인공지능

Project1

담당교수 : 박철수 교수님

학 번 : 2021202058

성 명 : 송채영

1. Introduction

이번 프로젝트에서는 Principal Component Analysis (PCA)을 활용하여 얼굴 인식 과정을 시뮬레이션 해보는 것이 목표이다. 얼굴 데이터베이스는 Labeled Faces in the World(LWF)를 활용하며 해당 Dataset 을 사용하여 얼굴의 주성분을 시각화 하여 확인해보고, PCA 를 적용한 훈련 데이터로 학습시켜 K-Nearest Neighbor (KNN) 알고리즘을 통해 정확도를 확인해본다. 더 나아가 같은 Dataset 을 활용하여 Convolutional Neural Network (CNN) 모델을 통해 학습시켜본 후 KNN 의 결과와 CNN 의 결과를 비교해본다.

2. Algorithm

우선 프로젝트에서 사용한 알고리즘에 대해 설명하겠다.

- PCA

주성분 분석 알고리즘인, PCA(Principal Component Analysis)는 다차원 데이터를 저차원으로 변환하는 것을 말하며, 데이터의 차원을 줄이는 데 사용된다. PCA 는 데이터의 주요 정보를 보존하면서 데이터를 새로운 축으로(주성분) 변환해 데이터의 분산을 최대화한다.

PCA 의 작동 방식은 다음과 같다. PCA 는 데이터를 평균이 0 이고 분산이 1 인 형태로 정규화하거나 표준화하여 스케일을 일치시킨 후, covariance matrix 를 계산하여 데이터의 상관성과 분산 정보를 파악한다.

$$S = \frac{1}{n-1}(X - \widehat{X})^T(X - \widehat{X})$$

이후 covariance matrix 를 고유값(eigenvalues)과 고유벡터(eigenvectors)로 분해하며, eigenvalues 이 큰 순서대로 eigenvectors 를 선택하여 주요 정보를 보존한다. 이때 eigenvalues 은 데이터의 분산을 가지며 eigenvectors 은 데이터의 주성분 방향을 나타낸다. 마지막으로, 선택된 eigenvectors 를 활용해 데이터의 차원을 축소하고 새로운 공간을 구성한다.

$$S_v = \lambda_v \ (v = eigenvector, \lambda = eigenvalues)$$

즉, PCA 는 주성분 분석의 결과로 주요 eigenvectors(주성분)와 해당 eigenvalues 을 얻어 데이터를 eigenvectors 의 선에 projection 하여 새로운 특성 공간으로 나타내는 과정을 말한다.

- Whitening

Whitening 은 다차원 데이터의 covariance 구조를 이용하여 데이터를 변환하는 과정을 말한다. 이 변환을 통해 데이터의 각 특성은 평균이 0 이고 분산이 1 인 독립적인 확률 변수가 되어 데이터의 상관 관계가 없어져 독립적으로 다룰 수 있다.

Whitening 의 동작방식은 다음과 같다. 데이터를 평균 중심화해, 데이터의 중심을 원점으로 이동시킨다. 이후 다차원 데이터의 covariance matrix 를 계산한다. 이때 covariance matrix 는 데이터 특성 간의 상관 관계를 나타내며 데이터의 분포 정보를 제공한다. 이후 covariance matrix 를 eigenvalues 과 eigenvectors 로 분해한 후 데이터를 whitening 변환을 한다.

$X_{white} = P \cdot diag(\lambda^{-0.5}) \cdot P^T \cdot X$

$(\mathrm{X} = \mathrm{data}, \mathrm{P} = \mathrm{eigenvector\ matrix}, \lambda = \mathrm{eigenvalue\ diagonal\ matrix})$

즉, Whitening 은 다차원 데이터의 특성 간 상관 관계를 제거하고 독립성을 강조하여 데이터 분석과 머신 러닝에 유용한 기술이다.

- KNN

K-최근접 이웃 알고리즘인, KNN(K-Nearest Neighbors)는 데이터의 Classification 및 Regression 문제를 해결하는데 사용되는 기계학습 알고리즘이며, 주어진 데이터 포인트 주변의 가장 가까운 이웃들을 사용하여 분류 또는 예측을 한다.

KNN 의 작동 방식은 다음과 같다. 새로운 데이터 포인트가 주어지면, 해당 데이터 포인트와 학습 데이터셋 간의 거리를 측정하며, 가장 가까운 K 개의 이웃을 선택한다. 이후 이웃들의 다수결 또는 가중 평균을 사용해 새로운 데이터 포인트를 분류하거나 예측한다.

즉 KNN 은 간단하면서 강력한 지도 학습 알고리즘으로, 데이터 분류와 예측 문제를 해결하는데 사용된다.

- CNN

합성곱 신경망인, CNN(Convolutional Neural Network)은 주로 이미지 처리 및 패턴 인식에 사용되는 딥러닝 알고리즘이며 입력 데이터에서 패턴을 자동으로 감지하고 인식하기 위해 설계된 신경망 구조를 가지고 있다.

