# Face Recognition With Olivetti Dataset

Luís Fonseca
*89066*
*DETI*
Aveiro, Portugal
luiscdf@ua.pt

Pedro Marques
*89069*
*DETI*
Aveiro, Portugal
pedromm@ua.pt

***Abstract*—Há já muitos anos vários estudos têm sido feitos no que diz respeito ao campo de reconhecimento facial. O primeiro estudo remonta aos anos 60, tendo sido feito por Woody Bledsoe. Através das melhorias tecnológicas e aumento do poder computacional que se deu desde então, mais e melhores técnicas podem ser aplicas. O presente documento pretende fazer um estudo a este problema de Face Recognition, fazendo uso de um dataset também antigo, contendo faces de 40 pessoas distintas.**

***Index Terms*—face recognition, machine learning, knn, neural networks, classification, supervised learning, pca**

## I. Introduction

Este relatório tem como objetivo explicar os métodos adotados aquando da elaboração do segundo projeto da cadeira de TAA - Tópicos De Aprendizagem Automática - da Universidade de Aveiro. Neste contexto, trabalhou-se sobre o dataset sugerido pela docente, - Olivetti faces - um dataset relativamente antigo, uma vez que consiste em fotografias faciais tiradas entre abril de 1992 e abril de 1994 em AT&T Laboratories Cambridge.

Assim, nos capítulos seguintes será demonstrado o processo tomado aquando da resolução do problema, começando por referir trabalhos relacionados, e de que forma foram tomados como ponto de partida, análise dos dados, implementação dos métodos e finalmente conclusão do resultados obtidos.

## II. Os Dados

### A. Descrição

O próprio website oficial [1] do dataset faz já uma descrição bastante detalhada do dataset, como podemos ver:

> There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

Contudo, por conveniência, não se fez uso dos dados fornecidos pela professora, tendo-se, tendo-se optado por adotar outra fonte [11]. Note-se que o dataset em si não é diferente, apenas a estrutura como estes se encontram mudou, passando de uma estrutura em folders para ficheiros *.npy*. Esta alteração visou apenas a evitar colocar foco no *load* dos dados, permitindo mais rapidamente passar ao tratamento e estudo dos mesmos. Desta forma, deparamo-nos com 2 ficheiros:

- olivetti_faces.npy
- olivetti_faces_target.npy

Correspondendo o primeiro aos dados em si, ou seja as fotografias das faces, e o segundo a identificação (em classes) dessas mesmas caras.

Posto isto, é possível apresentar este Olivetti Dataset com as seguintes propriedades:

- 10 imagens diferentes de cada uma das 40 pessoas
- total de 400 imagens
- luminosidade, tempo, expressão facial variante
- background negro
- tamanho 64*64
- identificação das pessoas feita por um inteiro no intervalo [0, 39]
- pixeis na escala [0, 1]

Posto isto há duas coisas importantes a referir:

No terceiro ponto, note-se que as alterações nas condições das imagens tem como objetivo diversificar ao máximo os dados, de forma a permitir que o modelo treinado consiga generalizar o mais possível.

No último ponto, há que ter especial atenção, uma vez que efetua logo um passo importante aquando do tratamento dos dados: normalização. Este tipo se *rescaling* tem como objetivo normalizar todos os pixeis das imagens tal que encaixem num certo intervalo (no caso, entre 0 e 1), sendo conhecido por *min-max normalization* [8]. Este processo tem como objetivo aumentar a eficácia e eficiência das técnicas de ML aplicadas.


Fig. 1. As 40 faces do dataset

### B. Análise

Posto isto, torna-se evidente a principal preocupação no dataset - a sua diminuta dimensão. Por outro lado, é possível verificar que há uma boa distribuição dos dados pelas classes - ou seja, as pessoas. Assim, com o dataset atual, optou-se por deixar 30% dos dados para testing e os restantes para training. Isto corresponde a basicamente 3 e 7 imagens, para cada "fase", respetivamente.

Uma das bibliotecas usadas - sklearn - possui ferramentas que permitem fazer esta separação dos dados por classe, garantindo que se "extraí" para testing o mesmo número de imagens por classe.

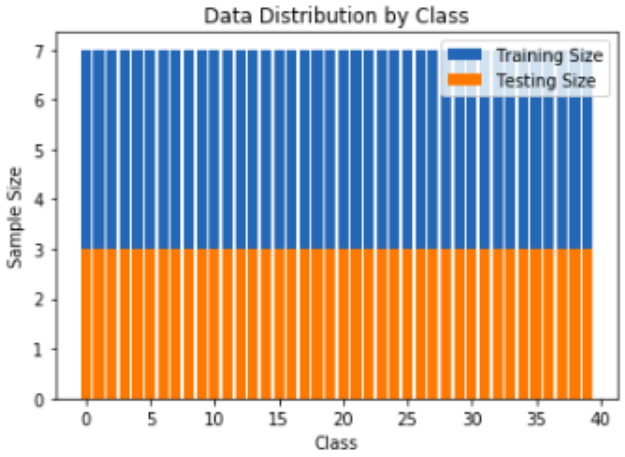


Fig. 2. Número de imagens usadas para treino e teste

## III. Modelos Aplicados

### A. Trabalho Relacionado

Antes de começar a implementar qualquer técnica para resolução deste problema, considerou-se importante primeiro procurar artigos sobre este problema de classificação, e em concreto de caras humanas.

Algo interessante, que era praticamente constante em todos os artigos relacionados com reconhecimento facial encontrados [2] [3] [4] era o facto de trabalharem com uma técnica conhecida por PCA - Principal Component Analysis.

Além disso, um dos artigos [2] salientava um outro método - LDA - que se acredita ser capaz de *out perform* PCA. Contudo, afirmava também que pesquisas indicavam que para training datasets pequenos, PCA poderia apresentar melhor performance.

Considerando isto e tendo em conta o reduzido tamanho do dataset com o qual se está a trabalhar, pareceu óbvio que PCA seria a técnica que devíamos eventualmente adotar.

Relativamente à "segunda" parte do processo, ou seja a aplicação de um método de ML para a *classificação* em si, já não havia tanto consenso, tendo sido aplicado KNN - K-Nearest Neighbor - por vezes até sem o uso de PCA, ou NN.

Desta forma, decidiu-se que o melhor percurso a tomar seria aplicar estas duas técnicas, ou seja KNN e NN.

