AI Image Detection on Noised Photos

Raynell Labayog

Company: Eigensee

Slogan: “Testing If Ai Can See”

Website: www.eigensee.gov

Phone: (334)444-3366

06 December 2024

Abstract

This paper looks into the scope of AI image readers and see how far we can take its image recognition abilities. We want to look into how much ‘noise’ we can add to a photo before it is no longer recognizable by the AI image reader. This will involve the use of Singular Value Decomposition (SVD) on a black and white image in order to reconstruct the image with varying values of k. K values dictate the number of singular values that are retained in order to reconstruct the image. Lower k values would result in a less accurate photo being outputted. We want to find out what that value is before the image is no longer detectable by the AI.

Introduction

The development of AI is still in its infancy age, where the tech is there, but still requires more work and research to be completely reliable. Services like Chatgpt or Google AI can converse and answer quite abstract questions to the user. Just recently Chatgpt added an image reader functionality with its services. This allows the user to upload an image and have the AI read what is on the image. Unfortunately, this is still being developed, and it cannot recognize individuals in pictures. We will be using Tineye's search engine for this paper instead, as it does use AI as part of its image recognition software. We want to find the limit of what this search engine can do to test the readability of its AI, we will be using 3 different images and adding various levels of noise onto them. To add noise onto these images we will need to rely on Singular Value Decomposition (SVD). When a computer is given an image it sees the data as a matrix of values. What this technique does is take the data and separate it into 3 distinct matrices as follows, with A being the original matrix.

$$A = U \cdot D \cdot V\top$$

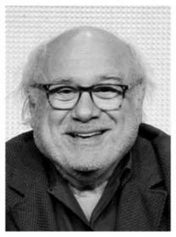
U represents the left most singular vectors, D represents the diagonal matrix, and V represents the right most singular vectors. The diagonal matrix represents the strength of the values from U and V corresponding to the reconstruction of matrix A. This shows how important the diagonal matrix is to the equation. To control how much noise is added to an image we will be manipulating the K value. The K-value represents the amount of vectors that are to be retained when approximating the image. The more K-values there are, the more accurate the picture will be. In this scenario, we want to find the minimum k-value before the photo is no longer identifiable by the AI. We want to do this because this is a way to identify the strength of these image detecting software. Benchmarking AI detecting software is great in the development of improved detecting software. One application I can see this helping improve would be the ability to identify individuals from grainy or corrupted footage.

Data and Methods

To create the sample photos to feed into the search engine we must perform SVD on the individual photos. When we perform SVD on the pictures it creates 3 different matrices, a U matrix, a V matrix and a D matrix. The D matrix is important to have since the length of the matrix dictates the values of K. This is because the D matrix indicates the importance for each pair of U and V. The larger the value is, the more important it is to the image, where as the smaller values are minute details. The photos we will be using in this paper are of Jack Black, Danny Devito, and Michelle Obama. We will also be using black and white photos done through greyscale as this reduces the amount of dimensions in the photo.

Original Grayscale Photo

Grayscale Photo



Original Grayscale Photo

Once we run the photo through SVD we want to identify the length of the D matrix and that will be the K-value for the photo. Each photo we use will have different a different k-value since they are at different dimensions. Devito's picture is 300x409, Black's picture is 2760x2760, and Obama's picture is 1800x1800.

```
svd_photo <- svd(photo)

svd_photo

U = svd_photo$u

D = diag(svd_photo$d)  # Convert singular values into a diagonal matrix

V = svd_photo$v

ktotal = length(svd_photo$d) # find total K values

ktotal

# [1] 2760
```