CNN 의 주요 구성 요소와 동작 방식은 다음과 같다. 우선 이미지 또는 다차원 데이터를 입력으로 받은 후 합성곱층(Convolutional Layer)을 이용해 입력 데이터를 필터와

합성곱하여 이미지의 특징을 추출한다. 이어서 풀링층(Pooling Layer)를 이용해 특징 맵의 크기를 줄이고 중요한 정보를 보존한 후 완전 연결층(Fully Connected Layer)를 사용해 특징 맵에서 최종 출력을 계산하고 예측한다. 마지막으로 CNN 은 손실 함수를 최소화하기 위해 backpropagation 알고리즘을 사용해 가중치를 조정한다.

즉, CNN 은 이미지 처리와 패턴 인식을 위한 딥러닝 알고리즘으로, 다차원 데이터의 특징 추출과 분류에 활용된다.

다음으로는 프로젝트에서 이용한 method 에 대해 설명하겠다.

- 이미지 전처리

이미지 데이터는 0 에서 255 사이의 값으로 표현되므로, 0 에서 1 사이의 범위로 스케일링하여 모델에 입력하였다.

- Train-Test Split

데이터를 훈련 및 테스트 세트로 분할하기 위해 train_test_split 함수를 사용하였다.

3. Result

- LWF(Labeled Faces in the World) Dataset

LWF(Labeled Faces in the world) dataset 은 컴퓨터 비전 및 얼굴 인식 연구를 위한 공개 데이터 셋 중 하나이며 얼굴 이미지와 해당 얼굴들의 레이블(이름) 정보를 포함하고 있다. 위의 사진은 LFW 데이터셋에 있는 이미지의 샘플을 출력한 사진이다.

```
people.images.shape : (3023, 87, 65)
class : 62
```

LWF dataset 에는 62 명의 얼굴을 찍은 이미지가 총 3023 개 있으며 각 이미지의 크기는 87*65 픽셀이다.

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

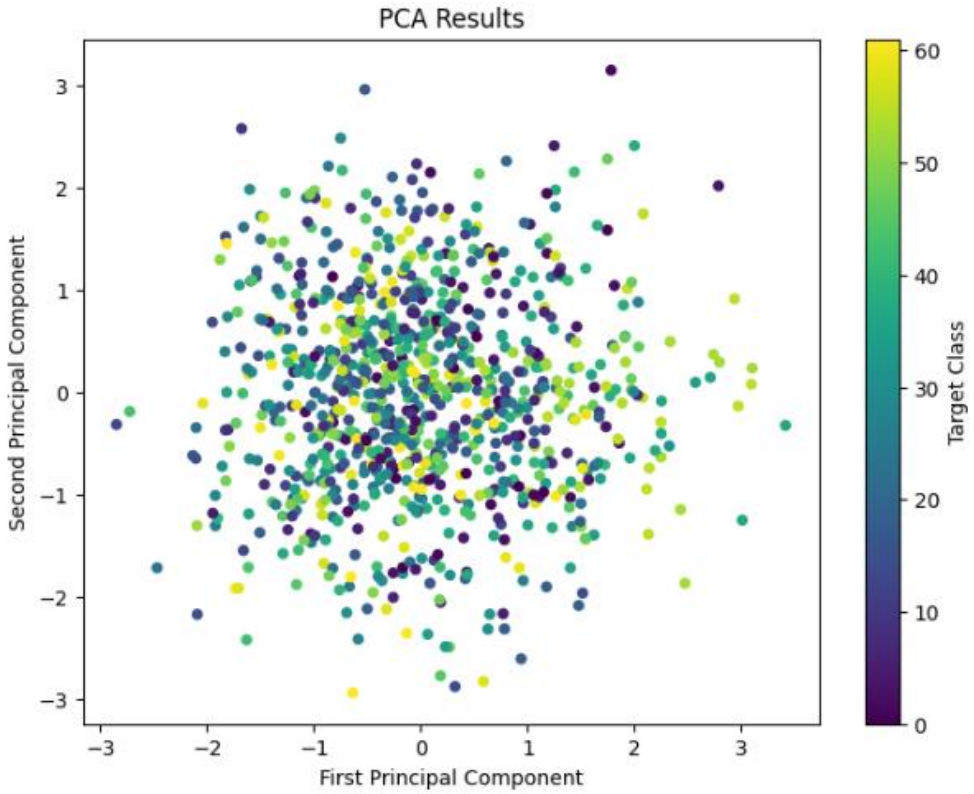
각각의 인물에 대한 이미지의 개수를 출력해보았다. 위 dataset 에는 George Bush 의 이미지가 530 개 있는 것을 통해 편중된 데이터가 있음을 확인하였다.

데이터가 많은 클래스의 경우 학습을 overfit 하는 경우가 발생할 수 있어, 편중된 데이터를 막기 위해 가장 작은 샘플 개수인 20 개에 맞추었다.