### B. Considerações

À semelhança do projeto anterior, aquando do uso da livraria *sklearn*, não é possível observar a evolução da função de custo por iteração. Desta forma, tendo em conta que o dataset é equilibrado, optou-se por usar a accuracy como métrica principal na avaliação dos resultados obtidos. Tal acontece durante o estudo de método de KNN. Aquando a aplicação de NN, tal já não acontece.

### C. Cross-Validation

Tendo em conta que apenas temos 10 imagens faciais por pessoa, aquando da escolha do método usada por CV, optou-se por usar o *Leave One Out* - como inclusive recomendado nos slides da cadeira.

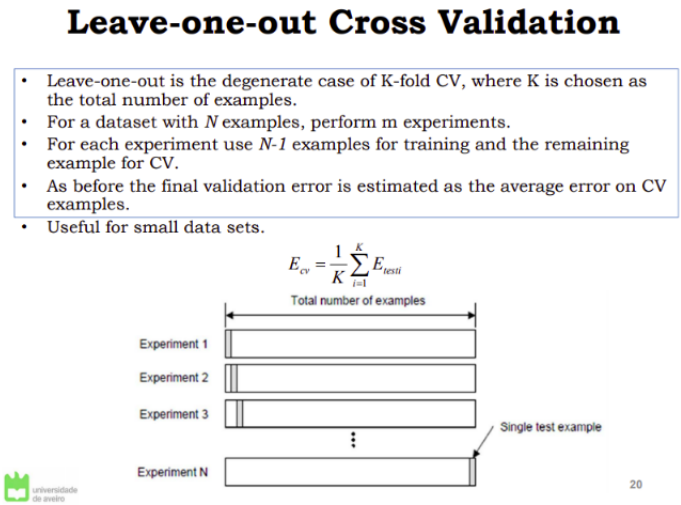


Fig. 3. Slide sobre Leave one Out CV

Neste tipo de *cross validation*, apenas um exemplo é usado para validação sendo os restantes usados para treino. Este procedimento é repetido de forma a que cada exemplo seja usado para validação. Assim, uma técnica que seria computacionalmente dispendiosa num dataset maior, neste contexto apresentar-se como um procedimento bastante rápido.

## IV. Principal Component Analysis

Como referido, tendo em conta que o dataset é bastante pequeno em tamanho, o número de features (tamanho das imagens) acaba por se fazer sentir bastante notório - 64*64=4096, pelo que a aplicação do método de unsupervised learning PCA, pode ajudar no sentido de "reduzir o tamanho" das imagens, selecionando apenas as features mais importantes.

Segundo o paper [2], PCA é usada para "descrever" imagens em termos de um conjunto de funções base conhecidas como "eigenfaces" - para mais detalhe sobre este termo, ver [3].

Este conceito apresenta-se como um dos usos mais poderosos de Principal Components Analysis no que diz respeito a reconhecimento (e deteção) de faces. Como se trata de um método unsupervised, não precisa de conhecer as classes (ou seja, pessoas) às quais as faces se referem. A figura 4 pretende ilustrar o resultado da aplicação da PCA.

Fig. 4. Exemplo de algumas Eigenfaces sobre o Dataset

### A. Number of Components

Neste sentido, surgiu a necessidade de selecionar o número de componentes ideal para o qual reduzir as imagens. Como isto era informação que não tínhamos à partida - uma vez que os papers analisados não referiam tal informação, ou tratavam-se de datasets distintos, considerou-se interessante analisar a forma como PCA funcionava. Desta forma, no sentido de visualizar o resultado deste algoritmo, selecionou-se 2 como o número de componentes, de forma a ser possível visualizar graficamente esta informação.

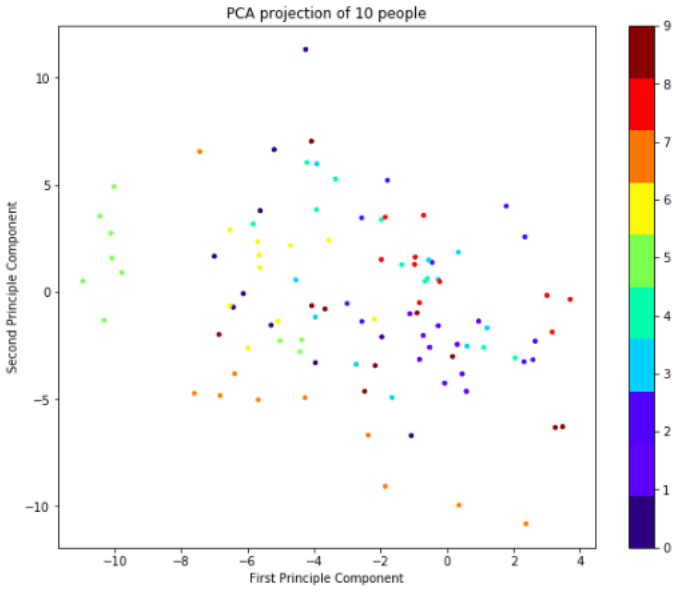


Fig. 5. Número de imagens usadas para treino e teste

Tal leva-nos à figura 5. Esta representa as 10 imagens de 10 pessoas, distribuídas por 2 componentes, como pretendido. Como se pode observar, enquanto alguns pontos da mesma classe se encontram bastante próximos, existem outros bastante mais dispersos. Da mesma maneira, existem pontos de classes distintas bastante próximos e até "misturados" uns com os outros. Embora se esteja a visualizar apenas a 2 dimensões, esta observação poderia vir a ser útil no futuro.

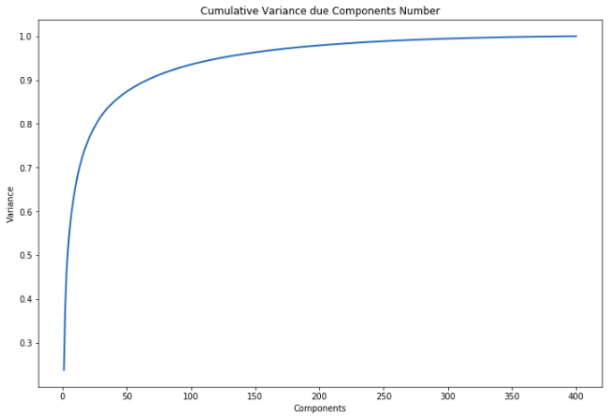


Fig. 6. Somatório da Variância em função do Número de Componentes

Segundo várias fontes encontradas pela Internet [5] [6] [7], não existe uma maneira "correta" para usar no momento de selecionar o número de componentes. Desta forma, decidiu-se optar pela "heurística" de *Percentage of Cumulative Variance*. Visando então a encontrar o número de componentes ideal, que representasse melhor as imagens, estudou-se a "explained variance". Este conceito está relaciona o somatório da variância dos dados originais em função do número de componentes [9]. Exemplificando, a *cumulative variance* da segunda componente é a soma da variância da primeira e da segunda componentes.

