인공지능 과제 1

수업 명:	인공지능
담당 교수님:	박철수 교수
학번:	2018202074
이름:	김상우
강의시간:	금 3,4

Introduction

해당 과제의 최종 목표는 주어진 데이터에 대해 PCA를 기반으로 전처리를 진행하고 이렇게 처리된 데이터를 기반으로 이미지 학습 및 재구축을 하는 것이다. 이미지 다중분류 데이터인 sklearn을 바탕으로 진행이 되며, 추가적으로 이를 기반으로 KNN과 CNN을 사용해 보며 학습에 있어 다양한 형태를 직접 실습, 모델을 만들고 fit시켜 정확도를 출력해 보는 것 또한 과제로서 두고 있다. 이에 대해 각 과정에 대한 분석과 비교를 진행하며 이에 대한 이해도를 쌓고 정리하는 것을 프로젝트의 보고서에 담아내야 한다.

Algorithm

PCA

PCA(Principal Component Analysis), 즉 주성분 분석은 특정 데이터들에 있어 차원을 줄이는 것을 목표로 하는 기술이다. 원본 데이터에 대해 가장 넓게 퍼져있는, 즉 분산이 최대가 되는 고유벡터(eigen vector)와 이에 직교하는 벡터를 구해 해당 벡터들을 기준으로 기존 데이터를 재정립하는 기술이다.(2차원) 이 과정에서 고유 벡터가 높은 데이터에 비해 높은 분산을 가진다는 점을 이용, 다른 벡터를 버리는 과정을 통해 차원을 축소할 수 있는 것이다. 이런식으로 더 많은 수의 벡터를 구하는 더 높은 차원에 대해서도 위와 같이 적용하는 것이 가능하다.(하나의 벡터를 줄이므로서 입체=>평면으로 변경하는 경우도 있다.)

벡터를 구하는 과정은 데이터들의 평균을 구하고 이를 바탕으로 데이터들의 평균이 0되는 데이터 셋을 구하고 이를 바탕으로 eigen vector와 eigen value를 구한다. 이렇게 구해진 행렬과 역행렬에 대해 행렬곱을 실행(이때 역행렬이 앞으로 오게 된다.), 이를 바탕으로 직교하는 벡터들을 구할 수 있다.

Whitening

Whitening 또는 sphering이라고도 불리며, 들어오는 input들에 대해 uncorrelated(혹은 상관관계가 적게)되게 만들고 각각의 variance를 1(동일한 분산)로 만들어주는 작업을 의미한다. 이 과정에서 convariance matrix에 대한 계산과 inverse, 즉 역행렬을 구하기 위한 계산이 필요하다.

KNN

KNN이란, K-최근접 이웃법으로 분류 문제를 해결하기 위해 사용되는 알고리즘이다. 특정 공간 내부에서 입력에 대해 가장 근접하는 요소를 k개 찾아, 일치하는 것(혹은 이를 기반으로 하여)으로 해당 요소를 분류하는 알고리즘이다. 분류 체계값을 모두 검사하여 높은 정확도를 지니고, 오류 데이터에 대해 비교대상에서 제외되는 특징이 있고, 데이터에 대해 거리로만 사용하므로 데이터에 대한 가정은 존재하지 않는다. 단, 데이터 양에 따라 처리시간이 크게 증가한다는 단점이 존재한다.

CNN

CNN(Convolutional Neural Network)는 이미지 분석 패턴을 찾는 데 유용한 알고리즘으로 학습을 통해 패턴을 학습, 이미지를 분류한다. 해당 기법은 convolution(필터를 일정간격으로 이동시키며 input data에 연산을 적용하고, 최종적으로 feature map을 뽑아내는 것)과 activation(활성화 함수로 데이터들에 대해 주로 비선형적으로 바꿔주는 작업을 하게 된다.), pooling(특정 feature만을 강조하기 위해 feature map에서 특정 범위를 기준으로 데이터를 뽑아내는 방식) 등 다양한 기법을 사용하게 된다. 다양한 기능을 사용해야 하는 만큼, model을 구성할 경우 이들을 layer로서 분류, layer를 쌓아가는 형태로 해당 기능들을 지원하게 된다.

위의 설명한 기능 외에 아래 모델을 구성하기 위해 다음과 같은 layer를 사용한다.

