4875407e+00 -4.2560062e+00 -2.2592430e+00 ... -8.0041748e-01
  -3.0601136e-02  4.3097934e-01]
 [ 8.0935061e-01  1.0014988e+00 -3.6709383e+00 ...  1.9842872e-01
   1.2617102e-01  3.9692372e-02]]
```

Feature 의 개수가 감소하고 각 성분들이 PCA 만 적용된 채 출력되었다.

주성분들을 사진으로 출력하면 다음과 같다.



94 개의 주성분 개수만큼 eigenface 를 눈으로 확인할 수 있었다.

이번엔 whitening 을 적용한 경우와 비교를 해본다.

```
pca 적용 전 shape:(930, 5655)
[[0.13333334 0.1254902  0.12287582 ... 0.53725487 0.53725487 0.53594774]
 [0.793464   0.75163394 0.70718956 ... 0.8104575  0.8300654  0.8718955 ]
 [0.19346406 0.22745098 0.19607843 ... 0.772549   0.75555557 0.74509805]
 ...
 [0.15686275 0.1882353  0.22614379 ... 0.4862745  0.5111111  0.52156866]
 [0.20915033 0.1764706  0.13202615 ... 0.6509804  0.72549015 0.7503268 ]
 [0.08235294 0.09281046 0.10980392 ... 0.7372549  0.75163394 0.76339865]]
pca 적용 후 shape:(930, 94)
[[-1.3099911   0.38011557 -0.82580847 ... -0.01928195  2.2012894
   0.38107875]
 [-0.26981127 -0.16837353  1.3065867  ... -1.5046667   0.50552446
  -1.8945845 ]
 [ 2.1009595   0.56722593  0.19000423 ... -0.01599385 -0.42684668
   2.3555348 ]
 ...
 [-0.9361421  -0.3812003  -0.4529751  ... -0.15633596 -1.1592754
  -0.47203127]
 [-1.2062777  -1.0589145  -0.6995119  ... -2.0966525  -0.08075396
   1.1398257 ]
 [ 0.13039017  0.24917765 -1.1366042  ...  0.51977384  0.33295527
   0.10497576]]
x_train_pca.shape
train형태:(930, 94)
x_test_pca.shape
test형태:(310, 94)
```

분산이 1 이 되게끔 스케일링이 되었다.

이를 통해 whitening 을 적용한 경우와 적용하지 않은 경우로 나누어 knn 모델을 학습시켜보고 정확도의 차이를 비교해본다. 주성분은 94 개로 고정하였다.

먼저 whitening 을 적용하지 않은 경우이다.

```
PCA적용한 knn 점수:
0.187
```

```
PCA 적용하지 않은 knn 점수: 0.187
선택한 주성분 개수: 94
누적 기여율: 0.90
```

Whitening 을 적용하지 않은 경우 PCA 를 사용하고 사용하지 않고의 점수 차이가 나지 않았다.

다음은 whitening 을 적용한 경우이다.