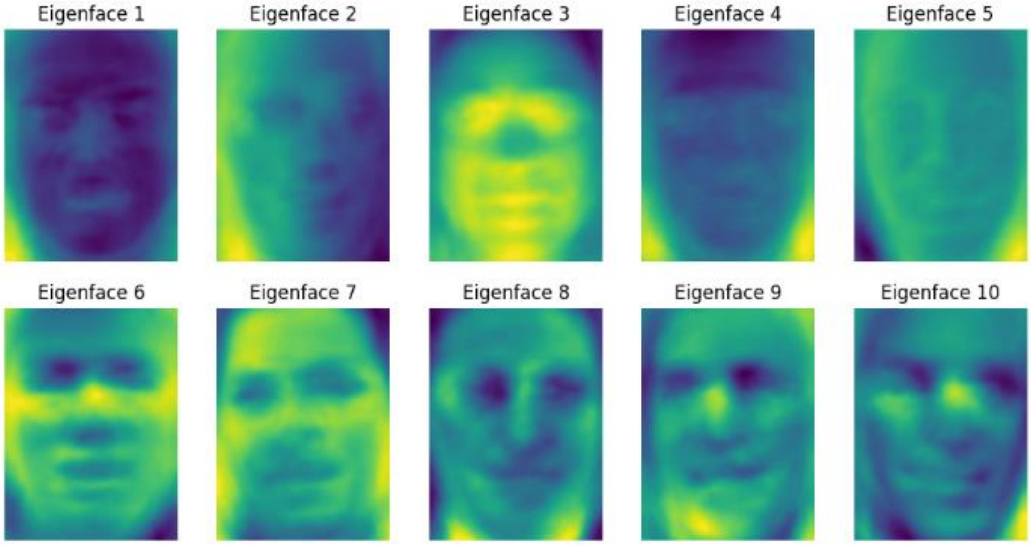
따라서 62 개의 클래스와 각각의 클래스는 20 개의 데이터가 있으며 이 dataset 의 이미지 크기는 87*65 픽셀로 이루어진 dataset 을 얻었다.

- PCA Whitening & PCA

PCA Whitening 을 적용하여 scatter plot 을 그려보았다.



픽셀 수준에서 얼굴을 비교할 때, 얼굴의 위치가 단 한 픽셀만 이동하더라도 큰 차이가 발생하여 다른 얼굴로 인식될 수 있다. 이는 얼굴 인식에서 위치와 크기 조정이 중요한 요소임을 나타내며, 이를 해결하기 위해 주성분 분석(PCA)과 화이트닝(whitening) 알고리즘을 사용하였다. PCA 와 화이트닝을 조합하면 주성분의 스케일을 동일하게 조정할 수 있으며 데이터의 스케일 변동을 보다 일관되게 처리할 수 있다. 결과적으로, 데이터가 회전하는 것 뿐만 아니라 스케일이 조정되어 그래프가 원 모양으로 바뀌는 것을 확인할 수 있다.



처음 10 개의 고유 얼굴 성분을 이미지로 출력해보았다.

- KNN

KNN 모델을 학습시키는 과정에서 PCA 를 사용하지 않았을 때의 테스트 정확도이다. 아래의 테스트의 기준이 될 정확도이다.

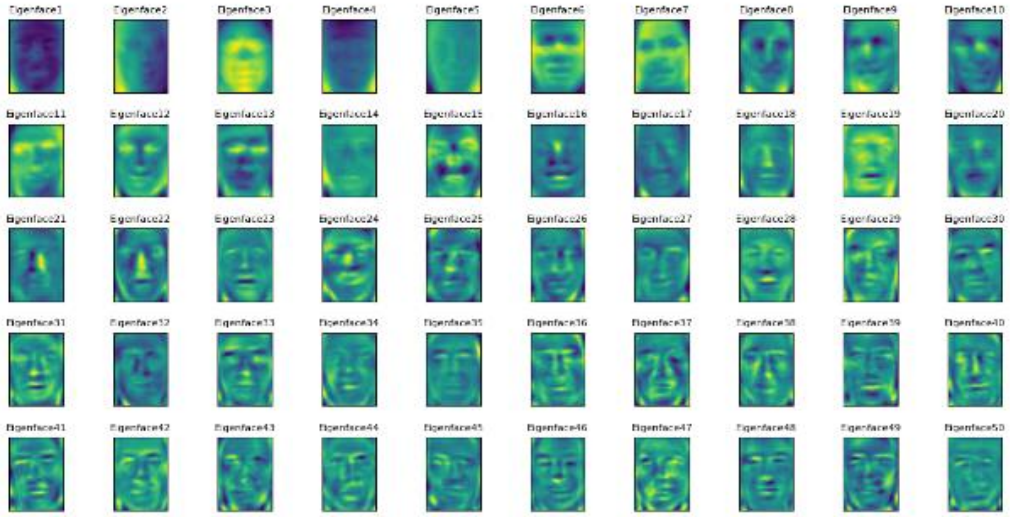
```
테스트 정확도 : 0.197
```

KNN 모델을 학습시키는 과정에서 주성분의 개수를 50 으로 설정한 후 각각 화이트닝을 적용했을 때와 하지 않았을 때의 정확도와 고유 얼굴 성분을 이미지로 출력하였다.

```
테스트 정확도: 0.229
```

```
테스트 정확도: 0.194
```

pca.components_.shape (50, 5655)



