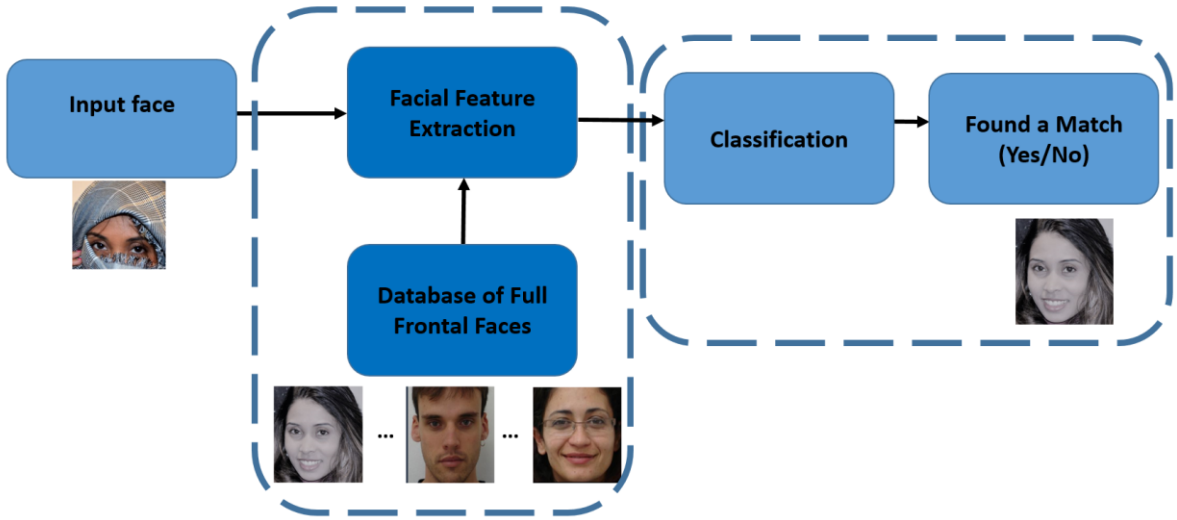
MeeKaaku - Method of Deep Face Matching and Recognition using Cosine Similarity Measure

Authors: Hassan Ugail and Ali Elmahmudi
Affiliation: Centre for Visual Computing, University of Bradford, Bradford, BD7 1DP, UK
Contact email: h.ugail@bradford.ac.uk
Keywords: Face Recognition, Deep Learning, Cosine Similarity

Full details of the method available at: Ali Elmahmudi and Hassan Ugail, Deep Face Recognition using Imperfect Facial Data, Future Generation Computer Systems, 99, 213-225, 2019.
https://www.sciencedirect.com/science/article/pii/S0167739X18331133

GRAPHICAL ABSTRACT



Overview of the method of MeeKaaku for face recognition.

ABSTRACT
We describe a method to determine the similarity between facial images through deep learning. We use the pre-trained VGG-Face model to extract deep facial features for the training phase. A cosine similarity based metric is utilised for facial image classification. We refer to this method as MeeKaaku – meaning who is this in Dhivehi (the native language of the Maldives).

- Deep facial features are extracted by means of the pre-trained VGG-Face model.
- Cosine similarity is utilised for feature classification and determining the similarity scores.
- MATLAB code (given in the appendix) can be used for testing and experimentation.

Method details

A Convolutional Neural Network (CNN), also known as Deep Learning, is a supervised machine learning technique that can extract “deep” knowledge from a dataset through rigorous example based training. To this end, CNNs have been successfully applied in many areas of objection recognition. And face recognition is one such area.

There are several ways to successfully develop a CNN. They are, training a network from scratch, fine tuning a pre-existing network and using off the shelf features from a pre-trained network. These latter two techniques are generally known as transfer learning. In general, it is sufficiently easier to utilise transfer learning as the method of choice. Here we use a transfer learning approach by means of utilising a pre-trained model known as the VGG-Face model [1]. The model was trained on a dataset containing 2.6 million faces of over 2600 individual identities. VGG-Face comprises of 38 layers. From the various experiments we have undertaken, we have found that features from the layer 34 appear provide the most favourable classification results.

Algorithm 1, shown below, provides details of how a number of face classes – belonging to a number of individuals – can be trained using the VGG-F model. In the case of small face samples belonging to a given class, data augmentation through linear transformation on the original images can also be performed [2].