```
knn.score(x_test, y_test)
 PCA 적용하지 않은 knn 점수: 0.187
```

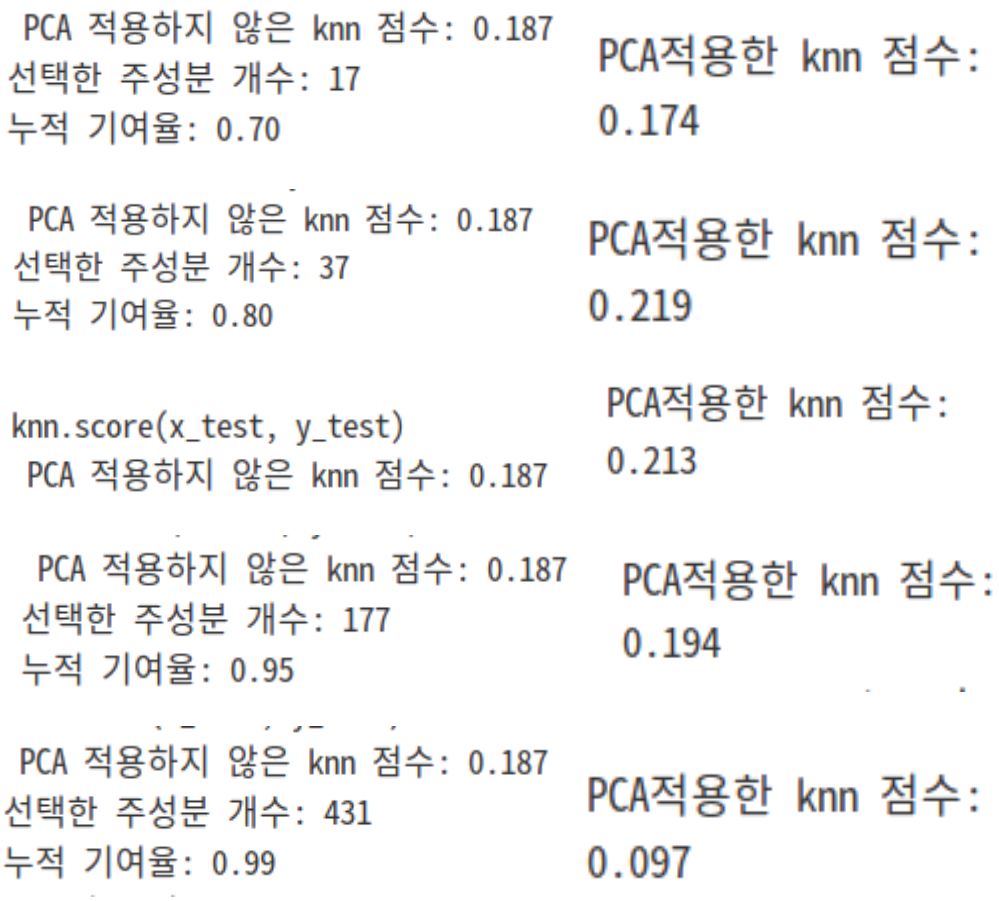
```
PCA적용한 knn 점수:
0.213
```

Whitening 을 적용한 경우 knn 의 정확도가 대략 2.6% 증가하였다.

주성분간의 상관관계가 줄어 성능을 더 개선시킨 것 같다는 예상이다.

다음은 주성분 개수에 따른 knn 의 정확도를 비교해보았다.

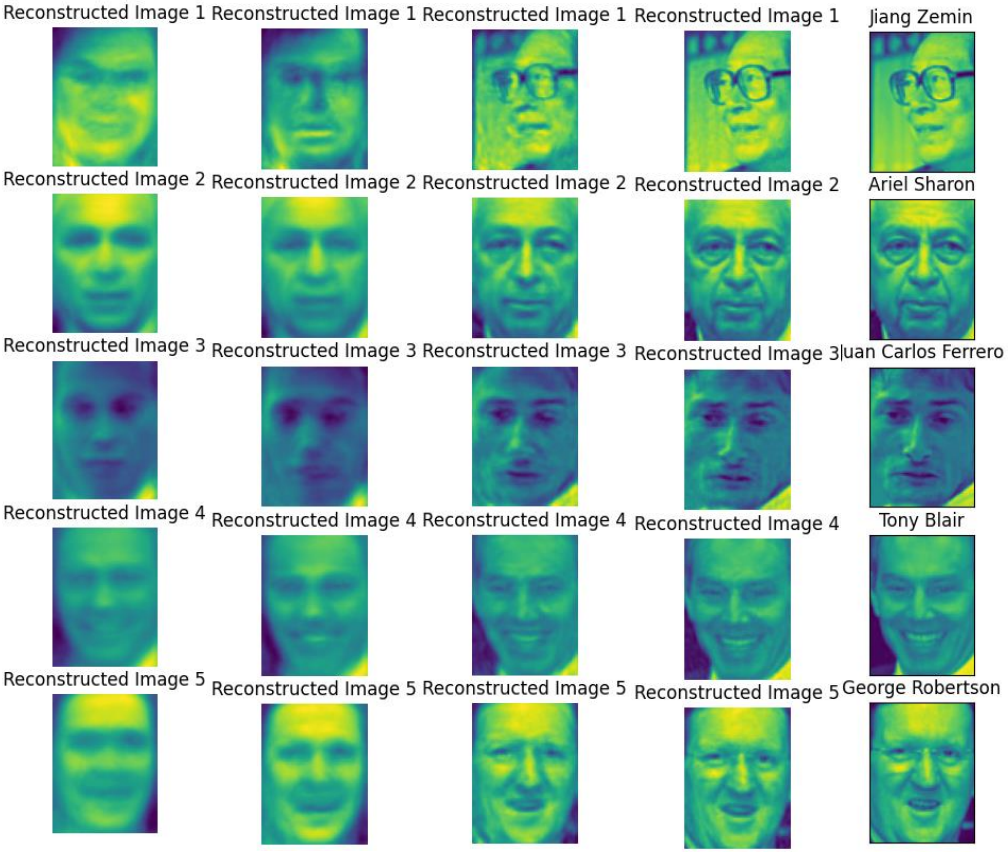
주성분 개수는 주성분의 개수에 따라 데이터셋을 얼마나 구분할 수 있는지를 척도로 하는 누적 기여율에 따라 선택하였다. 또한 whitening 옵션을 True 로 하였을 때 좋은 성능을 보였기에 True 로 진행하였다.



위에서부터 70%,80%,90%,95%,99%의 누적기여율일때의 결과이다. 주성분 개수를 데이터셋의 70% 설명할 수 있게끔 설정한 경우와 99%까지 설명할 수 있게끔 설정한 경우는 오히려 knn 의 정확도 점수가 PCA 를 적용했을 때 더 낮게 나오는 것을 확인할 수 있다. 반면에 80%,90%,95%까지 설명할 수 있게끔 한 경우는 PCA 기법을 사용하는 것이

knn 모델 성능 개선에 효과가 있었다. 너무 많거나 적은 주성분 개수는 오히려 데이터를 분류하는데 좋지 않게 작용한다는 것을 보여준다.

다음은 주성분의 개수에 따른 이미지 재구성이다.



<17 개> <37 개> <177 개> <431 개> <원본>

주성분의 개수가 늘어날수록 얼굴 이미지를 재구성하는 데는 도움이 되었다. 이는 87 X 65 = 5655 개의 feature 개수를 축소하는 것이 PCA 인데 최대한 많은 주성분 개수를 가지는 것이 원본 데이터의 정보를 잃지 않고 더 많이 가지고 있기에 복원에 더 효과적이었을 것이라고 생각한다.

다음은 Data Preprocessing 을 진행했을 때와 하지 않았을 때의 KNN 성능차이를 비교해본다.

전처리과정은 MinMaxScalar 를 사용하였고 누적 기여율 90%인 주성분개수 94 개일때와 비교하였다.