Flatten: convolution과 pooling 등을 통해 간소화된 이미지를 데이터에서 처리하기 위해 1차원 배열의 형태로 변경해주기 위해 사용된다. 위와 같은 역할을 하기에 pooling이나 convolution layer 이후에 추가된다.

Dense layer: 입력과 출력을 모두 연결하는 full associative형태를 띄는 layer이다. layer내의 뉴런들은 각각 가중치를 지니며 이 값을 이용해 입력/출력에 있는 뉴런들에 대해 영향력을 어느정도로 행사할지 정할 수 있다. 위에서 언급된 activation을 담고있는 layer이기도 하다 .

Dropout layer: Over fitting, 즉 train data에 대해 너무 과하게 학습되어 test에 적합한 결과를 내지 못하는 경우를 방지하기 위해 hidden layer의 일부를 작동하지 못하게 만드는 용도로 사용된다.

위와 같은 layer들이 쌓여 생겨난 model을 바탕으로 input을 받고 이를 바탕으로 propagation을 진행한다. 이렇게 생겨난 결과들을 바탕으로 좀 더 좋은 결과를 내는 모델로의 학습을 위해 layer를 역으로 거슬러 올라가며 가중치를 변화시키는 backpropagation을 진행하여 모델의 정확도를 올리게 된다. 이것을 반복하며 학습하는 것이 CNN의 기본적인 이론이다.

MinMaxScaler

0-1사이로 스케일링을 진행하는 것으로, 범위가 다른 Feature들에 대해 범위와 규칙을 일정하게 해주는 역할을 한다.

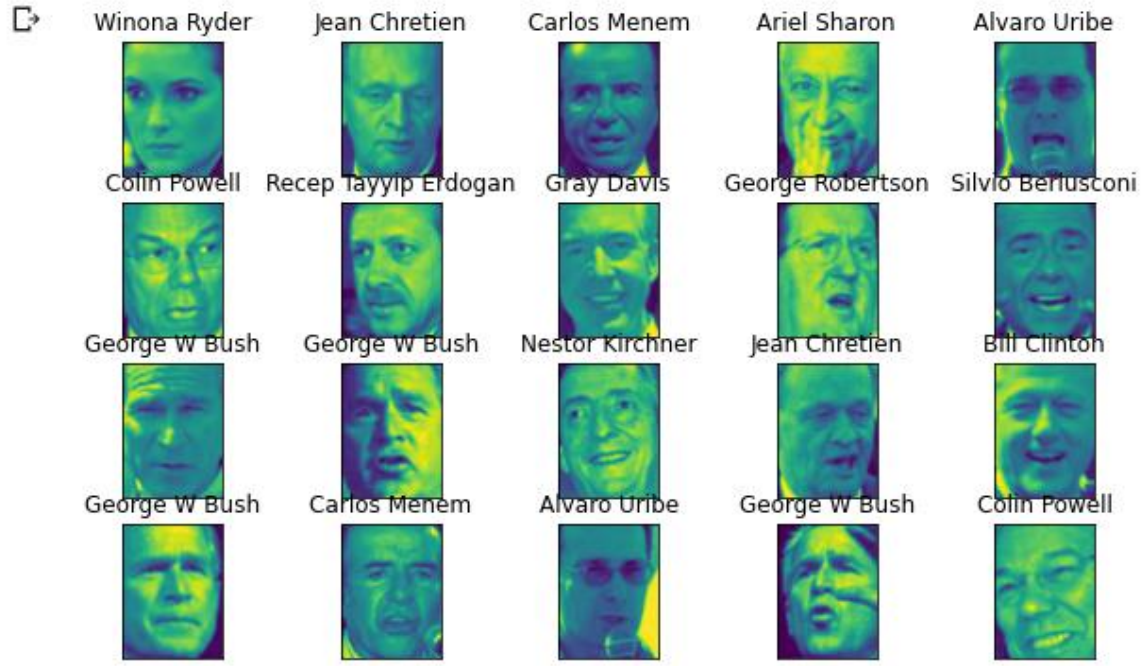
공식은 간단하게 (X - X중 최소값)/(X중 최대값 - X중 최소값)으로 진행이 된다.

Result

1.LWF(Labeled Faces In the World) Dataset

인공지능1.pdf에서 언급한 대로 fetch_lfw_people 함수를 통해 Dataset을 load하였다.

위의 내용들을 바탕으로 내부에 있는 이미지들을 출력해보았다.