O estudo desta "heurística" encontra-se apresentada no gráfico da figura 6.

Observando o mesmo, nota-se que o crescimento no eixo das ordenadas se torna bastante lento, a partir de certo ponto. Assim, assumindo que um valor válido para o número de componentes a ser escolhido passa por um valor de variance acima de 0.95, acabou por se escolher 100 como valor inicial para o número de componentes.

## V. K-Nearest Neighbors

Passando agora à fase de classificação do problema em si, o primeiro algoritmo a ser usado é o de k-nearest neighbors (KNN).

A razão por detrás de ser o primeiro, passa pelo facto de que se tratar de um algoritmo de aprendizagem supervisionada simples e fácil de implementar. Adicionalmente, é também um algoritmo que não foi usado no projeto anterior, daí ter despertado bastante interesse.

Posto isto, ao trabalhar com KNN "assume-se" que se trabalha com pontos da mesma classe próximos uns dos outros. Isto porque a forma como o mesmo funciona para classificar um novo exemplo, passa por calcular a distancia a todos os pontos já conhecidos, identificar os $k$ vizinhos mais próximos, e classificar esse exemplo com base na classe mais representada nesses $k$ vizinhos.

### A. *Escolha de K*

Posto isto, para o número de componentes de PCA, temos n_components = 100. Assim estudou-se a variação da accuracy em função do valor de K, isto é, do número de vizinhos. Tal pode-se ver-se no gráfico 7.

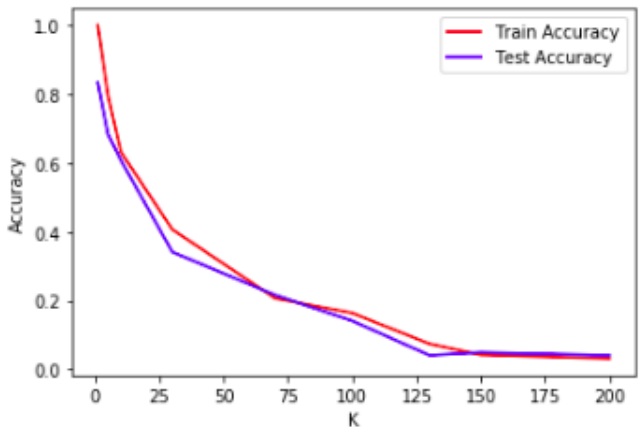


Fig. 7. Primeira aplicação de KNN

Como se pode observar, a accuracy reduz bastante à medida que se aumenta o número de vizinhos a ter em conta. Como tal, torna-se relevante estudar o comportamento do mesmo para valores de *K* mais pequenos. Esse estudo encontra-se no gráfico 8.

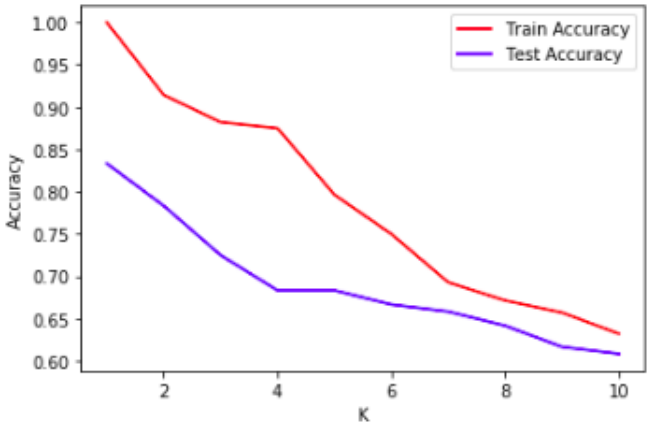


Fig. 8. KNN aplicado a K mais pequenos

Como se pode perceber, valores mais reduzidos de *K* têm maior valores de accuracy, tanto para treino como para teste. Não obstante, é óbvio o problema de overfitting. Uma vez que o modelo em questão é bastante simples, muito poucas medidas podem ser feitas visando a corrigir este problema, já que o único hiperparâmetro é *K*.

### B. *Voltando a PCA*

Posto isso, considerou-se relevante dar um "passo atrás" e verificar se se conseguia corrigir este problema alterando os número de componentes de PCA. Para tal, começou-se por aplicar KNN sem qualquer uso de PCA.

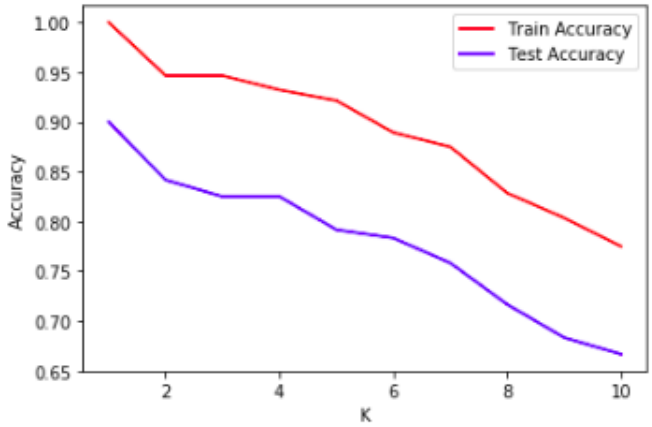


Fig. 9. KNN aplicado a K mais pequenos sem PCA

Como observado na figura 9, o facto de os valores de accuracy aumentarem em cerca de 5% revelou-se uma surpresa, uma vez que correspondem ao uso de dados sem este pré-processamento. Tendo em conta a explicação em [10], PCA não tem necessariamente uma correlação com a accuracy.