This prepares the k-value to be manipulated in the next step of the process. Next, we will create 4 different k-values along with the accompanying U, D, and V matrices. Starting at the first level k1, we are dividing the k-value by 2. Next level k2, we will be dividing the k-value from k1 by 4. Third level k3, we will be dividing k2 by 8. On the last level k4, we will be dividing k3 by 8. With the U, D, and V matrices for the accompanying k-values we want to set the range from the first column to the k-value for that level. The next step for this process is to reconstruct the photo with the U, D, and V matrices.

```
# Step 3: Retain k singular values: divide previous kvalues by doubling the previous divisor

k1 = round(ktotal/2, digits =0) # first k value when we half ktotal

k1 # 1380

U_k1 <- U[, 1:k1]

D_k1 <- D[1:k1, 1:k1]

V_k1 <- V[, 1:k1]

k2 = round(k1/4, digits = 0) # second k value

k2 # 345

U_k2 <- U[, 1:k2]

D_k2 <- D[1:k2, 1:k2]

V_k2 <- V[, 1:k2]

k3 = round(k2/8, digits = 0) # thrid k value

k3 # 58

U_k3 <- U[, 1:k3]

D_k3 <- D[1:k3, 1:k3]

V_k3 <- V[, 1:k3]

k4 = round(k3/8,digits =0) # fourth k value using /8 because /16 gives us too small of a number
that even we cannot make out what the celebrity is.

k4 # 5

U_k4 <- U[, 1:k4]

D_k4 <- D[1:k4, 1:k4]

V_k4 <- V[, 1:k4]
```

To recreate the photo with the matrices at specified k-values we have to remember the formula for SVD which is:

$$A = U * D * V^{\top}$$

When we reconstruct the matrices at the specified k-values we are evaluating from the first mode to the k-value as follows:

$$A\ =\ d_1 * u_1 * (vt)\_1 + \ldots + d_k * u_k * (vt)\_k$$

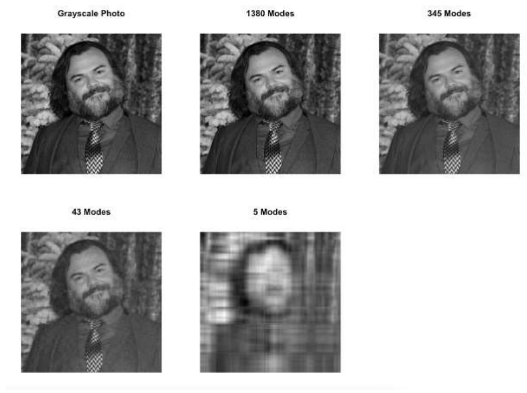
This allows us to approximate the image A at the various levels of k. One thing to note is that these variables are all singular matrices and not scalars. We must also remember that to successfully reconstruct the photo we must cross multiply the matrices with each other. This is due to the fact that the cross product results in a new matrix, whereas the dot product would give a scalar which is not what we need. We want the final result to be a matrix that represents A.

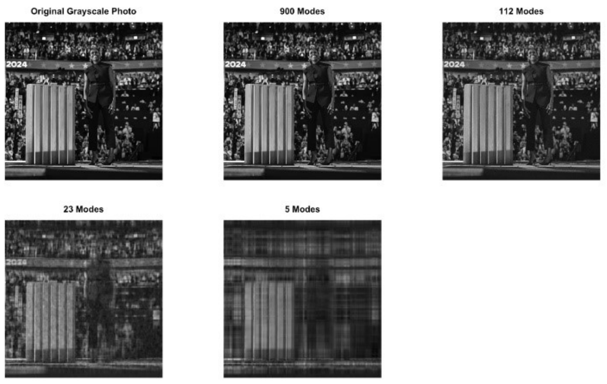
```
# Step 4: Reconstruct the photograph with different k values
par(mar = c(2,2, 2.5, 2.5), mfrow = c(2, 3))
photo_noised1 <- as.cimg(U_k1 %*% D_k1 %*% t(V_k1))
photo_noised2 = as.cimg(U_k2 %*% D_k2 %*% t(V_k2))
photo_noised3 = as.cimg(U_k3 %*% D_k3 %*% t(V_k3))
photo_noised4 = as.cimg(U_k4 %*% D_k4 %*% t(V_k4))
photo_noised5 = as.cimg(U_k5 %*% D_k5 %*% t(V_k5))
```

