Shijie Song, Ding Liang, Lu Zhang

Professor Muslea

INF 552

28 Nov 2017

1.Abstract

In this project, we compared three different machine learning algorithms' (SVM, KNN, CNN) behavior when acting on two different face data sets (AT&T faces and CACD faces). The Viola-Jones algorithm would be used to pretreat the CACD dataset. PCA algorithm is an efficient way that is used to extract features from the input sample. Lastly, the training analysis and testing result were discussed based on the accuracy of the result received from each algorithm.

2.Introduction

Past, now and future, facial recognition has become an indispensable technology that helps satisfying the growing needs for UX and security. Note that Apple has launched its newest product iPhone X with the ability of unlock screen by face recognition, this technology, would occupy every corner of our future life.

In this project, we want to compare several facial recognition system with machine learning algorithms. Before doing any detailed work with the face, we would need to detect a face from a complicated picture. The Viola-Jones object detection framework which is proposed in 2001 by Paul Viola, through our research, would be a sufficient way to detect the face from a photo. However, there exist a lot of cases that man in the picture that is not facing the camera, which is, same person facing different direction might have facial features that is hard to read. In this project, we will mainly target on the frontal face recognition. Now it comes to the hardest part of this project: distinguish different face. In this part, we are planning to compare a SVM (Support Vector Machine) which is a supervised learning model, a KNN algorithm, and a CNN algorithm which is a convolutional neural network algorithm to train and classify our data. A final analysis would be conducted to compare the difference based on accuracy of applying these algorithms to the two selected datasets, ORL faces and CACD faces.

3.Project and Experiment

3.1Data-Set

3.1.1.ORL

In this database,there are ten different imges of each of 40 distinct subjects. For some subjects, the images were taken at diffent times, varing the lighting, facial expressions(include open/closed eyes,smiling/not smiling) and facial details(glasses/no glasses).The files are in PGM format, and can conveniently be viewed on UNIX (TM) systems using the 'xv' program. The size of each image is 92x112 pixels, with 256 grey levels per pixel. The images are organised in 40 directories (one for each subject), which have names of the form sX, where X indicates the subject number (between 1 and 40). In each of these directories, there are ten different images of that subject, which have names of the form Y.pgm, where Y is the image number for that subject (between 1 and 10).

3.1.2.CACD2000

Cross-Age Celebrity Dataset (CACD) contains 163,446 images from 2,000 celebrities collected from the Internet. The images are collected from search engines using celebrity name and year (2004-2013) as keywords.We can therefore estimate the ages of the celebrities on the images by simply subtract the birth year.

3.2 Data pre-treat

The ORL dataset is frontal.But as for the CACD,the original data is rgb, three dimensions, so we need to do modify the data set.

3.2.1face detect

Face detection is the first part in the face recognition. Usually, the data or the picture can be achieved are consisted with not only the human face, other redundant elements such as cloths and hair that have nothing to contribute to the analysis are provided at the same time. Getting rid of these elements is necessary for the face recognition procedure to be more accurate. Among all, the viola jones algorithm is most commonly used to achieve this goal.

Viola-Jones algorithm is proposed on CVPR in 2001, known for its fast and efficient detection and are still in use today. The algorithm is well designed for frontal faces but not side faces. This algorithm can be summed up in four steps: feature selection, feature evaluation, feature learning with AdaBoost and cascading classifiers. According to the Haar-like features, all human faces share common looking especially in photos. For example, the eye area would be darker than the cheek area since cheek area is relatively higher. Also, nose area is much brighter than the face area around it. Therefore, there are laws for the distribution of the eyes, mouth, nose, eyebrows of all frontal faces. Based on the regulation, the Haar-like feature can calculate the location of pixels to determine the feature of the faces. The implementation of the algorithm is presented next.

```python
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('2.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    new_img = img[y:y+h, x:x+w]
    #img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),3)
    #roi_gray = gray[y:y+h, x:x+w]
    #roi_color = img[y:y+h, x:x+w]
    # eyes = eye_cascade.detectMultiScale(roi_gray)
    # for (ex,ey,ew,eh) in eyes:
    #     cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imwrite('output.png',new_img)
cv2.imshow('new_img',new_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

At first, we use the python to pre-treat data,as we said in mit-term, also, we use the matlab to crop.