(Dataset load와 데이터셋에 대한 설명)

PDF에 있는 이미지와 다른 이미지들이 출력되었다. 이를 기반으로 해당 데이터셋은 인물의 이름과 이미지가 한 세트로 묶여 있는 dataset임을 것을 알 수 있습니다. 또한 출력에서 pdf의 자료에 있는 인물 명과 겹치는 인물들(George W Bush 등)을 확인할 수 있다.

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

다음은 Data shape에 대한 출력을 진행한 것이다. 데이터 셋에 대한 수정이 있었는지, LWF Database는 현재 62개의 class의 데이터 3023으로 구성되어 있고, 각각의 얼굴의 이미지는 PDF와 동일하게 87*53로 구성되어 있다.

사진에 할당된 이름을 이용, 각 이름(label)당 이미지의 수를 출력한다.

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

위를 보면 데이터에 관한 숫자가 고르지 않은 것을 알 수 있다. 특히, 다수의 데이터는 100개미만의 개수를 가지지만, Colin Pwell 236, George W bush 530, Donald Rumsfeld 121, Gerhard Schroeder 109 등 이미지의 수가 3자리 수인 데이터들이 있는 걸 확인했다. 이를 통해 데이터셋에서 편중된 데이터가 있음을 확인할 수 있었다. 또한 위의 자료들을 통해 가장 적은 샘플은 20임을 알 수 있다.

위에서의 결과를 바탕으로 가장 적은 샘플을 기준, 20개의 데이터를 뽑아내며 data들에 대해 pixel scale을 기존 0-255이었던 값에서 0-1로 조정하였다. 출력 시 1240개의 데이터에 대해 people_image의 경우 이미지의 사이즈인 (87*65)를 받고 있으며, people_label또한 1240개의 labeling데이터를 받아 shape가 다음과 같아짐을 확인할 수 있었다.

2.PCA whitening & PCA

위에서 구해진 data에 대해 train_test_split함수를 사용, image와 label에 대해 test데이터와 train 데이터를 분류하였다.

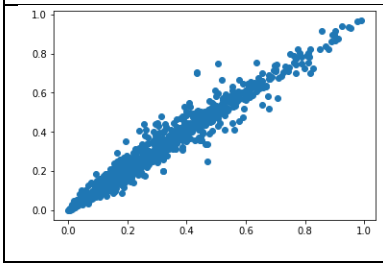
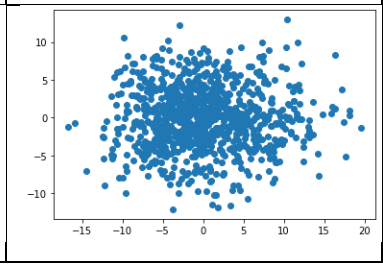
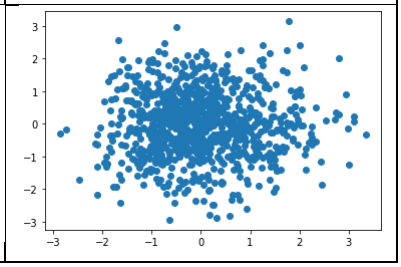
```
(930, 5655)
(930,)
(310, 5655)
(310,)
```

위의 결과로 train에 930, test에는 310개의 데이터가 배치된 것을 확인했다.

이후, 데이터에 대해 PCA모델을 적용하였다. 이때 whiten 값을 비교를 위해True와 false로나눴다.

또한 plt를 이용, scatter plot으로 이를 출력하고자 하였다. 또한 정확도를 출력하고자 knn모델을 사용하기 위해 선언 밑 fit를 진행하였다. (위 코드에는 KNeighbors Classifier을 사용하였다.)

(scatter plot을 이용한 PCA 화이트닝 옵션 적용 / PCA Whitening 사용)

Original data	Whitening X	Whitening O
		

Scatter plot에 대해서는 위와 같은 결과를 보였다.(component 1,2에 대해 출력) 출력된 Original data에 대해 2가지 벡터를 뽑아 출력한다고 가정할 경우, pca를 진행할 경우 새로운 기준 벡터가 생성되어 비교적 원형에 가깝게 변화한 것을 확인할 수 있었다. 또한 whitening을 적용할 경우 이에 더해 feature들에 대해 분산을 0으로 치환해 주므로 크기가 whitening을 하지 않은 경우보다 줄어든 것을 확인할 수 있었다.