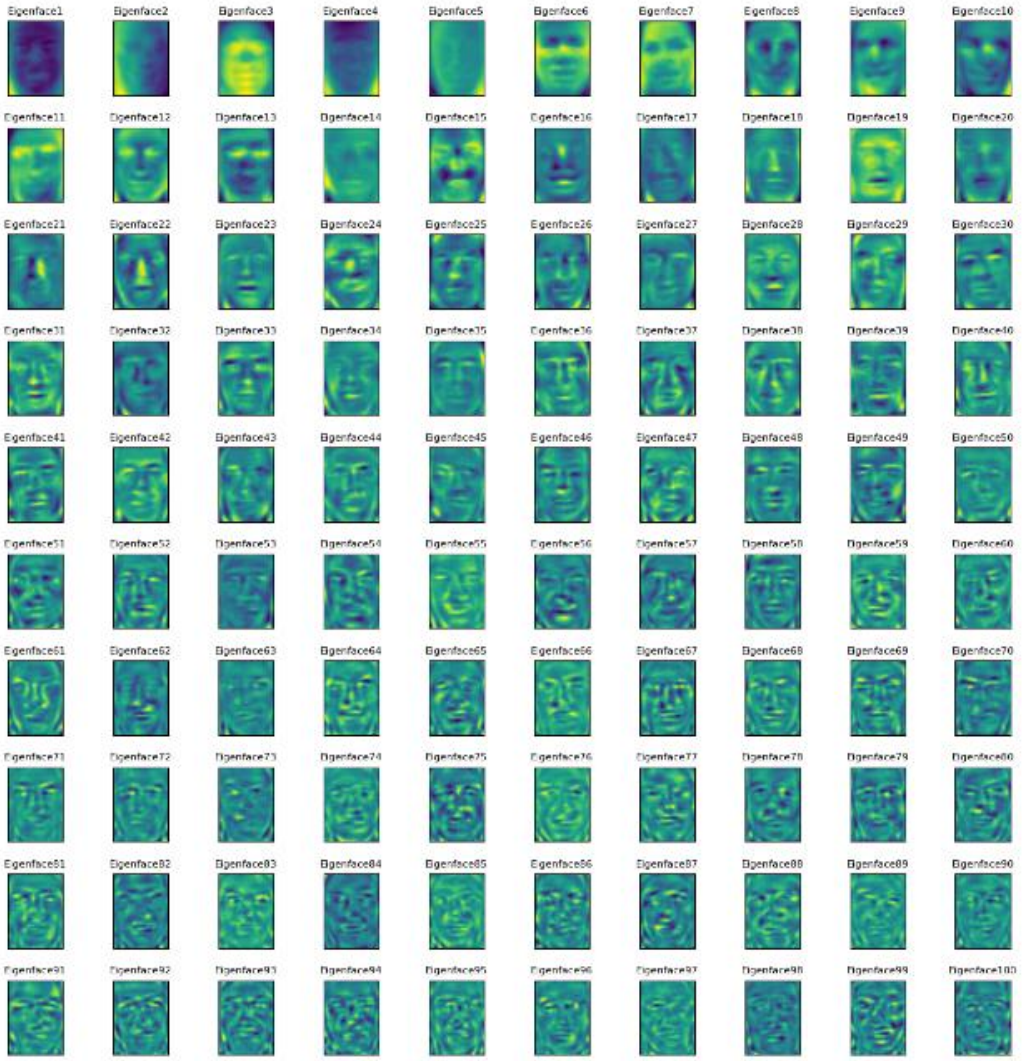
위의 결과에서 볼 수 있듯이 주성분을 50 으로 하였을 경우 화이트닝을 적용했을 때 정확도는 더 높게 나오지만, 사용하지 않았을 때는 더 적게 나오는 것을 확인할 수 있다.

KNN 모델을 학습시키는 과정에서 주성분의 개수를 100 으로 설정한 후 각각 화이트닝을 적용했을 때와 하지 않았을 때의 정확도와 고유 얼굴 성분을 이미지로 출력하였다.

테스트 정확도: 0.242

테스트 정확도: 0.197

pca.components_.shape (100, 5655)