Posto isto, tornou-se necessário fazer um estudo mais detalhado sobre a forma como PCA poderia influenciar os resultados. Assim começou-se por estudar a variação da accuracy em função do número de componentes, para K = 1,3,5,7 vizinhos. A escolha deste número ímpar de vizinhos é propositada pois tem o objetivo de "desempatar" a escolha, quando por exemplo duas classes disputam a primeira posição para maior número de vizinhos. Este estudo encontra-se representado nos gráficos 10, 11, 13 e 14.

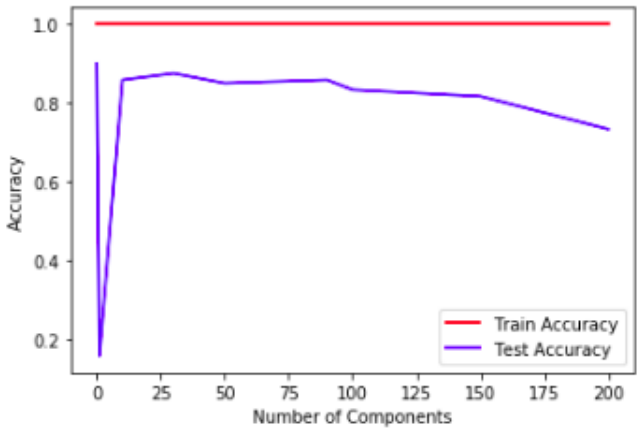


Fig. 10. Accuracy em função de Numero de Componentes para K = 1

Um fator importante a referir aqui, é que os primeiros 2 valores no eixo de *Number of Components* são 0 e 1. Zero corresponde à não aplicação do método PCA, enquanto 1 corresponde à aplicação de apenas uma componente. Assim, a razão para aquele "drop" acentuado passa apenas pelo facto de que, reduzindo a dimensão da imagem para um número muito diminuto de *features* (como 1, no caso) não permite representar

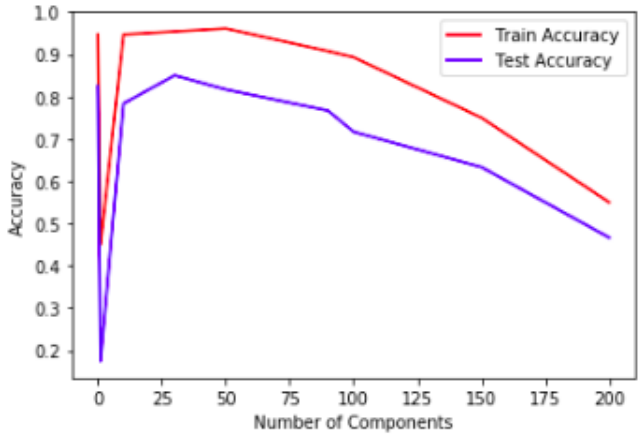


Fig. 11. Accuracy em função de Numero de Componentes para K = 3

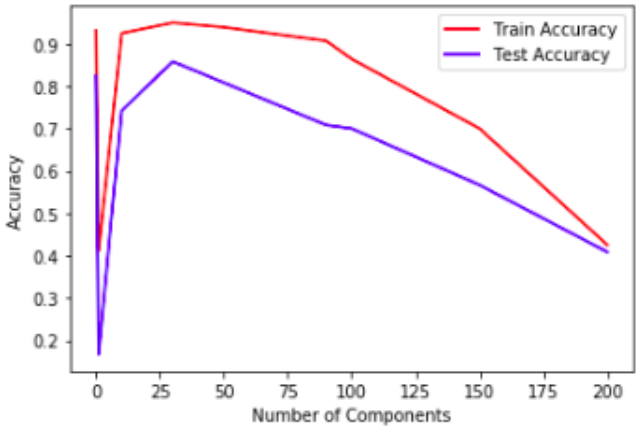


Fig. 12. Accuracy em função de Numero de Componentes para K = 4

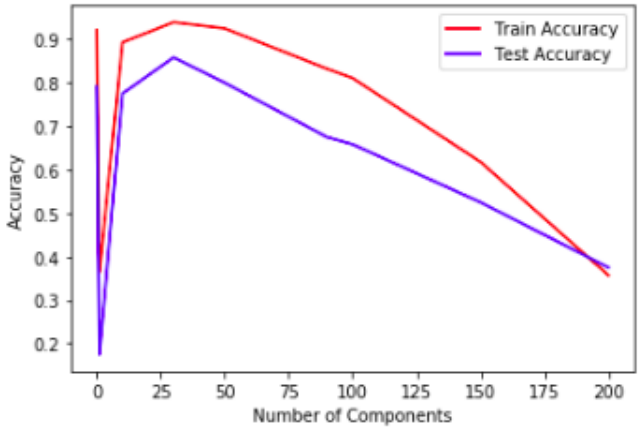


Fig. 13. Accuracy em função de Numero de Components para K = 5

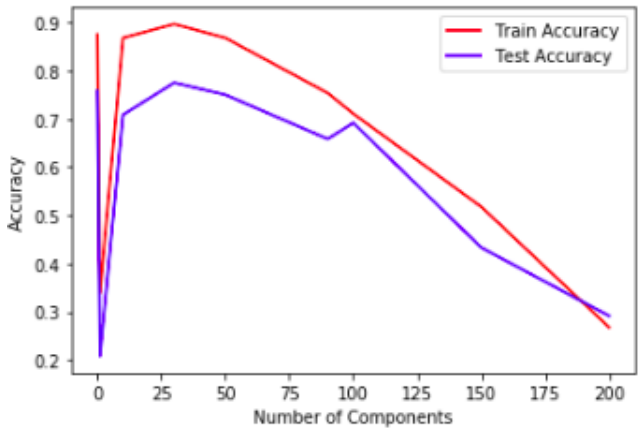


Fig. 14. Accuracy em função de Numero de Componentes para K = 7

TABLE I
RESULTADOS DOS MELHORES VALORES DE ACCURACY DE TESTE PARA CADA K

| K | Train Accuracy | Test Accuracy | n_components |
|---|---|---|---|
| 1 | 1.0 | 0.9 | 0 |
| 3 | 0.9535714285714286 | 0.866(6) | 30 |
| 4 | 0.95 | 0.8583(3) | 30 |
| 5 | 0.925 | 0.85 | 30 |
| 7 | 0.9107142857142857 | 0.775 | 30 |

bem uma imagem, afetando negativamente a accuracy.