```matlab
        FaceDetector=buildDetector();
        [bbox,~,~,~] =    detectFaceParts(FaceDetector,img1,2);
        recFace.x=bbox(1,1);
        recFace.y=bbox(1,2);
        recFace.width=120;
        recFace.height=120;
        ptFaceCenter.x    = recFace.x + recFace.width / 2;
        ptFaceCenter.y    = recFace.y + recFace.height / 2;
        recFace.x         = ptFaceCenter.x - recFace.width * 0.4;
        recFace.y         = ptFaceCenter.y - recFace.height * 0.35;
        recFace.width     = recFace.width * 0.8 ;
        recFace.height    = recFace.height * 0.8 ;
        mFaceResult       = uint8(imcrop(img1,
[recFace.x,recFace.y,recFace.width,recFace.height]));
        name=strcat(num2str(j),'.jpg');
        path=strcat('new/s',num2str(i));
        pat=strcat(path,'/');
        out=rgb2gray(mFaceResult);
        imwrite(out,[pat,name]);
        end
```

We need to nomalize the the size of the matrix,so we use the following code

```matlab
        recFace.width=120;
```

```
    recFace.height=120;
```

Also, when we read the testclass we need to lable the realclass

```
for j=1:m
    facepath=cachepath;
    facepath=strcat(facepath,num2str(j));
  %input the testing samples

    facepath=strcat(facepath,'.jpg');
    img1=imread(facepath);
    realclass((i-1)*m+j)=i;
```

In this project, one of our data set has the problem of overdosed contents. For example, in this picture (Fig 2 left), the suit, the bow tie and the fashion hair cut will only result in the bias in the final prediction since the hair and the cloth would be the most instable factor in the photo for they change a lot. Below is the result of getting rid of the useless contents.

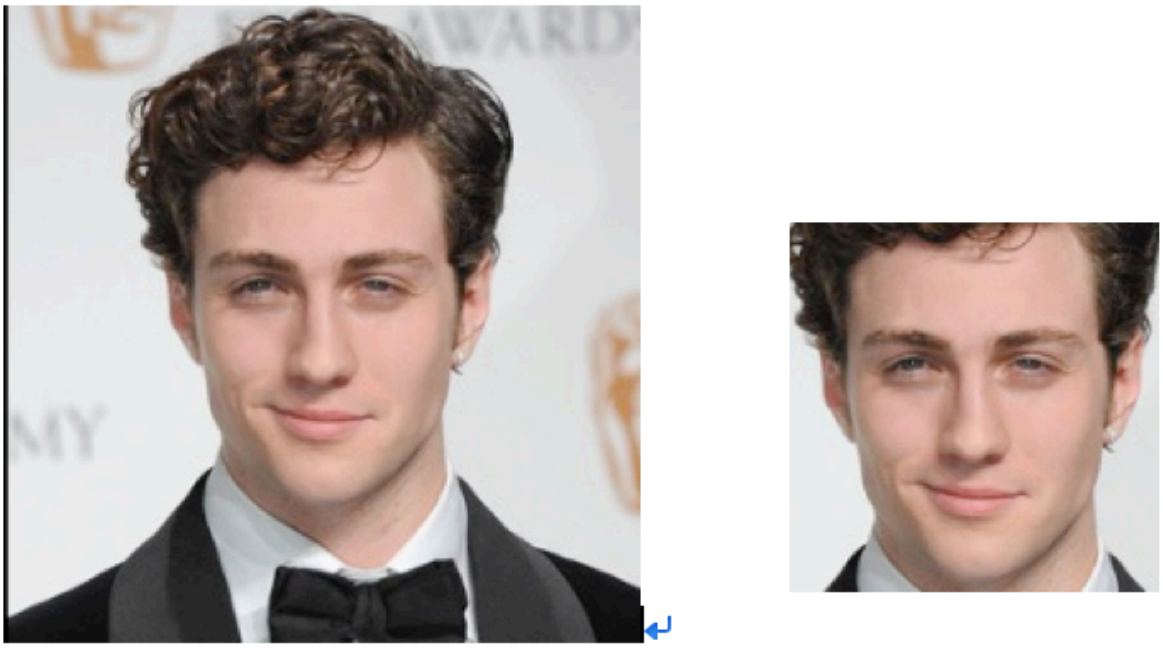


Fig 1: Photo before (left) and after (Right) Viola-jones detection

3.2.2 PCA to extract the features

We read the images from the detection part. When we use the data to train svm we need to extract some principal features. In our project, as for the CACD dataset, the original matrix is 97*97, so we use pac to

change this matrix to a smaller matrix.

