Eigenvectors & Image Recognition

Warner Alexis

2024-02-18

Eigenvectors & Image Recognition

With the attached data file, build and visualize eigenimagery that accounts for 80% of the variability. Provide full R code and discussion

The eigenvalues can tell you how much variances and diversity in a image. Eigenvalues and eigenvectors are widely used in image processing and signal processing task such as denoising, image compression and feature extraction.

we load the packages :

package $foreach $ is used to allows for iterating over elements in a collection, without the use of an explicit loop counter.

package *jpeg* is used to access and read all the image located in the folder jpeg.

package *EBImage* is used for image processing and analysis. EBImage offers tools to segment cells and extract quantitative cellular descriptors

package *doParallel* is used to provide a mechanism needed to execute foreach loops in parallel.

We created a folder to read all the photo together.

This is similar to using tensorflow. first, you create a directory to access the all images and use tensorflow keras to processing the image in which it will regenerate matrix for the images.

```
# loading the library

library(foreach)
library(jpeg)
library(EBImage)
library(doParallel)
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
# read the file in my directory

files=list.files(path='~/CUNY/DATA 605 - Computational Mathematics/Assignment/Week 4/jpg',pattern="\\.jp
```

Looking inside the Files

This step is used to create all the rater array for specific given location and adjust the pixel for every location. The function *rasterImage*() will match the cell values with the array at the given location and rearrange the images for every cell.

```
###################Set Adj. Parameters#############################
height=1200
width=2500
scale=20
plot_jpeg = function(path, add=FALSE) #initialize function
{
  require('jpeg')
  jpg = readJPEG(path, native=T) # read the file
  res = dim(jpg)[2:1] # get the resolution, [x is 2, y is 1]
  if (!add) # initialize an empty plot area if add==FALSE
    plot(1,1,xlim=c(1,res[1]),ylim=c(1,res[2]), #set the X Limits by size
         asp=1, #aspect ratio
         type='n', #don't plot
         xaxs='i',yaxs='i',#prevents expanding axis windows +6% as normal
         xaxt='n',yaxt='n',xlab='',ylab='', # no axes or labels
         bty='n') # no box around graph
  rasterImage(jpg,1,1,res[1],res[2]) #image, xleft,ybottom,xright,ytop
}
#####################################
```

Load the Data into an Array

the variable *im* is created to store the array and arrange it into every cell so a single shoes could represent by a single cell.

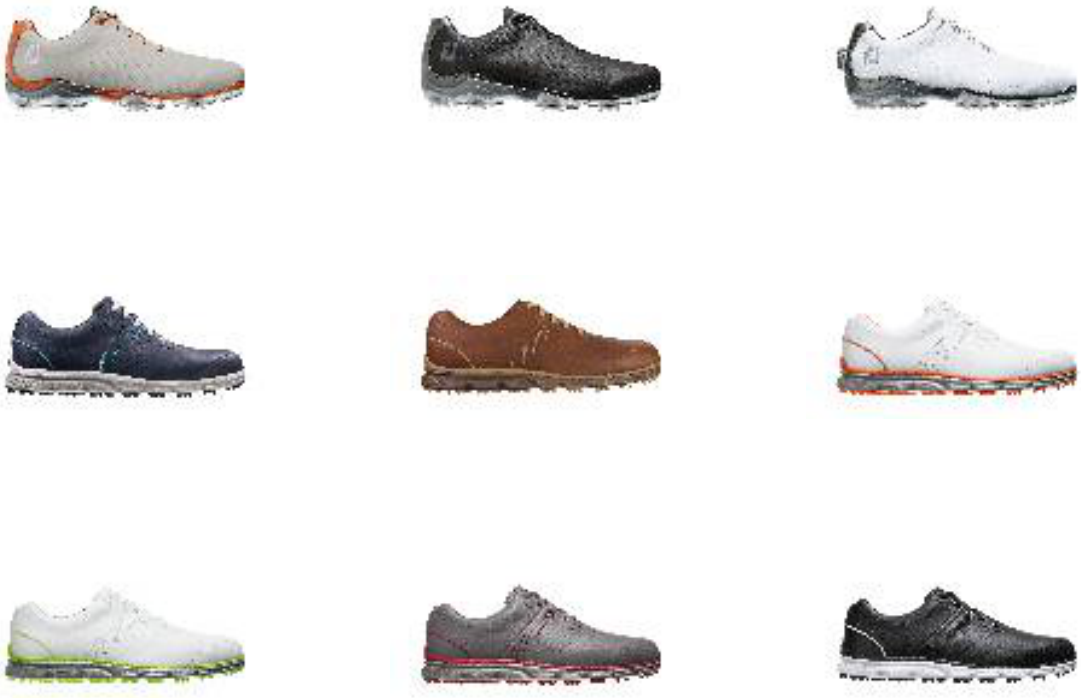
```
#initialize array with zeros.
im=array(rep(0,length(files)*height/scale*width/scale*3),
         #set dimension to N, x, y, 3 colors, 4D array)
         dim=c(length(files), height/scale, width/scale,3))

for (i in 1:length(files)){
  #define file to be read
  tmp=paste0("~/CUNY/DATA 605 - Computational Mathematics/Assignment/Week 4/jpg/", files[i])
  #read the file
  temp=EBImage::resize(readJPEG(tmp),height/scale, width/scale)
  #assign to the array
  im[i,,,]=array(temp,dim=c(1, height/scale, width/scale,3))
}
```