Ignorando, de momento, as accuracies sem PCA e comparado os gráficos, vê-se rapidamente que o valor de K mais promissor é será entre 3 e 5. Da mesma maneira, o melhor n_components parece rondar 30. Com o intuito de verificar qual era o valor ideal de K, decidiu-se também averiguar para o valor K=4. Este caso foi já incluído no conjunto de gráficos apresentado, tratando-se no caso do gráfico 12. Analisando a tabela I, é possível verificar que de facto, é para K=3 que se obtêm os melhores resultados. Desta forma, um estudo mais cuidadoso deve ser feito à volta destes valores (K=3 e n_components = 30).

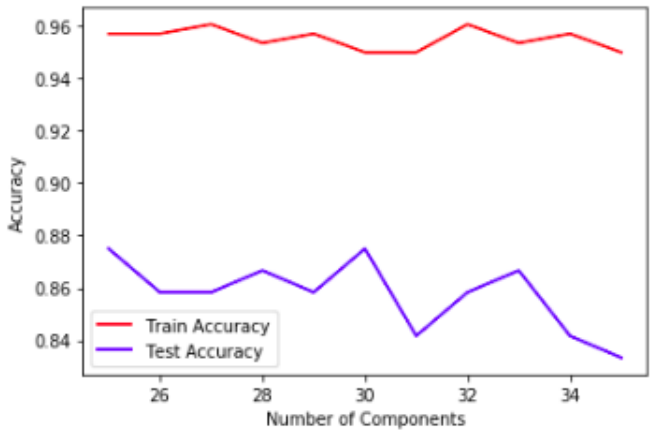


Fig. 15. Accuracy em função de um Intervalo mais Reduzido Numero de Componentes para K = 3

TABLE II
RESULTADOS DO MELHOR VALOR DE ACCURACY DE TESTE PARA CADA K = 3 APÓS TUNNING

| K | Train Accuracy | Test Accuracy | n_components |
|---|---|---|---|
| 3 | 0.9571428571428572 | 0.875 | 25 |

Tal estudo pode-se observar no gráfico 15. Como podemos ver, o valor que maximiza a accuracy acontece quando n_components = 25, assumindo o valor de 0.875. Tendo em conta que se obteve um aumento bastante ligeiro da accuracy com este tunning, e tendo em conta todas as verificações feitas até ao momento, pode-se assumir que não é eficiente continuar a maximizar a accuraccy deste modelo.

Assim, para o modelo em questão obtemos como melhor resultado os valores apresentados na tabela II, considerando a aplicação de PCA. Isto leva-nos à seguinte matriz de confusão (figura 16) e "classification report" (figura 17).

Contudo, a melhor accuracy ocorreu sem o uso de PCA, como indicado na primeira linha da tabela I, com uma accuracy de 90%. Desta forma, apresentam-se as figuras 18 e 19, revelando a matriz de confusão e o "classification report" deste modelo, respetivamente.

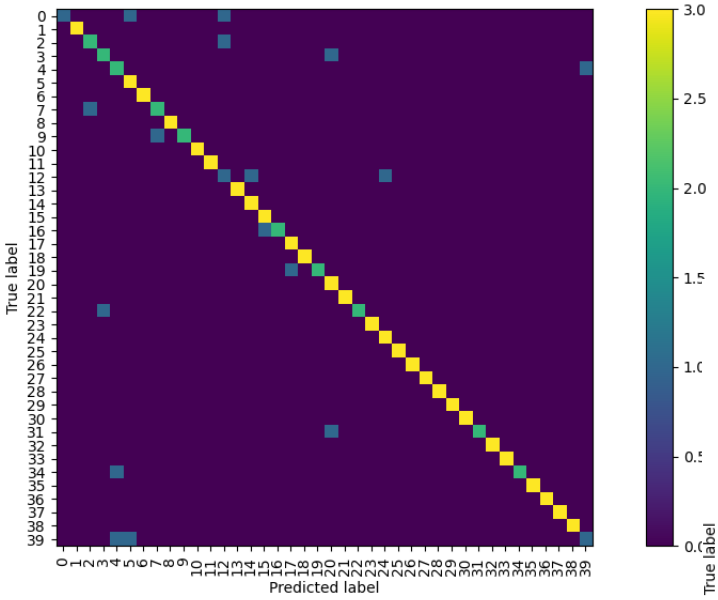


Fig. 16. Matriz de Confusão do Modelo Final de KNN com PCA

## VI. NEURAL NETWORKS

### A. Resumo do trabalho feito com Neural Networks

Quanto a aplicação de redes neurais como referido anteriormente, decidimos optar pela criação de modelos de redes neurais convolucionais, visto que estas trabalham bastante bem com imagens. Foram então criados 3 modelos, todos com estrutura e número de layers diferentes, mas com os mesmos hyperparameters.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.33 | 0.50 | 3 |
| 1 | 1.00 | 1.00 | 1.00 | 3 |
| 2 | 0.67 | 0.67 | 0.67 | 3 |
| 3 | 0.67 | 0.67 | 0.67 | 3 |
| 4 | 0.50 | 0.67 | 0.57 | 3 |
| 5 | 0.60 | 1.00 | 0.75 | 3 |
| 6 | 1.00 | 1.00 | 1.00 | 3 |
| 7 | 0.67 | 0.67 | 0.67 | 3 |
| 8 | 1.00 | 1.00 | 1.00 | 3 |
| 9 | 1.00 | 0.67 | 0.80 | 3 |
| 10 | 1.00 | 1.00 | 1.00 | 3 |
| 11 | 1.00 | 1.00 | 1.00 | 3 |
| 12 | 0.33 | 0.33 | 0.33 | 3 |
| 13 | 1.00 | 1.00 | 1.00 | 3 |
| 14 | 0.75 | 1.00 | 0.86 | 3 |
| 15 | 0.75 | 1.00 | 0.86 | 3 |
| 16 | 1.00 | 0.67 | 0.80 | 3 |
| 17 | 0.75 | 1.00 | 0.86 | 3 |
| 18 | 1.00 | 1.00 | 1.00 | 3 |
| 19 | 1.00 | 0.67 | 0.80 | 3 |
| 20 | 0.60 | 1.00 | 0.75 | 3 |
| 21 | 1.00 | 1.00 | 1.00 | 3 |
| 22 | 1.00 | 0.67 | 0.80 | 3 |
| 23 | 1.00 | 1.00 | 1.00 | 3 |
| 24 | 0.75 | 1.00 | 0.86 | 3 |
| 25 | 1.00 | 1.00 | 1.00 | 3 |
| 26 | 1.00 | 1.00 | 1.00 | 3 |
| 27 | 1.00 | 1.00 | 1.00 | 3 |
| 28 | 1.00 | 1.00 | 1.00 | 3 |
| 29 | 1.00 | 1.00 | 1.00 | 3 |
| 30 | 1.00 | 1.00 | 1.00 | 3 |
| 31 | 1.00 | 0.67 | 0.80 | 3 |
| 32 | 1.00 | 1.00 | 1.00 | 3 |
| 33 | 1.00 | 1.00 | 1.00 | 3 |
| 34 | 1.00 | 0.67 | 0.80 | 3 |
| 35 | 1.00 | 1.00 | 1.00 | 3 |
| 36 | 1.00 | 1.00 | 1.00 | 3 |
| 37 | 1.00 | 1.00 | 1.00 | 3 |
| 38 | 1.00 | 1.00 | 1.00 | 3 |
| 39 | 0.50 | 0.33 | 0.40 | 3 |
| accuracy | | | 0.87 | 120 |
| macro avg | 0.89 | 0.87 | 0.86 | 120 |
| weighted avg | 0.89 | 0.87 | 0.86 | 120 |