(KNN 모델을 학습시키는 과정에서 PCA Whitening의 여부를 통한 정확도 차이 비교)

MinMaxScaler X	MinMaxScaler O
accuracy(whitening 0) : 0.24194 accuracy(whitening X) : 0.20323	accuracy(whitening 0) : 0.26129 accuracy(whitening X) : 0.22258

위는 정확도에 대해 출력한 결과이다. Whitening을 하는 경우 더 높은 정확도를 얻는 것이 확인되었다.

또한 위과정에서 MinMaxScaler를 사용, data preprocessing여부를 비교해보았다. 기존 데이터는 feature마다 수치 범위가 다르나, 해당 과정을 거치며 Feature들에 대해 정규화가 된 후, 정확도가 상승하는 것을 확인할 수 있었다.

이번엔 pca시 주성분의 개수를 다르게 하여 출력해보았다. 조건은 whitening이 true인 상태에서 다른 옵션은 유지한 채 n_components만 수정하였다.

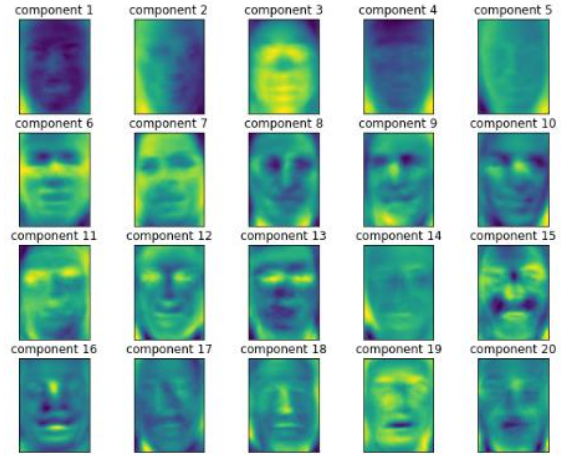
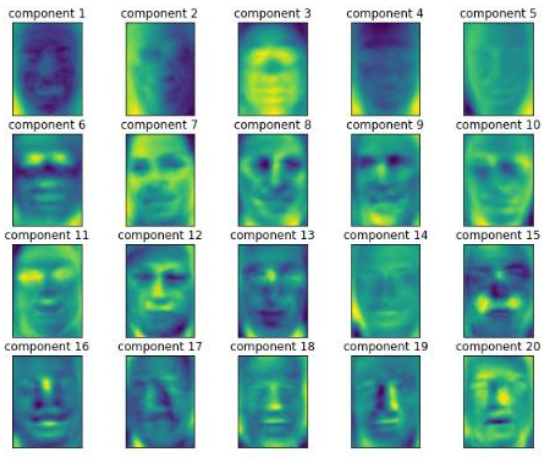
MinMaxScaler X	MinMaxScaler O
accuracy(component 62) : 0.24516 accuracy(component 10) : 0.15161 accuracy(component 50) : 0.22903 accuracy(component 100) : 0.24194 accuracy(component 200) : 0.20323 accuracy(component 500) : 0.09355	accuracy(component 62) : 0.24516 accuracy(component 10) : 0.15161 accuracy(component 50) : 0.24516 accuracy(component 100) : 0.26129 accuracy(component 200) : 0.20968 accuracy(component 500) : 0.09355

결과는 위와 같이 어느정도까지는 n_component 수를 올리면 정확도가 올라가지만, 특정 시점에 도달하면 점점 정확도가 떨어지는 것을 확인할 수 있었다. 이는 curse of dimensionality 때문이다. 분류기준이 많아질수록 구체적인 분류가 가능해 performance가 올라가나 어느순간 기준의 수가 필요 이상으로 많아지며 performance가 저하되는 것이다.

마찬가지로 위과정에서 MinMaxScaler를 사용, data preprocessing여부를 비교해보았다. Component를 100으로 고정했을 때와 마찬가지로 component수가 50, 100, 200일 때 더 높은 정확도를 보이는 것을 확인하였다.