```
function [ pcaA,V] = fastPCA( A,k,mA)
%reduce the matrix A to k dimensions
m=size(A,1);
Z1=(A-repmat(mA,m,1));
Z=double(Z1);
T=Z*Z';
[V,D]=eigs(T,k);
V=Z'*V;
for i=1:k
    l=norm(V(:,i));
    V(:,i)=V(:,i)/l;
end
pcaA=Z*V;
end
```

3.2.3scaling the data

```
function [ scaledface] = scaling( faceMat,lowvec,upvec )
%use the max and min to normalization the data
upnew=1;
lownew=-1;
[m,n]=size(faceMat);
scaledface=zeros(m,n);
for i=1:m
    scaledface(i,:)=lownew+(faceMat(i,:)-lowvec)./(upvec-lowvec)*(upnew-
lownew);
end
end
```

3.3 SVM algorithms

Suport Vactor Machines aer supervised learning models with associated learning algorithms that can be used for classificaion and regression analysis. Face Recognition is a problem of classification. So we use the svm to do some analysis.

3.3.1 Kernal function

In machine learning, the (Gaussian) radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

$$K_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\left[-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right]$$

where γ is a parameter that sets the "spread" of the kernel.

```
function [ K ] = kfun_rbf(u,v,gamma)
%RBF
m1=size(u,1);
m2=size(v,1);
K=zeros(m1,m2);
for i=1:m1
    for j=1:m2
        K(i,j)=exp(-gamma*norm(u(i,:)-v(j,:))^2);
    end
end
end
```

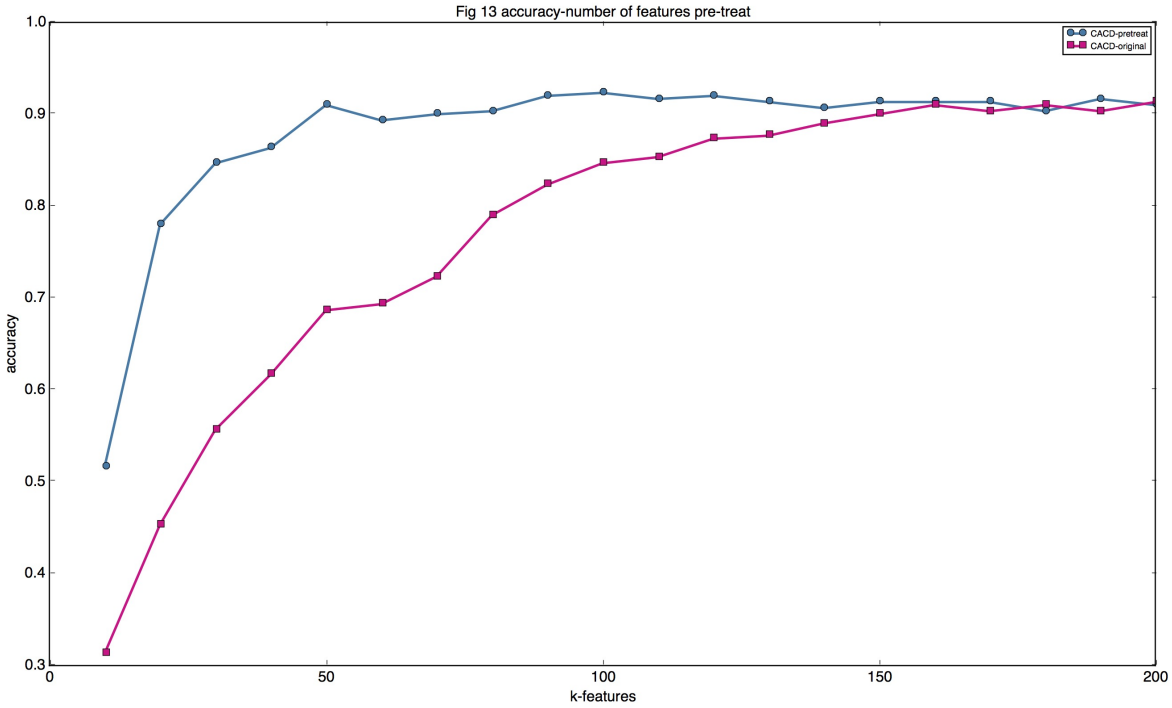
3.3.2 SVM train

```
function [ multiSVMstruct ] =multiSVMtrain( traindata,nclass,gamma,c)
global m;
m=70;
for i=1:nclass-1
    for j=i+1:nclass
        X=[traindata(m*(i-1)+1:m*i,:);traindata(m*(j-1)+1:m*j,:)];
        Y=[ones(m,1);zeros(m,1)];
        multiSVMstruct{i}
{j}=svmtrain(X,Y,'Kernel_Function',@(X,Y) kfun_rbf(X,Y,gamma),'boxconstrai
nt',c);
    end
end
end
```

3.3.3 SVM classify