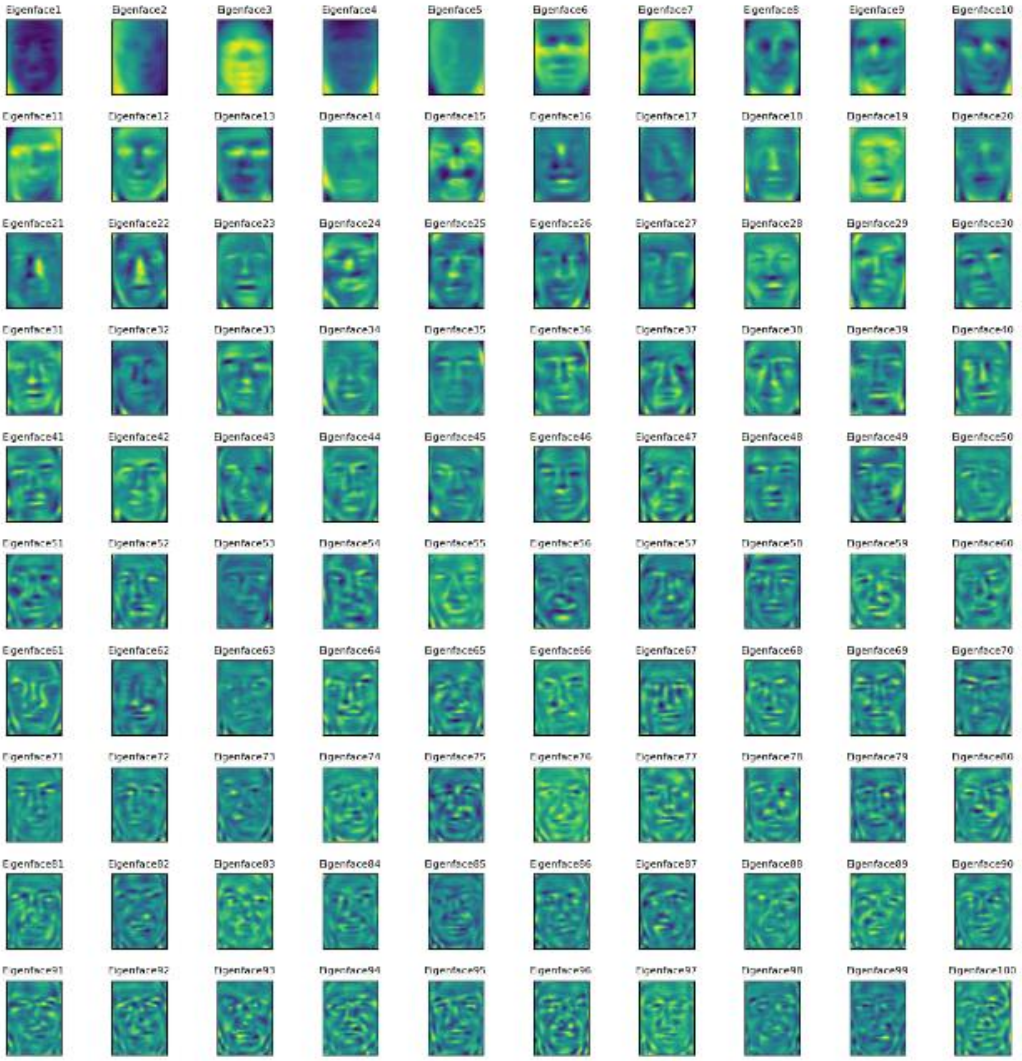
위의 결과에서 볼 수 있듯이 주성분을 100 으로 하였을 경우 화이트닝을 적용했을 때 정확도는 더 높게 나오지만, 사용하지 않았을 때는 같게 나오는 것을 확인할 수 있다.

KNN 모델을 학습시키는 과정에서 주성분의 개수를 200 으로 설정한 후 각각 화이트닝을 적용했을 때와 하지 않았을 때의 정확도와 고유 얼굴 성분을 이미지로 출력하였다.

테스트 정확도: 0.206

테스트 정확도: 0.197

pca.components_.shape (200, 5655)