Fig. 17. *Classification Report* do Modelo Final de KNN com PCA

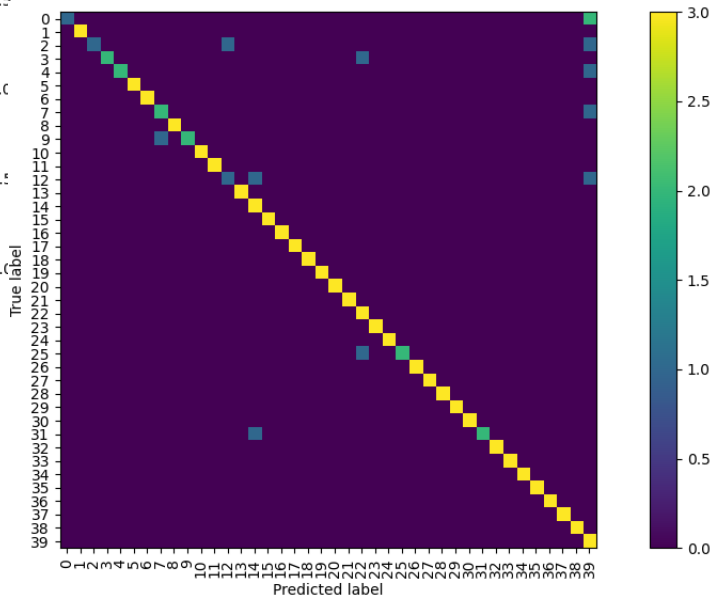


Fig. 18. Matriz de Confusão do Modelo Final de KNN sem PCA

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.33 | 0.50 | 3 |
| 1 | 1.00 | 1.00 | 1.00 | 3 |
| 2 | 0.67 | 0.67 | 0.67 | 3 |
| 3 | 1.00 | 0.67 | 0.80 | 3 |
| 4 | 0.67 | 0.67 | 0.67 | 3 |
| 5 | 1.00 | 1.00 | 1.00 | 3 |
| 6 | 1.00 | 0.67 | 0.80 | 3 |
| 7 | 0.67 | 0.67 | 0.67 | 3 |
| 8 | 1.00 | 1.00 | 1.00 | 3 |
| 9 | 1.00 | 0.67 | 0.80 | 3 |
| 10 | 1.00 | 1.00 | 1.00 | 3 |
| 11 | 1.00 | 1.00 | 1.00 | 3 |
| 12 | 0.50 | 0.33 | 0.40 | 3 |
| 13 | 1.00 | 1.00 | 1.00 | 3 |
| 14 | 0.75 | 1.00 | 0.86 | 3 |
| 15 | 1.00 | 1.00 | 1.00 | 3 |
| 16 | 1.00 | 1.00 | 1.00 | 3 |
| 17 | 1.00 | 1.00 | 1.00 | 3 |
| 18 | 1.00 | 1.00 | 1.00 | 3 |
| 19 | 1.00 | 1.00 | 1.00 | 3 |
| 20 | 0.50 | 0.67 | 0.57 | 3 |
| 21 | 1.00 | 0.67 | 0.80 | 3 |
| 22 | 0.60 | 1.00 | 0.75 | 3 |
| 23 | 0.75 | 1.00 | 0.86 | 3 |
| 24 | 0.75 | 1.00 | 0.86 | 3 |
| 25 | 1.00 | 0.67 | 0.80 | 3 |
| 26 | 1.00 | 1.00 | 1.00 | 3 |
| 27 | 1.00 | 1.00 | 1.00 | 3 |
| 28 | 1.00 | 1.00 | 1.00 | 3 |
| 29 | 0.75 | 1.00 | 0.86 | 3 |
| 30 | 1.00 | 1.00 | 1.00 | 3 |
| 31 | 1.00 | 0.67 | 0.80 | 3 |
| 32 | 1.00 | 1.00 | 1.00 | 3 |
| 33 | 1.00 | 1.00 | 1.00 | 3 |
| 34 | 1.00 | 0.67 | 0.80 | 3 |
| 35 | 1.00 | 1.00 | 1.00 | 3 |
| 36 | 1.00 | 1.00 | 1.00 | 3 |
| 37 | 1.00 | 1.00 | 1.00 | 3 |
| 38 | 0.75 | 1.00 | 0.86 | 3 |
| 39 | 0.60 | 1.00 | 0.75 | 3 |

Fig. 19. *Classification Report* do Modelo Final de KNN sem PCA

No final de cada treino, verificou-se a accuracy, loss, o gráfico de validation accuracy por epoch, de test accuracy por epoch e a matriz de confusão.

### B. Processamento dos dados

Sabendo que o dataset é constituido por 400 imagens, para os dados de input foi necessário alterar a shape do array de (400, 64, 64) para (400, 64, 64, 1) ou para (400, 1, 64, 64), dependendo do resultado de *keras.backend.imagedataformat*. Devido à semelhança no formato das imagens, não foram necessárias mais alterações aos dados de input. Para os dados de output, foi necessário separar os 400 resultados em 40 categorias através da função *keras.utils.tocategorical*.