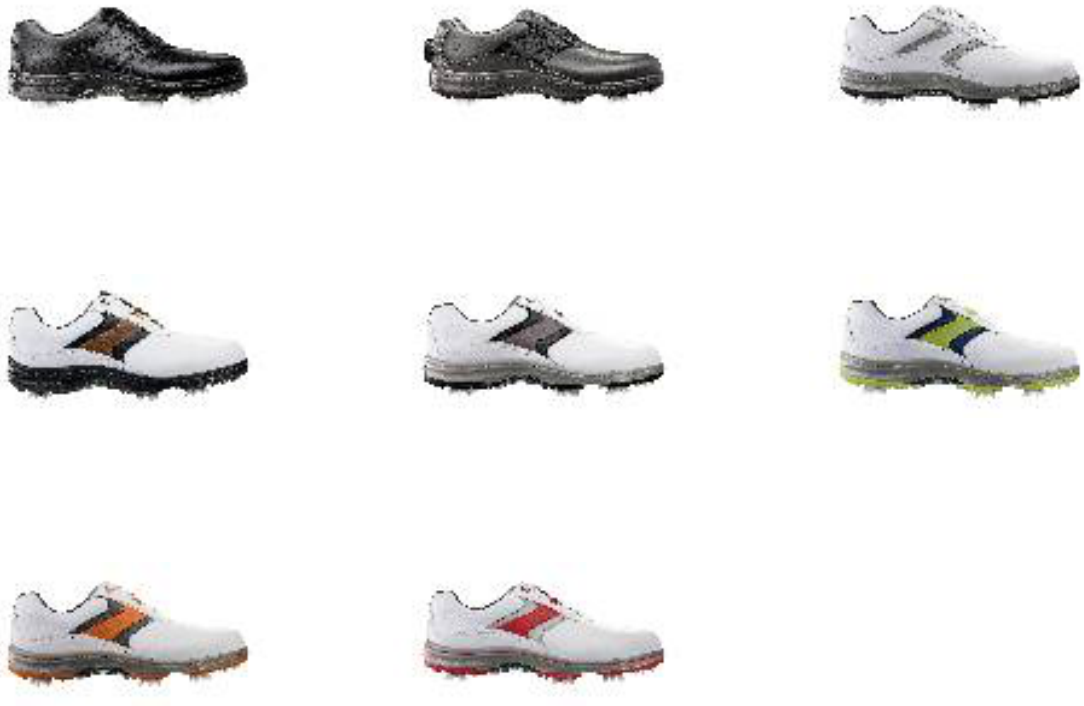
Actual Plots

This step shows all the pictures at their given location aligned by three rows.

```
####Old Shoes##################
par(mfrow=c(3,3)) #set graphics to 3 x 3 table
par(mai=c(.3,.3,.3,.3)) #set margins
```

```
for (i in 1:17){  #plot the first images only
plot_jpeg(writeJPEG(im[i,,,]))
}
```





Principal component analysis is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

```
###################Generate Variables#############################
height=1200
width=2500
scale=20
newdata=im
dim(newdata)=c(length(files),height*width*3/scale^2)
mypca=princomp(t(as.matrix(newdata)), scores=TRUE, cor=TRUE)
sum(mypca$sdev^2/sum(mypca$sdev^2)) #verify that sum of variance=1
```