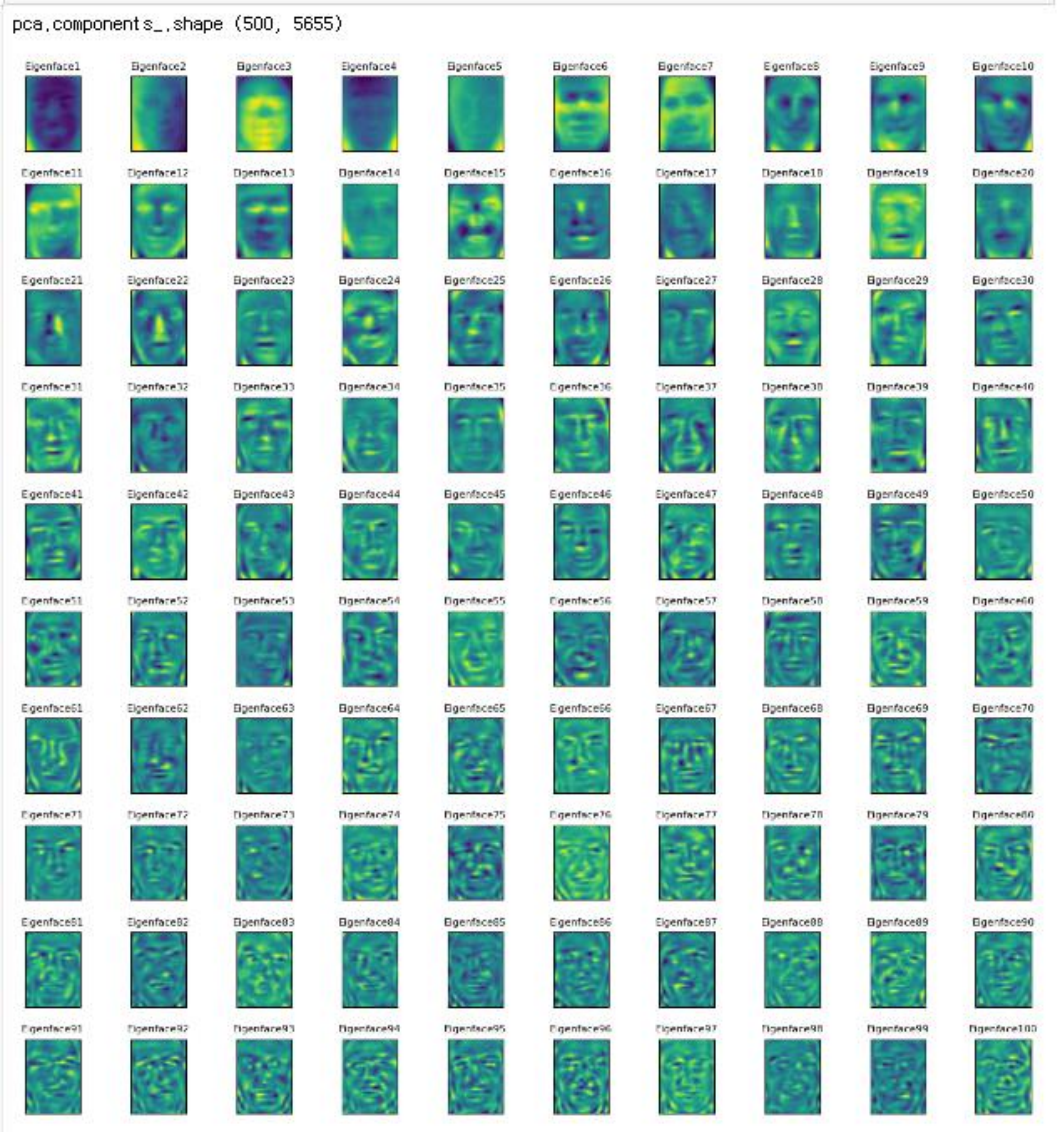


위의 결과에서 볼 수 있듯이 주성분을 200 으로 하였을 경우 화이트닝을 적용했을 때 정확도는 더 높게 나오지만 주성분을 100 으로 하였을 때 보다는 작게 나왔으며, 사용하지 않았을 때는 같게 나오는 것을 확인할 수 있다.

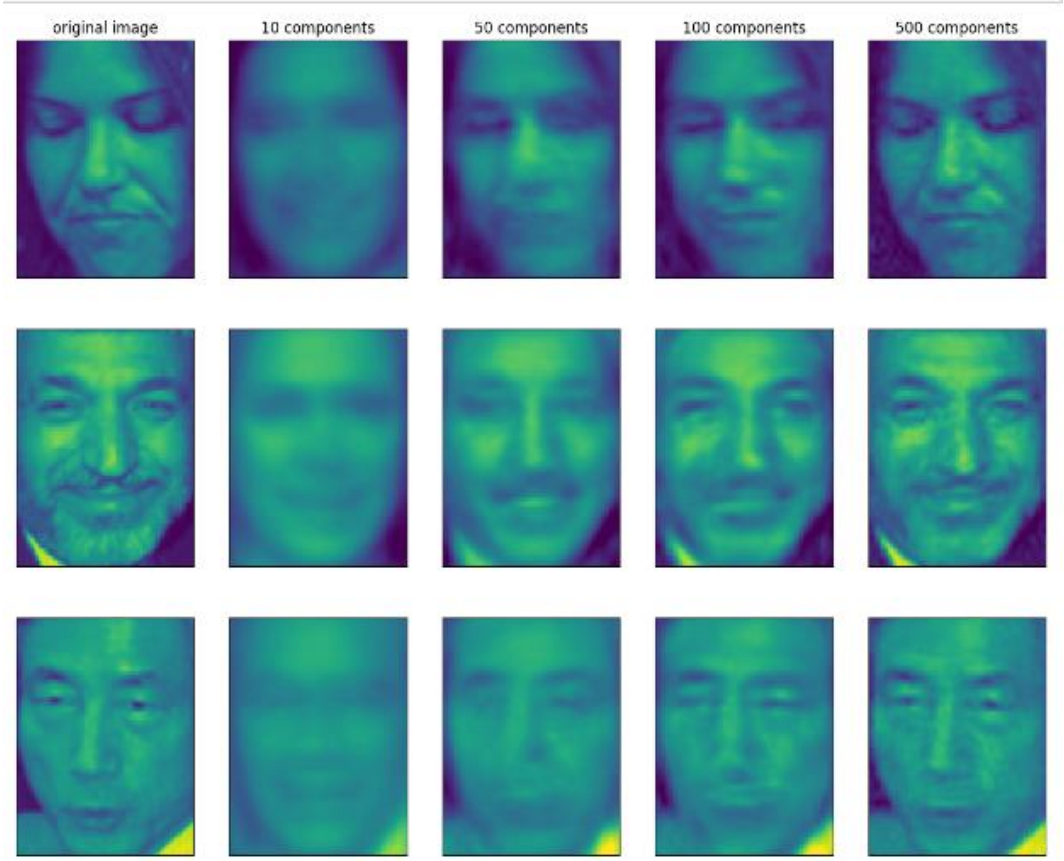
KNN 모델을 학습시키는 과정에서 주성분의 개수를 500 으로 설정한 후 각각 화이트닝을 적용했을 때와 하지 않았을 때의 정확도와 고유 얼굴 성분을 이미지로 출력하였다.