### C. Convolutional Neural Networks

Como referido, utilizamos as capacidades de processamento de imagem que as redes convolucionais apresentam para obter os melhores resultados possíveis na nossa pesquisa. Foram desenvolvidos 3 modelos diferentes para a análise dos dados. Todos eles são compilados com o optimizer adam com learning rate de 0.001.Este optimizer é um stochastic gradient descent method, computacionalemente efficiente, pouca memory requirement e bom para problemas com grande quantidade dados e parametros, como é o nosso caso. Outro aspeto importante referir durante a compilação é a loss function escolhida. Visto que reconhecimento facial é um problema em que só um resultado pode estar correto, isto é, as fotografias presentes no dataset só pode pertencer a uma pessoa, foi escolhido categorical crossentropy como loss function. De seguida vão ser descritos os três modelos desenvolvidos. O primeiro é constituído por:

- Input Convolutional + MaxPooling Layer
  - Filters = 64
  - Kernel Size = (3,3)
  - Activation Function = relu
  - Pool Size = (2,2)
  - Input Shape = samples.shape
- Convolutional + MaxPooling Layer
  - Filters = 64
  - Kernel Size = (3,3)
  - Activation Function = relu
  - Pool Size = (2,2)
- Output Dense Layer
  - Output shape = 40
  - Activation Function = softmax

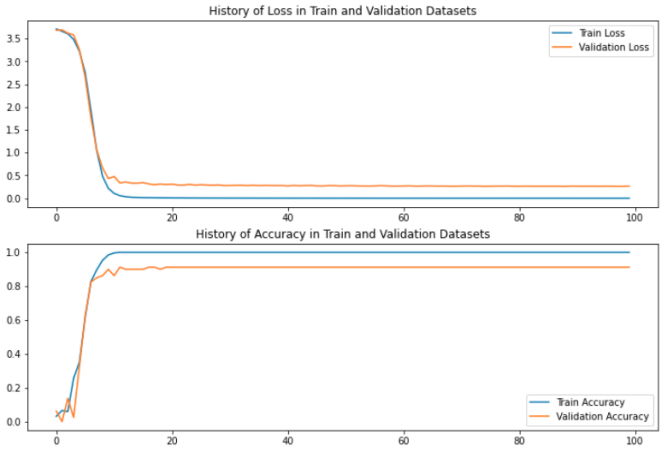


Fig. 20. Accuracy e Loss do modelo 1

Test Loss = 0.1241
Test Accuracy = 0.9624
Train Accuracy = 1.0000

Os resultados apresentados mostram que apesar do modelo ser simples apresenta uma boa accuracy, no entanto é evidente a existência de algum *overfit*. Para combater isto, no modelo a seguir foi introduzido *Dropout*.

O segundo modelo é constituído por:

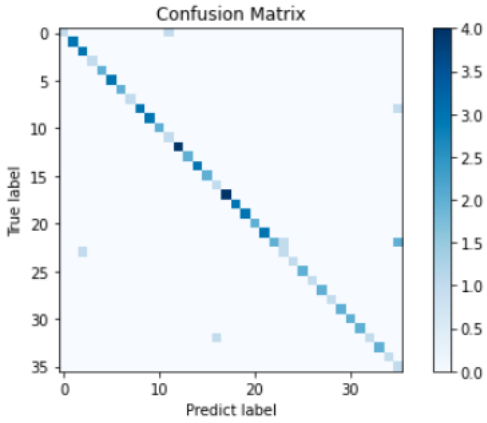


Fig. 21. Matriz de confusão do modelo 1

- Input Convolutional + MaxPooling Layer
  - Filters = 64
  - Kernel Size = (3,3)
  - Activation Function = relu
  - Pool Size = (2,2)
  - Input Shape = samples.shape
  - Dropout = 50
- Convolutional + MaxPooling Layer
  - Filters = 64
  - Kernel Size = (3,3)
  - Activation Function = relu
  - Pool Size = (2,2)
  - Dropout = 50
- Output Dense Layer
  - Output shape = 40
  - Activation Function = softmax

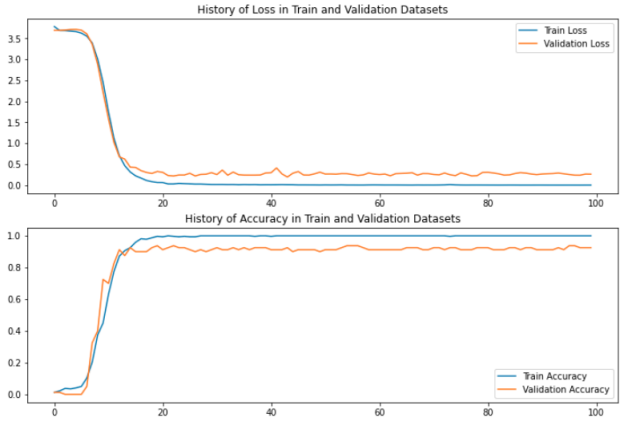


Fig. 22. Accuracy e Loss do modelo 2

Test Loss = 0.0683
Test Accuracy = 0.9624
Train Accuracy = 0.9969

Os resultados são semelhantes ao do modulo apresentado anteriormente, mas pela análise do gráfico verifica-se que o

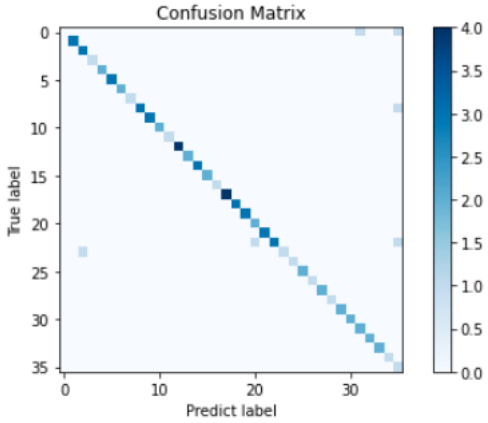


Fig. 23. Matriz de confusão do modelo 2

dropout não é tão acentuado como o do primeiro modelo, no entanto continua a existir. Isto deve-se principalmente ao facto do dataset só ter 400 imagens. Para o modelo final, foi introduzido mais dense e convolutional layers com o objetivo de aumentar a accuracy do modelo. O ultimo modelo é constituído por:

- Input Convoltional Layer
  - Filters = 32
  - Kernel Size = (3,3)
  - Activation Function = relu
  - Input Shape = samples.shape
- 3 Convolutional Layer + Maxpooling com Dropout
  - Filters = 64
  - Kernel Size = (3,3)
  - Pool Size = (2,2)
  - Activation Function = relu
  - Dropout = 50
- Dense Layer
  - Output shape = 120
  - Activation Function = relu
- Output Dense Layer
  - Output shape = 40
  - Activation Function = softmax

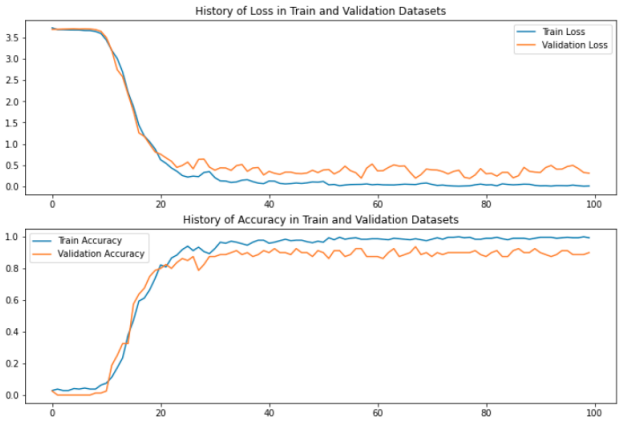


Fig. 24. Accuracy e Loss do modelo 3

Test Loss = 0.3120
Test Accuracy = 0.9589
Train Accuracy = 0.9937

Em termos de overfit, está semelhante ao do modelo anterior, no entanto os valores de accuracy sofreram um pequeno decréscimo com o aumento da complexidade adicional do modelo 3.

## VII. Conclusão

### A. Comparação de Resultados

Tendo em conta que os modelos criados foram baseados nos papers referidos [2] [3] [4], torna-se interessante comparar os resultados obtidos de maneira a perceber o que foi feito.

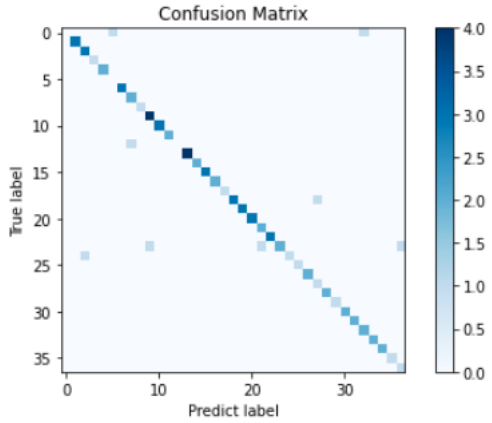


Fig. 25. Matriz de confusão do modelo 3

Nos papers, de maneira geral não é referida a percentagem de dados usados para cada fase dos modelos. Isto está relacionado com a forma como o trabalho nestes foi abordado, pelo que comparações literais não seriam muito "úteis".

Contudo, no paper [2] são referidas as percentagens de accuraccy de 66.67% com a aplicação de PCA "apenas", e 93.33% com redes neurais.

Num outro paper [4], são reveladas as accuracies obtidas com o mesmo dataset, aplicando KNN. Neste relatório em questão, são apresentados então vários valores de accuracy em função do número de vizinhos. Neste sentido, é interessante observar que comparativamente ao nosso modelo, para cada K nos encontramos apenas um ou dois pontos percentuais abaixo. Como justificação a tal facto, é importante referir que o modelo deles foi implementado em C, o que confere um maior grau de controlo sobre o modelo a ser criado. Finalmente, num outro paper [12], não referido até agora, uma vez que ao fazer uso de técnicas mais complexas, não pareceu sensato aplicar, são apresentados vários valores de accuracy obtidos com as mais diversas variações nas técnicas aplicadas. Uma vez que faz uso do mesmo dataset, é interessante referir que obtiveram como valores máximos de accuracy cerca de 98%. Embora no nosso projeto, o valor máximo atingido tenha rondado os 96%, pensa-se que a diferença está na técnica de pre-processamento aplicada - PCA -, em prol da usada no paper em questão - LDA, que foi inclusive referida no inicio do relatório.

### *B. Conclusões Finais e Melhorias*

Tendo em conta os resultados obtidos, estamos bastante satisfeitos com estes. Embora tendo ficado ligeiramente abaixo dos valores obtidos nos papers enunciados, cremos que não ficamos longe se os atingir. Como referido no inicio, a razão para a aplicação de PCA baseou-se no facto de nos deparamos com o dataset bastante pequeno. Um potencial ponto de melhoria seria então a aplicação de LDA, como referido no paper [2] - pois tem em atenção as classes ao contrário de PCA - e trabalhado no paper [12]. Adicionalmente, favorecer modelos mais simples parece promissor, pois como se observou na terceira NN, o aumento da complexidade acabou por piorar os resultados. Finalmente, um fator crucial na realização deste trabalho foi o tempo disponível para elaboração do mesmo, na medida em que havendo mais tempo para investir no mesmo, eventualmente melhores resultados seriam alcancáveis.

## REFERENCES

[1] http://cam-orl.co.uk/facedatabase.html
[2] https://arxiv.org/ftp/arxiv/papers/1210/1210.1916.pdf
[3] http://ijarcet.org/wp-content/uploads/IJARCET-VOL-1-ISSUE-9-135-139.pdf
[4] http://download.garuda.ristekdikti.go.id/article.php?article=315337&val=5535&title=Implementation%20of%20K-Nearest%20Neightbors%20Face%20Recognition%20on%20Low-power%20Processor
[5] https://stackoverflow.com/questions/53802098/how-to-choose-the-number-of-components-pca-scikitliear
[6] https://www.quora.com/What-is-the-best-way-to-choose-the-number-of-components-in-PCA-during-dimensionality-reduction
[7] https://www.mikulskibartosz.name/pca-how-to-choose-the-number-of-components/
[8] https://en.wikipedia.org/wiki/Feature_scaling#Rescaling_(min-max_normalization)
[9] https://www.ibm.com/support/knowledgecenter/SSLVMB_23.0.0/spss/tutorials/fac_cars_tve.html
[10] https://stackoverflow.com/a/30779817
[11] https://github.com/hieunguyen1053/keras_Facical_Recognization
[12] https://www.researchgate.net/publication/282599102_Regularized_generalized_Eigen-Decomposition_with_applications_to_sparse_supervised_feature_extraction_and_sparse