```
function [ class] = multiSVM(testFace,multiSVMstruct,nclass)
m=size(testFace,1);
voting=zeros(m,nclass);
for i=1:nclass-1
    for j=i+1:nclass
        class=svmclassify(multiSVMstruct{i}{j},testFace);
        voting(:,i)=voting(:,i)+(class==1);
        voting(:,j)=voting(:,j)+(class==0);
    end
end
[~,class]=max(voting,[],2);
end
```

3.3.4 SVM Result Analysis

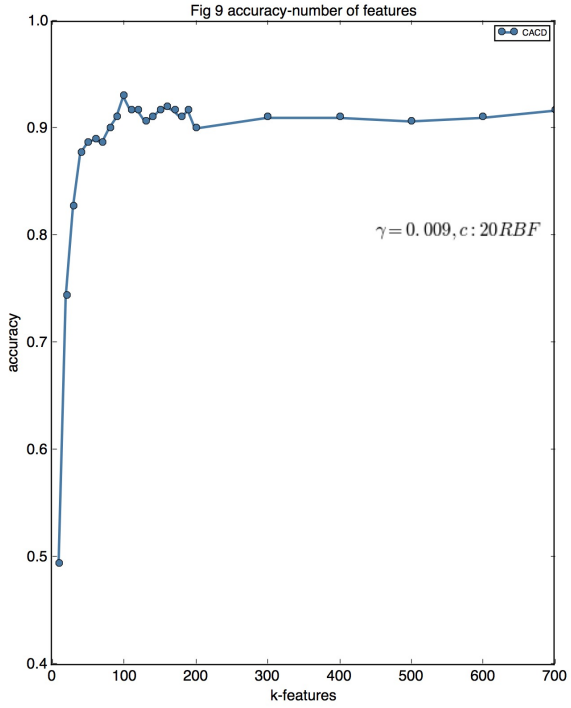


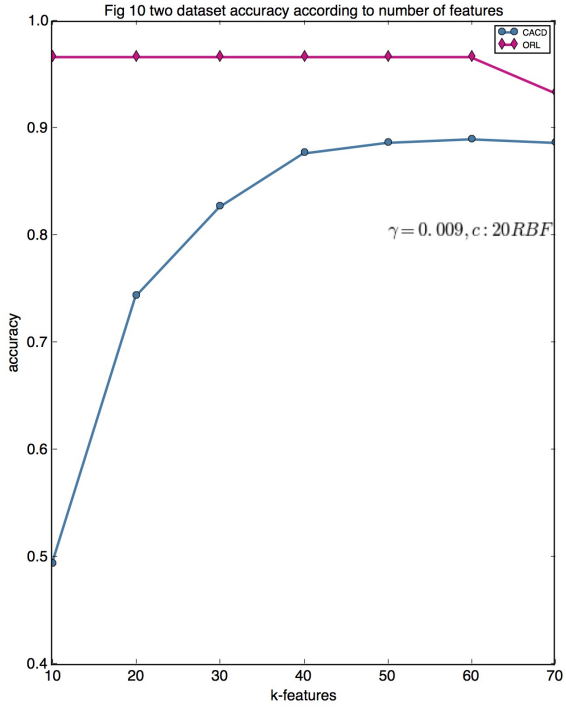
We use the original CACD data set ,which is 250*250*3,including RGB and the pre-treated data in this part. The original data is three dimensions and we transform it to the two dimensions.

As we can see from the Fig 13,We can find that after pre-treat the data, the speed of the algorithm to converge becomes much better, compared to the original.

Reason 1.We think pre-treated data has less noises,and pixels, so we can just use the principal features to train our model. 2.When we do the PCA,ignoring the input, we use same number to implement PCA, so the influence of PCA on the original is less significant.

Test Method Change the number of features, compare the result.





3.4accuracy

Because we use the label data to test, so we can compare the result of svmclassify with the initial label to calculate wether classification is right.

```
accuracy(ll)=sum(class==realclass)/length(class);
```

3.5KNN Alogorithms

Another common-used way of face recognition algorithm is KNN (K-Nearest Neighbor), a non-parametric method used for classification first proposed by Cover and Hart. The traditional KNN algorithm consists of several parameters including: training dataset, testing dataset, class label, user-defined constant k, distance metric and the classification strategy. Each of the parameter design will be discussed in the flowing paragraphs.