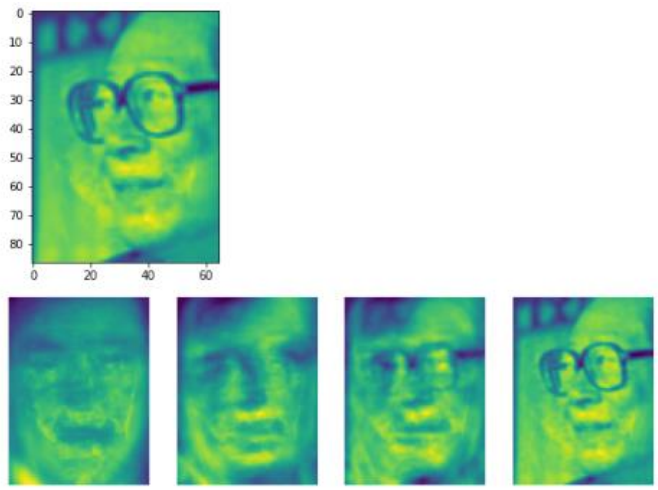
위에서 진행한 pca(whitening = true, component=100)를 바탕으로 주성분을 출력해보았다.

(PCA를 하여 고유 얼굴 성분을 출력)

MinMaxScaler X	MinMaxScaler O
	

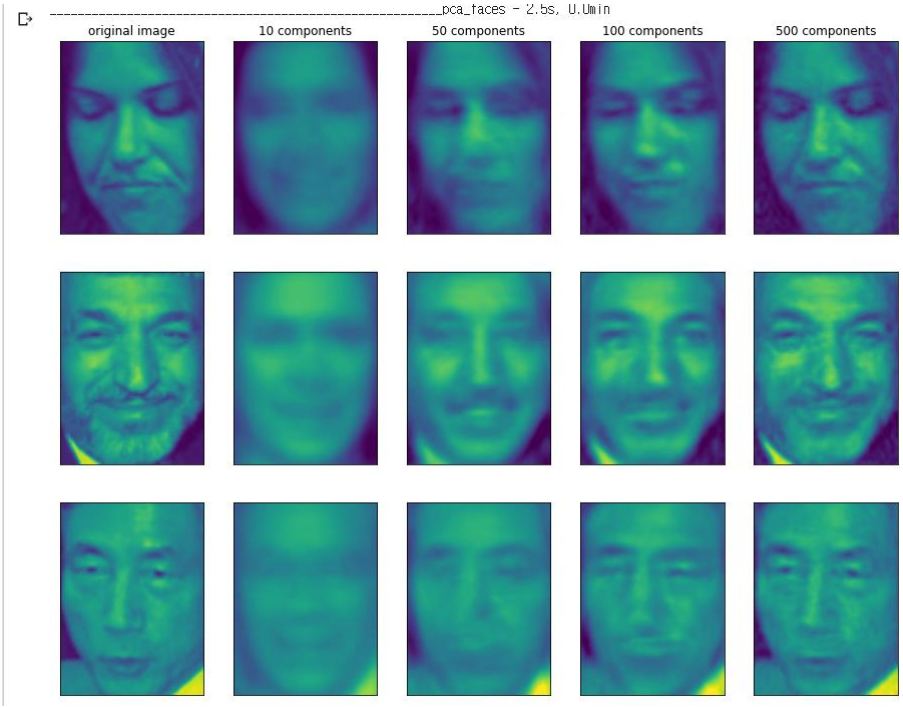
주성분들이 정상적으로 뽑힌 것을 확인했다.

(주성분의 개수의 차이를 두어 얼굴 이미지를 재구성 및 결과 설명).



위는 original data와 pca의 n_component를 각각 10, 50, 100, 500으로 한 data들을 기반으로 inverse_transform을 진행한 결과이다. Pca시 n_component 수를 늘릴수록 좋은 기존 이미지에 가까워짐을 확인할 수 있었다. 이러한 기능을 지원하는 함수로도 이를 확인할 수 있다.

Mglearn에 내장되어 있는 pca를 기반으로 하는 얼굴에 대한 재구성을 위한 함수(plot_pca_faces)를 사용하기 위해 환경을 맞추고 train data와 test data를 넣어 위 함수를 통해 확인하였다



결과는 위와 같다. Reconstruction에 대해 사용하는 component가 10, 50, 100, 500인 경우에 대해 이미지 재구성을 진행함을 확인할 수 있었다. Reconstruction의 경우 component, 즉 eigenimages에 대해서 개수가 많을수록 더 높은 복구율을 보이는 것을 확인할 수 있었다.

3.KNN & CNN

KNN의 경우 위를 통해 정확도를 구해내는 것에 성공했다. 위 결과를 바탕으로 pca시 주성분의 개수를 class의 수에 가깝게 할수록, whitening을 진행한 경우 더 높은 정확도를 보이는 것을 확인했다.