테스트 정확도: 0.094

테스트 정확도: 0.200



위의 결과에서 볼 수 있듯이 주성분을 200 으로 하였을 경우 화이트닝을 적용했을 때 오히려 정확도가 낮게 나오며, 사용하지 않았을 때는 가장 높게 나오는 것을 확인할 수 있다.



마지막으로 mglearn.plots.plot_pca_faces 함수를 사용하여 주성분 개수를 비교하여 시각화 하였다. 위 함수는 주성분이 50, 100, 200, 500 일때를 고정으로 하여 제공하기 때문에 위의 결과에서도 해당 값에 맞추어 과제를 진행하였다.

결론적으로 화이트닝을 적용한 경우 주성분과의 상관관계가 줄어 성능은 더 개선시키는 것 같다. 마지막 사진에서 볼 수 있듯이 주성분의 개수가 늘어날수록 원본 데이터의 특징을 더 많이 가지고 있어 이미지를 재구성하는데 도움이 된다. 하지만 knn 모델에 적용하였을 때의 결과는 너무 적거나 많은 경우에는 좋은 결과가 나타나지 않는 것 같다.

이어서 data preprocessing 에 따른 결과를 살펴보겠다.

데이터 샘플을 50 개의 데이터 샘플만 선택할 경우를 살펴보겠다.

주성분은 100 일때이며 각각 화이트닝을 적용했을 때와 하지 않았을 때의 결과이다.

테스트 정확도: 0.306

테스트 정확도: 0.236

테스트의 정확도가 화이트닝을 했을 때와 하지 않았을 때 모두 올라간 것을 확인할 수 있다. 더 많은 데이터를 사용하여 모델이 학습할 기회가 많아지기 때문에 정확도는 더 올라갈 수 밖에 없어 위와 같은 결과가 나왔다고 생각한다.

다음으로 데이터 샘플은 기존과 동일하게 하고 데이터 스케일링을 하지 않았을 경우의 결과이다.

테스트 정확도: 0.242

테스트 정확도: 0.197

우선 데이터 스케일을 조정하기 위해 각 픽셀의 값을 0 에서 1 사이의 값으로 정규화하는 Min-Max 스케일링을 활용하였다. 동일한 스케일을 가지게 되어 결과가 더 좋아질 것이라 생각했지만 같은 결과가 나왔다. 이미 이미지 데이터의 픽셀 값이 동일한 범위에 있기 때문에 큰 차이가 나지 않은 것이라는 생각이 들었다.

- CNN

마지막으로 CNN 모델의 결과이다.

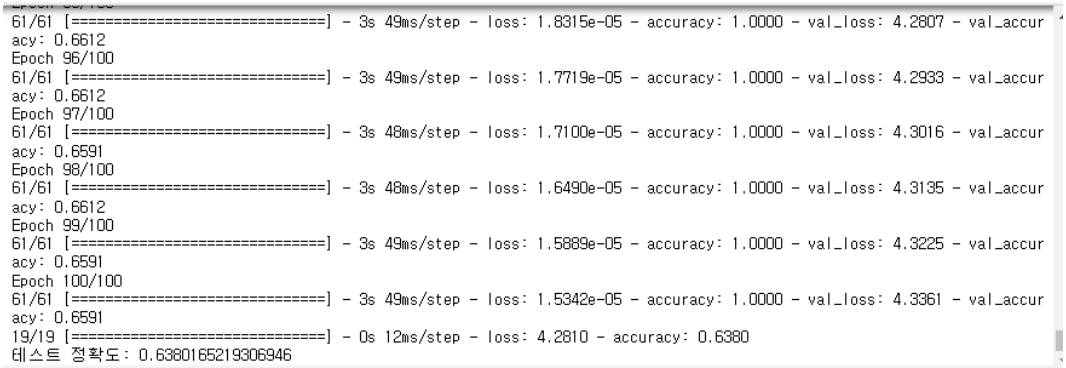
우선 기존 코드로 했을 때의 결과는 다음과 같다.

```
19/19 [==============================] - 2s 131ms/step - loss: 3.6942 - accuracy: 0.1791
Epoch 93/100
19/19 [==============================] - 2s 130ms/step - loss: 3.6880 - accuracy: 0.1791
Epoch 94/100
19/19 [==============================] - 2s 130ms/step - loss: 3.6929 - accuracy: 0.1791
Epoch 95/100
19/19 [==============================] - 2s 130ms/step - loss: 3.6852 - accuracy: 0.1791
Epoch 96/100
19/19 [==============================] - 2s 130ms/step - loss: 3.6901 - accuracy: 0.1791
Epoch 97/100
19/19 [==============================] - 2s 130ms/step - loss: 3.6915 - accuracy: 0.1791
Epoch 98/100
19/19 [==============================] - 2s 130ms/step - loss: 3.6995 - accuracy: 0.1791
Epoch 99/100
19/19 [==============================] - 2s 129ms/step - loss: 3.6858 - accuracy: 0.1791
Epoch 100/100
19/19 [==============================] - 2s 131ms/step - loss: 3.6969 - accuracy: 0.1791
Test loss :  3.714705228805542
Test accuracy :  0.16033057868480682
```