```
x=zeros(20,1);
y=zeros(20,1);
for i = 1:20
    class = knnclassify(scaledtestface,scaledface,trainclass,i,'cosine','consensus');
    accuracy(i)=sum(class==realclass)/length(class);
    x(i) = i;
    y(i) = accuracy(i);
end
```

Training and testing dataset is divided by fraction from the entire dataset, for example, if 7 of a person's face is used for training and the rest 3 faces would be used as testing. The input dataset, as described earlier, is an m*n matrix where m stands for the total number of sample and n describes the feature of each sample. Meanwhile, the training example is distracted from all samples and assigned with a class label. To make the class type easy to modify and use, the realclass (Fig 1) of size m*1 where m is the training sample size is assigned to the training set when the training set is being segmented. Several constant K number is used to test the smoothness and accuracy of the algorithm, where K stands for the number of closest training examples in the feature space. In this project, K is designed from 1 to 10 to compare the error of the classification results.

The most common ways of measure the distance between kernel and each sample feature are Euclidean distance, cityblock (the sum of absolute differences), cosine, correlation (one minus the sample correlation between points) and Hamming. However, not all methods would be accurate enough to fit in the dataset, for example, the hamming method is the percentage of bits that differ and this method would only work for binary data. Therefore in this project, two methods are chosen and brought in for compare. The Euclidean distance is picked as the default distance calculator where the distance of two samples in our m*n field is calculated by:

$$D(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad .$$

Another way chosen to apply to the system is the cosine similarity, the measure of difference of angle between two samples which is:

$$D(p,q) = 1 - \cos(\theta) = 1 - \frac{p \cdot q}{||p||||q||} \quad .$$

When the sample grows larger or the features tends to be similar, an interesting problem is beginning to head above water: the two samples may have very similar or exactly the same distance to the kernel. This brings us to the last parameter of the KNN algorithm. There are three ways commonly supported to solve this problem: find the nearest point tie-break, random point and consensus rule. Since all human shares some same face features, the distance of two different samples tends to be very similar and we will only discuss the