(KNN 모델을 생성하여 학습시켜 결과를 설명)

MinMaxScaler X	MinMaxScaler O
accuracy(whitening O) : 0.24194 accuracy(whitening X) : 0.20323 accuracy(component 62) : 0.24516 accuracy(component 10) : 0.15161 accuracy(component 50) : 0.22903 accuracy(component 100) : 0.24194 accuracy(component 200) : 0.20323 accuracy(component 500) : 0.09355	accuracy(whitening O) : 0.26129 accuracy(whitening X) : 0.22258 accuracy(component 62) : 0.24516 accuracy(component 10) : 0.15161 accuracy(component 50) : 0.24516 accuracy(component 100) : 0.26129 accuracy(component 200) : 0.20968 accuracy(component 500) : 0.09355

(CNN를 학습시켜 결과를 비교, 그림 9, 10을 참고해 Feature 추출 방법, 성능 비교)

CNN model을 위해 위와 같이 데이터에 대해 data 전처리를 하였다. 위 과정에서는 MinMaxScaler를 이용해 스케일링을 진행, 값을 0과 1사이로 정의하고 label데이터를 기반으로 계층화를 진행하였다. 이후 train/test 데이터로 나누어주었다.

to_categorical을 이용, label데이터를 one-hot의 형태로 변경하고 image데이터에 대해선, reshape를 이용, 1차원 데이터로 되어있던 image데이터를 2차원형태로 변경해주었다.(5655->87*65)

```
(930, 87, 65)
(310, 87, 65)
(930, 62)
(310, 62)
```

변경한 data의 shape는 위와 같다.

이후 model 계층을 pdf와 동일하게 설정해주었다. Model은 sequential의 형태로 쌓이게 되며, 내부에는 conv2d(convolution layer), maxpooling2D, Dropout, Flatten, Dense, Dropout layer가 사용되었다.

단, pdf와 다르게 마지막 dense layer, 즉 output layer에 대해 수치를 변경하였는데, 이는 pdf에서 보이는 class수 53과 다르게 class수가 62이기 때문에 해당 데이터에 맞추어 변경을 해준 것이다.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 87, 65, 32)        832

max_pooling2d_2 (MaxPooling  (None, 43, 32, 32)        0
2D)

conv2d_3 (Conv2D)            (None, 43, 32, 64)        8256

max_pooling2d_3 (MaxPooling  (None, 21, 16, 64)        0
2D)

dropout_2 (Dropout)          (None, 21, 16, 64)        0

flatten_1 (Flatten)          (None, 21504)             0

dense_2 (Dense)              (None, 32)                688160

dropout_3 (Dropout)          (None, 32)                0

dense_3 (Dense)              (None, 62)                2046

=================================================================
Total params: 699,294
Trainable params: 699,294
Non-trainable params: 0
_________________________________________________________________
```

Summary를 통해 생성된 model을 확인하면 위와 같다. Pdf와 동일하게 작성됨을 확인할 수 있다.

위에서 생성된 model을 기반으로 pdf에서 정의한 설정에 맞춰 model을 fit하였고, 이후 test data를 이용해 evaluate, 즉 평가를 진행하였다. 이때 손실을 판단하는 척도로서 categorical crossentropy, 즉 다중 분류문제에 특화된 loss를 사용하였으며 optimizer로서는 adam을 사용하였다.

또한 epochs, 즉 전체데이터에 대해 20번 진행하였고, batch_size는 4로 지정하였다.

(CNN 모델 학습, 결과 비교)