Algorithm 1: Feature extraction from the face dataset

Input: Training set M, with m classes

n_j = number of images in a given class

for *i=1 to m* **do**

  for *j=1 to* n_j **do**

    $im \rightarrow$ read an image;

    $im \rightarrow$ resize(image);

    $im \rightarrow$ normalize(image);

    $imfeatures \rightarrow$ ExtractFeatures(CNNs(im));

  end

end

A cosine similarity (CS) is a measure between two non-zero vectors. It uses the inner product space to measure the cosine of the angle between those two vectors. The Euclidean dot product formula as in Equation 1 can be used to compute the cosine similarity such that,

$$a \,.\, b = ||a||\ ||b||\ cos\ \theta, \tag{1}$$

where a and b are two vectors and θ is an angle between them.

By using the magnitude or length, which is the same as the Euclidean norm or the Euclidean length of the vector $x = [x_1, x_2, x_3, \ldots, x_n]$ as in Equation 2, the similarity S is computed using the formulation given in Equation 3.

$$\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}, \tag{2}$$

$$\begin{aligned} S = \cos\theta &= \frac{A \,.\, B}{\|\ A\ \|\ \|\ B\ \|}, \\ &= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \end{aligned} \tag{3}$$

where, A and B are two vectors.

For classification, we compute the CS to find the minimum "distance" between the test image $test_{im}$ and training images $training_{im}^{n}$ by using Equations 4 and 5 such that,

$$CS_{min} = min\left(dist(test_{im}, training_{im}^{n})\right), \tag{4}$$

where, im is an image number and n is a total images in the training set and,

$$CS_{dist}(test_{im}, training_{im}^{n}) = \frac{\sum_{j=1}^{m} training_{jm}^{i}\ test_{im}}{\sqrt{\sum_{j=1}^{m} training_{jm}^{i^2}}\sqrt{\sum_{j=1}^{m} test_{im}^{2}}}, \tag{5}$$

where, m is the length of the vector.

The CS measure can be utilised to compute the distance between a probe face and the faces in individual classes. In the case of computing the similarity measure between two given faces, features from the layer 34 on VGG-Face can be extracted for each of the face and then the CS measure between them can be computed. Appendix A provides the full MATLAB code for the method described here.

Acknowledgements This work was supported in part by the European Union's Horizon 2020 Programme H2020-MSCA-RISE-2017, under the project PDE-GIR with grant number 778035.