nearest point method and the random chosen in this case. Nearest point is mostly common used, the distance of all samples are stored into a vector and sorted, and the first K elements of the vector are returned. Random point, as its name implies, if there are 3 points comes into the field but we only want two, then two of these points would be chosen randomly. This may leads to a lot of problem, because these two points may belongs to different classes and this would lead to a huge bias in some case.

Using the controlled variable method, we first immobilized everything else to compare the training and testing data set fraction. The result of taking fraction of training to testing as 8:2, 7:3 and 5:5 is put into a form below (Fig 2).

	A	B	C	D	E	F
1	K	Trainning 9	Trainning 8	Trainning 7	Trainning 6	Trainning 5
2	1	0.925	0.9375	0.958333333	0.90625	0.815
3	2	0.925	0.9375	0.958333333	0.90625	0.815
4	3	0.875	0.9375	0.95	0.875	0.815
5	4	0.9	0.925	0.933333333	0.88125	0.81
6	5	0.825	0.8875	0.908333333	0.84375	0.775
7	6	0.825	0.9	0.908333333	0.83125	0.765
8	7	0.825	0.8625	0.9	0.80625	0.76
9	8	0.825	0.85	0.875	0.8125	0.745
10	9	0.825	0.85	0.858333333	0.81875	0.745
11	10	0.85	0.8375	0.858333333	0.7875	0.72
12						

Fig 2: Different fraction of the test and training number

From the result, it is not hard to see the fraction of 7:3 tends to have the most accurate result. The reason we guess which leads to this result is the relatively small data set size since we have only 10 pictures of a single person, a small test set would results in high inaccuracy because each false predict would be a large contribution factor in the overall accuracy. Therefore, 7:3 tends to be the most reasonable choice for the dataset. And the rest analysis and prediction would be implemented based on this result.

The next parameter we are going to distinguish is the distance metric, the two method we are going to compare here are Euclidean and cosine while keeping the input fraction as 7:3. Below is the result of these two different methods(Fig 3)；

	A	B	C	
1	K	Euclidean	Cosine	
2	1	0.958333333	0.958333333	
3	2	0.958333333	0.958333333	
4	3	0.95	0.941666667	
5	4	0.933333333	0.933333333	
6	5	0.908333333	0.908333333	
7	6	0.908333333	0.908333333	
8	7	0.9	0.883333333	
9	8	0.875	0.866666667	
10	9	0.858333333	0.85	
11	10	0.858333333	0.858333333	
12				

Fig 3: difference between two distances metric

From the result, we can tell that there is no much difference between the two distance methods. This shows that the sample is well clustered and bias between every class is relatively low. However, the Euclidean distance is still showing a little bit of advantage in the accuracy competition.