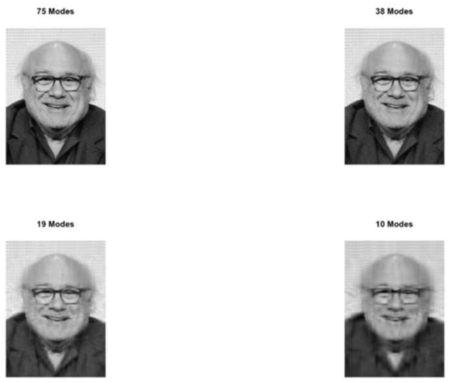
The result of these noised photos will look more and more noised at each further level. This is the direct effect of the differing k-values getting smaller and smaller from the original photo. Next, we will upload the individual photos onto the Tineye search engine and see if the image detection AI can read and identify the image.

Results

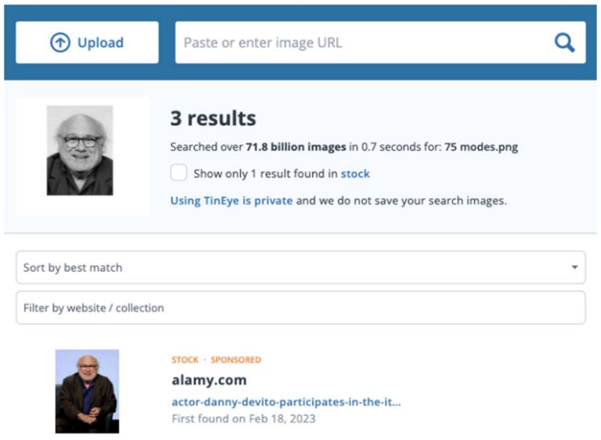
Here is the result of the noise added to the photos by the SVD operation. The images are compiled into a grid but all were individually uploaded to Tineye to search for the photo.







We found that going from 2 to 3 makes a noticeable difference with the higher resolution photos. With Devito photos, notice how the 10 mode image is not that different from the other photos with reduced modes. Compare this to Obama's and Black's, which have a much noticeable difference with the images at 5 modes. This highlights how important resolution can be when it comes to SVD reconstruction.



Upload

Paste or enter image URL

3 results

Searched over **71.8 billion images** in 0.7 seconds for: **75 modes.png**

Show only 1 result found in stock

Using TinEye is private and we do not save your search images.

Sort by best match

Filter by website / collection

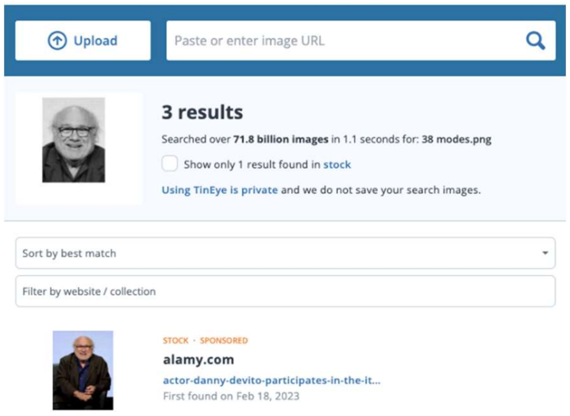
STOCK · SPONSORED

alamy.com

actor-danny-devito-participates-in-the-it...

First found on Feb 18, 2023

Devito's 75 mode image was able to find matches



Upload

Paste or enter image URL

3 results

Searched over **71.8 billion images** in 1.1 seconds for: **38 modes.png**

Show only 1 result found in stock

Using TinEye is private and we do not save your search images.