정확도가 16 퍼가 뜨는 것을 확인할 수 있다. CNN 모델의 결과가 가장 좋아야 한다고 생각했기 때문에 모델을 다시 쌓아보았다.

구현한 CNN 구축에 대해 설명하겠다. 우선 Sequential 모델을 생성하고, Convolutional layer 를 추가하였다. Convolutional layer 는 이미지에서 특징을 학습하며 32 개의 필터와 ReLU 활성화 함수를 사용하였다. MaxPooling layer 를 추가하여 맵의 크기를 줄였다.

Convolutional layer 와 MaxPooling layer 를 반복해서 쌓은 후 Fully Connected layer 를 추가하여 이미지를 클래스로 분류해주었다.



학습결과 정확도가 63 퍼까지 오른 것을 확인할 수 있다.

결론적으로 CNN 은 원본 이미지 데이터를 사용하므로 고차원 이미지 데이터의 원시 픽셀 값을 사용하며 이미지의 특징을 자동으로 학습한다. 또한 Convolutional layer 와 MaxPooling layer 를 사용해 이미지에서 특징을 추출하고 크기를 줄이고 Fully Connected layer 를 통해 추출된 특징을 사용해 이미지를 분류하고 클래스를 예측한다. KNN 은 PCA 를 사용하여 데이터의 주성분을 추출하고 데이터를 낮은 차원으로 투영하며 whitening 을 적용해 주성분의 스케일을 균일하게 조정한 후 변환된 데이터를 기반으로 KNN 분류 모델을 학습한다.

일반적으로 CNN 은 이미지 분류 작업에서 더 좋은 성능을 보인다. 원본 이미지와 데이터를 사용하기 때문에 이미지의 특징을 효과적으로 학습할 수 있다. 반면에 PCA 와 Whitening 을 사용한 KNN 은 주로 데이터의 차원 축소와 스케일 조정에 사용되므로 어떤 데이터를 작업하는지에 따라 다른 결과가 나오는 것 같다. 따라서 훈련이 따로 필요하지 않으며 계산량이 적은 KNN 을 활용할 경우에는 간단한 dataset 을, 이미지와 같은 dataset 을 사용할 경우 CNN 을 활용하여 각각의 장점에 맞는 상황에 쓰는 것이 좋다고 생각한다.

4. Consideration

이번 과제를 진행하면서 PCA, KNN, whitening알고리즘에 대해 처음 접하게 되어 기본적인 개념을 익히는데 우선시했던 것 같다. 또한 이번 과제의 목적이 성능 개선이 아닌, 이러한 알고리즘을 직접 실습해보는 것이라고 느꼈기 때문에 여러 개를 실험해보며 과제를 진행했던 것 같다. 과제를 진행하면서 느꼈던 점이나 아쉬웠던 점에 대해 서술해보겠다. 우선 데이터를 20개로 선택하였지만 더 많은 데이터를 선택했다면 KNN의 정확도를 올릴 수 있었을 것 같다. 20개의 데이터를 test와 train으로 또 나누었기 때문에 실질적으로

학습할 수 있는 데이터의 양이 너무 적었다고 생각한다. 또한 기존에 제공해주신 CNN의 코드의 정확도가 16퍼센트밖에 나오지 않은 점 역시 궁금하다. CNN이 더 잘 나와야 할 것 같았는데 KNN보다 정확도가 떨어지다 보니 모델을 다시 쌓아보며 정확도를 올리려고 했던 것 같다. CNN은 방학동안 한 번 스터디를 해보며 경험해본 적이 있어 보다 쉽게 진행하였지만, 파이썬, KNN, PCA등은 처음 접하다 보니 자료를 보고 코딩을 하면서도 정확히 어떤 원리로 진행되는지 이해가 되지 않았던 부분도 많았던 것 같다. 다음에는 KNN으로 실습했을 때 정확도가 높게 나오는 자료를 가지고 실습을 한 번 더 해보고 싶다. 이론으로만 배웠던 내용을 코드로 직접 구현해보니 이해 면에서도 도움이 됐으며 선형대수학의 중요성에 대해 다시 한 번 느끼게 된 것 같다.