```
Epoch 1/20
233/233 [==============================] - 14s 9ms/step - loss: 4,1360 - accuracy: 0,0151
Epoch 2/20
233/233 [==============================] - 2s 7ms/step - loss: 4,1326 - accuracy: 0,0054
Epoch 3/20
233/233 [==============================] - 2s 9ms/step - loss: 4,1289 - accuracy: 0,0108
Epoch 4/20
233/233 [==============================] - 2s 10ms/step - loss: 4,0645 - accuracy: 0,0387
Epoch 5/20
233/233 [==============================] - 1s 5ms/step - loss: 3,8213 - accuracy: 0,0667
Epoch 6/20
233/233 [==============================] - 1s 5ms/step - loss: 3,5196 - accuracy: 0,1183
Epoch 7/20
233/233 [==============================] - 1s 5ms/step - loss: 3,2539 - accuracy: 0,1667
Epoch 8/20
233/233 [==============================] - 1s 5ms/step - loss: 3,0927 - accuracy: 0,1871
Epoch 9/20
233/233 [==============================] - 1s 5ms/step - loss: 2,8763 - accuracy: 0,2280
Epoch 10/20
233/233 [==============================] - 1s 5ms/step - loss: 2,6634 - accuracy: 0,2849
Epoch 11/20
233/233 [==============================] - 1s 5ms/step - loss: 2,5079 - accuracy: 0,3000
Epoch 12/20
233/233 [==============================] - 1s 5ms/step - loss: 2,4138 - accuracy: 0,3301
Epoch 13/20
233/233 [==============================] - 1s 5ms/step - loss: 2,3131 - accuracy: 0,3366
Epoch 14/20
233/233 [==============================] - 1s 5ms/step - loss: 2,1530 - accuracy: 0,3774
Epoch 15/20
233/233 [==============================] - 1s 5ms/step - loss: 2,0905 - accuracy: 0,4000
Epoch 16/20
233/233 [==============================] - 1s 5ms/step - loss: 1,9618 - accuracy: 0,4355
Epoch 17/20
233/233 [==============================] - 1s 5ms/step - loss: 1,9543 - accuracy: 0,4258
Epoch 18/20
233/233 [==============================] - 1s 5ms/step - loss: 1,8485 - accuracy: 0,4645
Epoch 19/20
233/233 [==============================] - 1s 5ms/step - loss: 1,7354 - accuracy: 0,4957
Epoch 20/20
233/233 [==============================] - 1s 5ms/step - loss: 1,7472 - accuracy: 0,4806
Test loss: 3,841670513153076
Test accuracy 0,23548386991024017
```

결과는 위와 같다. Loss와 accuracy에 대해 각각 3.8416, 0.23548386... 이라는 결과를 내었다.

MinMaxScaler X	MinMaxScaler O
accuracy(component 62) : 0.24516 accuracy(component 10) : 0.15161 accuracy(component 50) : 0.22903 accuracy(component 100) : 0.24194 accuracy(component 200) : 0.20323 accuracy(component 500) : 0.09355	accuracy(component 62) : 0,24516 accuracy(component 10) : 0,15161 accuracy(component 50) : 0,24516 accuracy(component 100) : 0,26129 accuracy(component 200) : 0,20968 accuracy(component 500) : 0,09355

이는 위에서 확인한 PCA를 통해 Feature를 추출한 KNN에서의 정확도이다. 최종적인 정확도는 KNN은 동일한 component 변경, whitening진행에 관계없이 CNN모델에서 batch_size=4, epoch 20에 대해 정확도가 낮게 측정되는 것을 확인할 수 있었다. (epoch 9에서 정확도가 KNN때의 최대를 넘긴다.)

단, layer가 쌓여 만들어지고, Propagation과 Backpropagation을 통해 가중치 변경이 진행되는 만큼, CNN이 PCA를 통해 Feature를 추출한 KNN에 비해 시간이 길게 드는 것을 확인할 수 있었다.

Consideration

해당 과제를 진행하며 CNN과 KNN을 사용하여 보았고, 이를 바탕으로 가장 기초적인 인공지능 구조들이 어떤 식으로 동작이 되는 지 대략적으로 알 수 있었습니다. 특히 KNN의 PCA나 CNN에서의 MinMaxScaler를 사용하고 이에 따라 정확도의 차이가 나는 것을 보며 Data 전처리의 중요성을 어느정도 알 수 있었습니다. 또한 component차이와 whitening여부에 따라서도 큰 정확도 차이가 나는 걸 보며 어떤 기법을, 어떤 기준으로 사용하는 지가 좋은 결과를 내는 데 크게 영향을 미친다는 점에서도 해당 선택의 중요성을 다시금 알 수 있었습니다. 이를 바탕으로 이후 인공지능 수업에서 이론에 대한 이해도를 높여가는 것이 가능할 것 같습니다.

Reference

Chapter 1 (Intro).pdf

Principal Component Analysis.pdf (PCA정의)

https://youngq.tistory.com/m/40 (CNN정의)

https://mongxmongx2.tistory.com/m/26 (Dropout정의)

https://computer-science-student.tistory.com/m/56 (KNN정의)

https://velog.io/@yuns_u/flatten%EA%B3%BC-dense (flatten dense 정의)