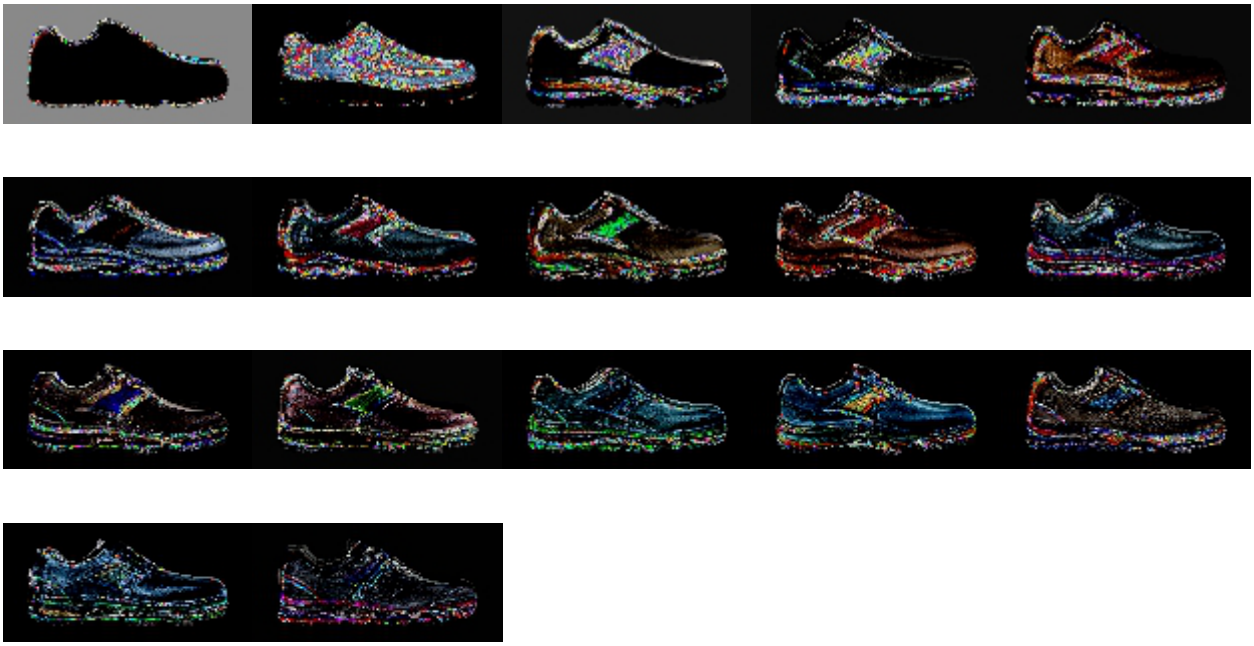
```
## [1] 1
```

```
mycomponents=mypca$sdev^2/sum(mypca$sdev^2)
sum(mycomponents[1:17]) #first 19 components account for 80% of variability
```

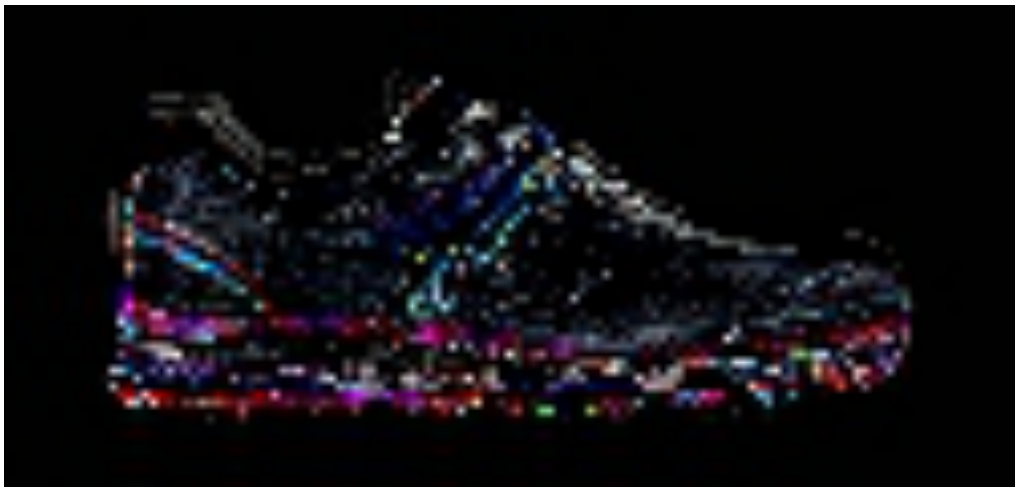
```
## [1] 1
```

Eigenshoes

```
##################Eigenshoes#####################################
mypca2=t(mypca$scores)
dim(mypca2)=c(length(files),height/scale,width/scale,3)
par(mfrow=c(5,5))
par(mai=c(.001,.001,.001,.001))
for (i in 1:length(files)){  #plot the first 81 Eigenshoes only
plot_jpeg(writeJPEG(mypca2[i,,,], quality=1,bg="white"))
}
```



```
plot_jpeg(writeJPEG(mypca2[i,,,], quality = 1, bg = "white"))
```



Referrences

1. A step-by-step explanation of principal component analysis (PCA). Built In. (n.d.). https://builtin.com/data-science/step-step-explanation-principal-component-analysis
2. Digital Image Processing Laboratory: Eigen-decomposition of ... (2023a, February 22). https://engineering.purdue.edu/~bouman/grad-labs/Eigen-Image-Analysis/pdf/lab.pdf