Last but not least, the final parameter that affect the result of the KNN model is tested. In this case we run three random test (Fig 4) to determine the classification of the same distance points and received the results below:

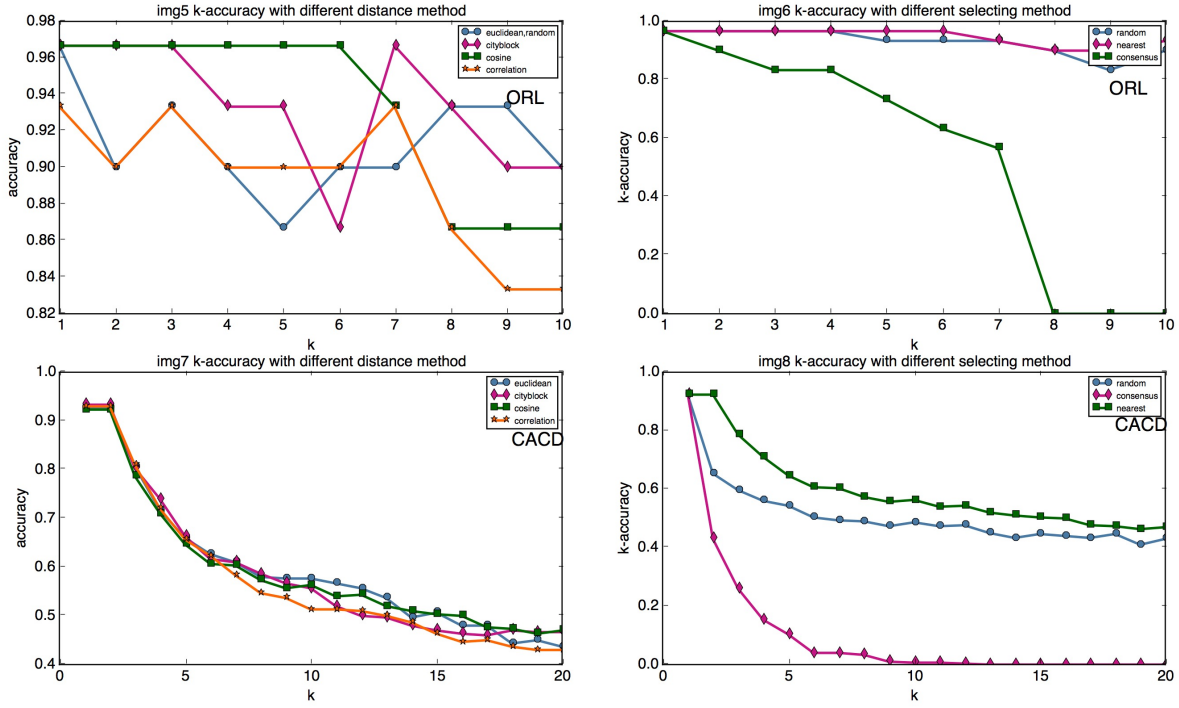
	A	B	C	D	E
1	K	Nearest	Random1	Random2	Random3
2	1	0.958333333	0.958333333	0.958333333	0.958333333
3	2	0.958333333	0.891666667	0.9	0.883333333
4	3	0.95	0.925	0.933333333	0.941666667
5	4	0.933333333	0.891666667	0.875	0.908333333
6	5	0.908333333	0.85	0.875	0.866666667
7	6	0.908333333	0.85	0.85	0.833333333
8	7	0.9	0.816666667	0.816666667	0.825
9	8	0.875	0.808333333	0.833333333	0.833333333
10	9	0.858333333	0.8	0.8	0.783333333
11	10	0.858333333	0.783333333	0.758333333	0.783333333

Fig 4: random selection of point tie-break

This test informed that the nearest point tie-break would be the most sufficient way of analyzing the dataset in this case. The random pick strategy, as mentioned earlier, may result in some unnecessary bias. For example, the field consists of 4 training samples where two of them are in the same class, two random pick under this circumstance can be the same as toss a coin and could result in some completely different result. Therefore the Nearest point tie-break stand out to be the most sufficient way to fit in this algorithms. Also, from this result, we can tell that choose K = 3 as the parameter would be the most reasonable choice because finding 3 points nearby tends to have the most stability among all the random point choices.

To conclude, the KNN algorithm, has its own advantages and disadvantages. It is easy achievable and straightforward so the algorithm can be used on determine faces. Also KNN does not need to train the data set first and this is why it is called the lazy learning algorithm, this results in an easier way of implementing but makes the algorithm relatively slow.

Finally we draw this picture to show conclude the performance KNN algorithm in our data set.

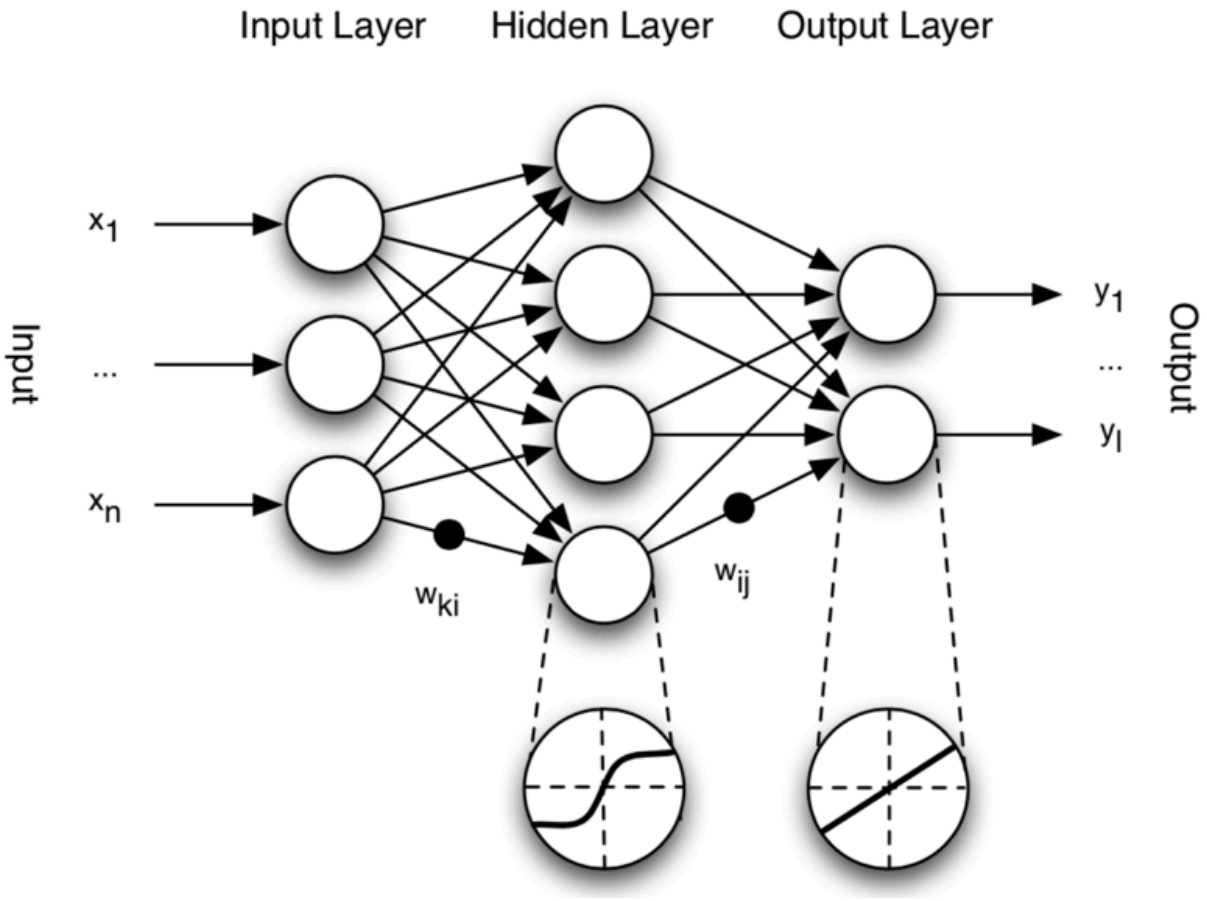


3.6CNN Algorithm

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.[1] They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics

3.6.1 Hidden Layer



In MLP (Multi-Layer Perceptron), the input is fed from the input layer to the middle layer and weighted with the connection weights.

$$a_i^{(1)} = \sum_{k=1}^{n} w_{ki}^{(1)} x_k$$

the output is the hidden layer is then calculated using a transfer function f, b is bias unit