```
PCA 적용하지 않은 knn 점수: 0.187
PCA적용한 knn 점수:
0.219
```

<전처리 O>

```
knn.score(x_test, y_test)
 PCA 적용하지 않은 knn 점수: 0.187
PCA적용한 knn 점수:
0.213
```

<전처리 X>

MinMaxScaler 를 사용하면 각 특성이 동일한 스케일을 가지게 되어 PCA 의 결과가 모든 특성에 균일하게 적용된다. 따라서 전처리를 진행했을 때 성능이 소폭 상승한 것을 확인하였다.

마지막으로 CNN 모델과의 비교이다. CNN 모델 또한 같은 방법으로 데이터셋을 만들고 학습시켰다. 모델은 CNN 의 대표적인 모델 중 하나인 RasNet50 을 사용하였다. 결과는 다음과 같았다.

```
Accuracy: 45.56%
```

주성분 개수를 조절하고 전처리하여 가장 높게 KNN 이용하여 분류작업을 진행하였을 때 21.9%의 정확도를 얻었으나 CNN 은 그보다 두배 이상 좋은 성능을 보여주었다.

CNN 은 데이터에서 특징을 자동적으로 학습하고 분류의 용이한 특징을 잘 뽑아내는 작업을 각 계층에서 수행할 수 있고 이미지의 공간적인 특징 또한 KNN 과는 다르게 보존할 수 있다는 장점이 있다. 따라서 고차원적인 이미지를 분류하는데 KNN 보다 더 우수한 성능을 낼 수 있었을 것이라고 생각된다. 반면에 KNN 은 특징 공간의 데이터 포인트 거리에 따라 클래스를 결정하기 때문에 픽셀이 하나씩 이동할 때 급격한 변화를 보이는 이미지라는 데이터 특성이 knn 이 이미지를 분류할 때 어려움을 주었을 것이라고 생각된다.

KNN 은 훨씬 간단하고 훈련이 필요하지 않다는 장점이 있고 이는 계산량이 매우 적다는 장점을 가지기에 단순한 데이터셋을 분류하는 데에는 적합할 것이다.
CNN 은 모델을 훈련해야 하고 이미지를 분류하기 위해 많은 계산을 해야 한다는 점이 단점이다.

따라서 크기가 작은 사진이나 단순한 데이터셋에서는 PCA 와 whitening, 전처리등의 기법을 활용하여 픽셀간 변동성을 어느정도 줄이면서 KNN 알고리즘을 적용한다면 계산 상의 이점을 챙길 수 있고 복잡하고 고차원의 데이터를 분류해야 한다면 CNN 을 활용하는 것이 더 좋다고 생각한다.

4. Consideration

결과적으로는 CNN 이 좋은 결과를 보였지만 PCA 의 약간의 정보 손실은 존재하나 고차원의 데이터를 저차원의 데이터로 변환할 수 있다는 점은 매우 유용한 기법이라고 생각한다. 이는 적은 차원으로 충분히 분류가 가능하나 불필요하게 높은 차원을 가지는 데이터들 혹은 다수의 클래스를 가진 데이터를 1 차적으로 이분법적으로 분류해야 하는 경우등 특수한 상황에서 PCA 를 적용해볼 수 있을 것 같다. PCA whitening 은 픽셀간에 급격한 차이를 스케일링을 통해 조절하여 변동성에 민감한 KNN 의 성능을 끌어올려주는 것이 매우 좋은 옵션이라고 생각했다. 하지만 이와 같은 기법이 오히려 특정 상황에서는 정보손실을 야기할 수 있으니 상황에 맞게 잘 사용하는 것이 중요하다고 생각했다. CNN 과 KNN 을 설계하고 학습시켜보면서 머신러닝 딥러닝을 실습해볼 수 있어 좋은 경험이었던 것 같다.

5. Reference

https://taeu.github.io/cs231n/deeplearning-cs231n-Neural-Networks-2/ PCA whitening