Appendix A

```
****************** MeeKaaku Face Recognition ****************************
% NB: Any work arising from the use of this code must provide due credit and cite the following work of the authors.
% Ali Elmahmudi and Hassan Ugail, Deep face recognition using imperfect facial data, Future Generation Computer
%Systems, 99, 213-225, 2019.

% MeeKaaku Face Recognition Code

% Install and compile MatConvNet by using the link below
untar('http://www.vlfeat.org/matconvnet/download/matconvnet-1.0-beta25.tar.gz') ;
cd matconvnet-1.0-beta25
run matlab/vl_compilenn ;
% Setup MatConvNet
run matlab/vl_setupnn ;

% Download pre-trained model from this website
% http://www.vlfeat.org/matconvnet/pretrained/

% Load the pre-trained model
net = load('vgg-face.mat') ;
net = vl_simplenn_tidy(net) ;
no_items=34;

% Select a training path
TrainingPath=uigetdir('..\','Select path for the Training set');
TrainingPath =strcat(TrainingPath,'\');
[X,Y,~]=FeaturesExtraction(TrainingPath,net,no_items);

% Select a testing path
TestPath=uigetdir('..\','Select path for the Testing set');
TestPath =strcat(TestPath,'\');
[X_test,Y_test,~]=FeaturesExtraction(TestPath,net,no_items);

% Training the classifiers
h = busydlg('Please wait while training all classifiers...','Training phase');
 try

    LinearClassifier = fitcecoc(X,Y);

    t = templateSVM('Standardize',1,'KernelFunction','polynomial', 'KernelScale', 'auto', 'PolynomialOrder', 1.5);
    polynomialClassifier = fitcecoc(X, Y,'Learners',t,'FitPosterior',1);

    t2 = templateSVM('Standardize',1,'KernelFunction','gaussian', 'KernelScale', 'auto');
    KernelClassifier = fitcecoc(X,Y,'Learners',t2);

  delete(h);
  msgbox('!!! Training process complete !!! ');
catch
  delete(h);
 end
 % End of training stage for SVMs

 % Testing phase

 no_items=size(X_test,1);
 cou_rbf=0; cou_lsvm=0;cou_cs=0;cou_psvm=0;
   rbf=0;linear=0;pol=0;cs=0;
    for cl_im=1:no_items

        %%% RBF SVMs
        tic
        [label_rbf, ~,~]=predict(KernelClassifier,X_test(cl_im,:));
        toc
        rbf=rbf+toc;
```

```
        if strcmp(label_rbf,Y_test(cl_im))==1
           cou_rbf=cou_rbf+1;
        end

        %%% Lieaner SVMs
        tic
        [label_SVML, ~,~]=predict(LinearClassifier,X_test(cl_im,:));
        toc
        linear=linear+toc;
        if strcmp(label_SVML,Y_test(cl_im))==1
            cou_lsvm=cou_lsvm+1;
        end

        %%% Pol SVMs
        tic
        [label_pol, ~,~]=predict(polynomialClassifier,X_test(cl_im,:));
        toc
        pol=pol+toc;
        if strcmp(label_pol,Y_test(cl_im))==1
           cou_psvm=cou_psvm+1;
        end

        %%% CS
        tic
        cos_res=cose_s(X,Y,X_test(cl_im,:));
        toc
        cs=cs+toc;
        if strcmp(cos_res,Y_test(cl_im))==1
           cou_cs=cou_cs+1;
        end

     end
    % Results of classification
    Percentage_rbf=(cou_rbf/no_items)*100;
    Percentage_pol=(cou_psvm/no_items)*100;
    Percentage_cs=(cou_cs/no_items)*100;
    Percentage_lsvm=(cou_lsvm/noi)*100;
*********************************************************
********** Feature extraction Function ****************
function [X,Y,Z]=FeaturesExtraction(Dir,net,no)

S = dir(Dir);
c=sum([S.isdir]);
X=[];
Y={};
Z={};
N=numberofimages(Dir);
str1=strcat('Please wait...You have: ',num2str(N),' images');
h=waitbar(0,str1,'Name','Training Feature extraction');
o=0;
for i=3:c
    Dir2=strcat(Dir,S(i).name);
    imgs = dir(fullfile(Dir2, '*.jpg'));
    %%%%
    % Convert all images 2D into 1D
    for j=1:length(imgs)
        o=o+1;
       im = imread(fullfile(Dir2, imgs(j).name));
       z= imgs(j).name;
       %convert image from grayscale into rgb
       if length(size(im))<3
         im=cat(3,im,im,im);
       end
       im=single(im);
       %Resize image to 224*224
       [ro, co,~] = size(im);
```

```
      if (ro~=224)|| (co~=224)
         im=imresize(im, net.meta.normalization.imageSize(1:2));

      end

      im=im - net.meta.normalization.averageImage ;

      %Run the CNN
      res = vl_simplenn(net, im) ;
      feature = squeeze(gather(res(no).x));
      feature = feature(:);
      %add features to matrix X and their label to Y
      X = [X; feature'];
      y = S(i).name;
      Y = [Y; y];
      Z = [Z; z];
      waitbar(o/N,h);
   end
end
 delete(h);
 end
**********************************************************************
********* Cosine Similarity ***********************************************

function [predicted_result]=cose_s(c1,lab1,c2)


 for i=1:size(c2,1)
    min1 = pdist2((c1(1,:)),(c2(i,:)),'cosine');
    ind=1;
    for j=1:size(c1,1)
       D = pdist2((c1(j,:)),(c2(i,:)),'cosine');
       if D<min1
           ind=j;
           min1=D;
       end
    end
       predicted_result=lab1(ind);
  end
**********************************************************************
```

References:

[1] P. Omkar, V. Andrea, Z. Andrew, Deep face recognition, in: X. Xianghua,J. Mark, T. Gary (Eds.), Proceedings of the British Machine Vision Conference (BMVC), BMVA Press, 2015, pp. 41.1-41.12.

[2] A. Elmahmudi and H. Ugail, Experiments on deep face recognition using partial faces. in 2018 International Conference on Cyberworlds (CW). 3-5 Oct 2018, Singapore, 2016, pp. 357-362.