$$y_i^{(1)} = f^{(1)}(a_i^{(1)} + b^{(1)})$$

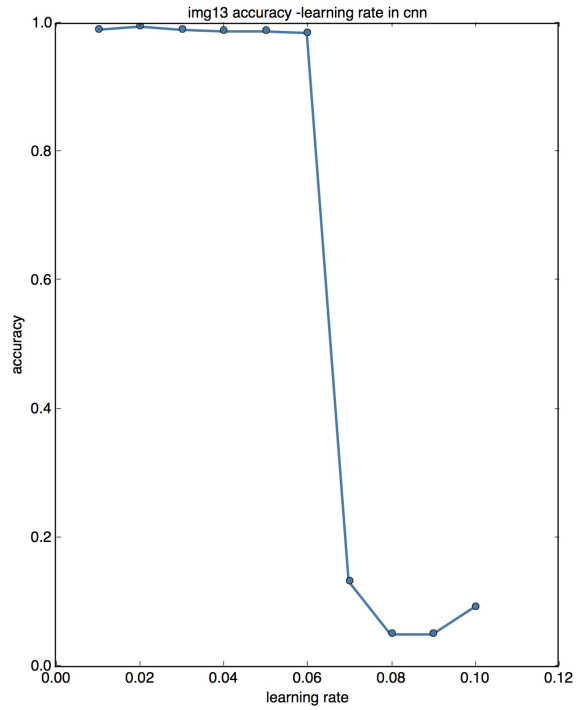
3.6.2 Result Analysis

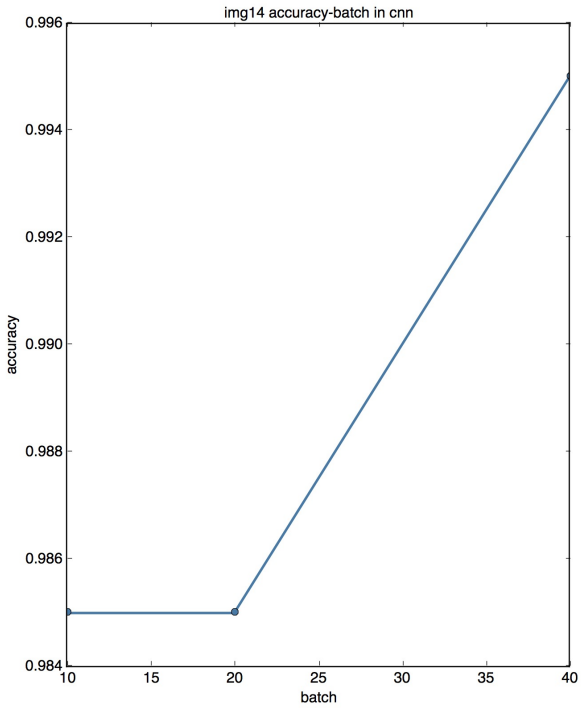
In this case, firstly, we want to make sure in which fraction of training set and testing set we can get the best accuracy. From Figure XXX, we can see that the fraction of 8:2 has a better accuracy.

training	testing	accuracy
8	2	99.50%
7	3	98.75%

Fig5 Different fraction of the test and training number

Next, we try to change the learning rate from 0.01 to 0.1. The result is shown in XXX. From the result, we can easily find that there is a distinct difference between and . The reason is that for , learning rate is too large, and we have the problem of overshooting, which means we miss the local optimum.





Batch size is also a factor needed to be considered, which includes all samples in one batch and affects the optimum degree and speed in our model. In batch learning, we compute the direction in which each parameter needs be changed and the magnitude of this change. As the result shown in Figure 13 and figure 14, the difference of batch size among 10,20 and 40 is not big, but we can still notice the fact that the accuracy of batch size of 40 is higher than others. Because the dataset we choose is relatively small, for each batch whose size is 10 or

20, we could only get images of 1 or 2 people, the optimum speed is slow compare to a large dataset.

As shown in fig 15, the accuracy tends to be better when convolutional kernels in convolutional layers is 5 and 10. The reason why accuracy with 2 and 5 convolution kernels is lower is that we do not get enough features. We know that more kernels mean more features we can extract from the data, so to some extent, it will help us to improve our accuracy. For example, it is hard for people to differentiate twins, because they look like each other. In this case, if our model could get more features from their faces, it may perform better. However, too many convolutional kernels will lead to the of complex calculation and time cost. When we use 10 and 30 convolutional kernels, the time cost is obviously higher.

convolutional kernels	accuracy
2,5	98.50%
5,10	99.50%
10,20	99.25%
10,30	99%

Fig15 different accuracies with different convolutional kernels

3.7Summery

For facial recognition, we use three algorithms, KNN, SVM and CNN to calculate the accuracy. From our research, CNN performs better than other two algorithms, because it is possible for us to extract features that we have never noticed before.