Sort by best match

Filter by website / collection

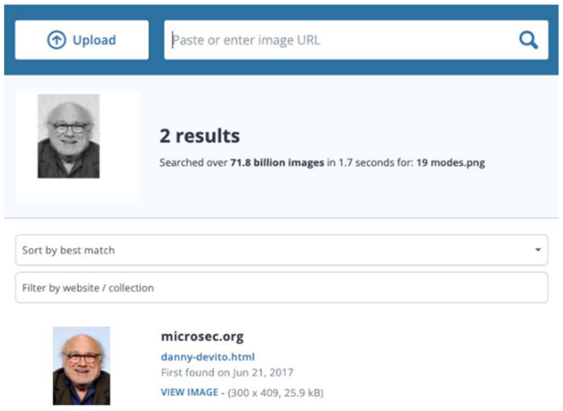
STOCK · SPONSORED

alamy.com

actor-danny-devito-participates-in-the-it...

First found on Feb 18, 2023

Devito's 38 mode image was able to find matches.



Upload

Paste or enter image URL

2 results

Searched over **71.8 billion images** in 1.7 seconds for: **19 modes.png**

Sort by best match

Filter by website / collection

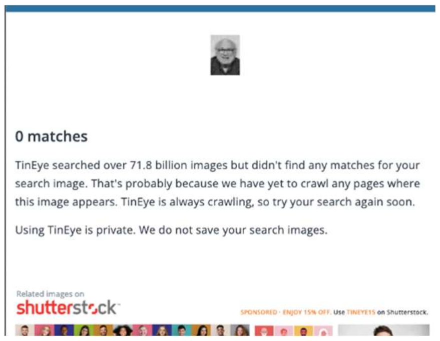
microsec.org

danny-devito.html

First found on Jun 21, 2017

VIEW IMAGE - (300 x 409, 25.9 kB)

Devito’s 10 mode image was able to find matches.

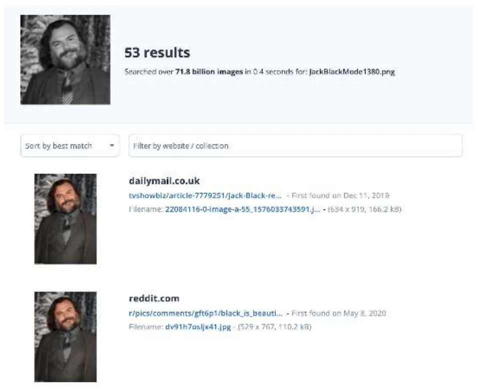


0 matches

TinEye searched over 71.8 billion images but didn't find any matches for your search image. That's probably because we have yet to crawl any pages where this image appears. TinEye is always crawling, so try your search again soon.

Using TinEye is private. We do not save your search images.

Devito’s 10 mode image failed to find any matches.



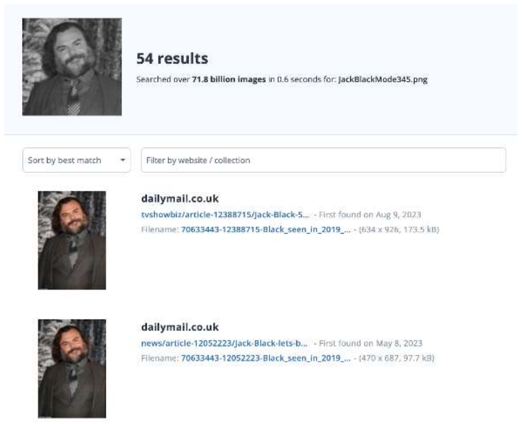
53 results

Searched over **71.8 billion images** in 0.4 seconds for: JackBlackMode1380.png

dailymail.co.uk
tvshowbiz/article-7779251/Jack-Black-re... - First found on Dec 11, 2019
Filename: 22084116-0-image-a-55_1576033743591.j... - (634 x 919, 166.2 kB)

reddit.com
r/pics/comments/gft6p1/black_is_beauti... - First found on May 8, 2020
Filename: dv91h7osljx41.jpg - (529 x 767, 110.2 kB)

Black’s 1380 mode image was able to find matches



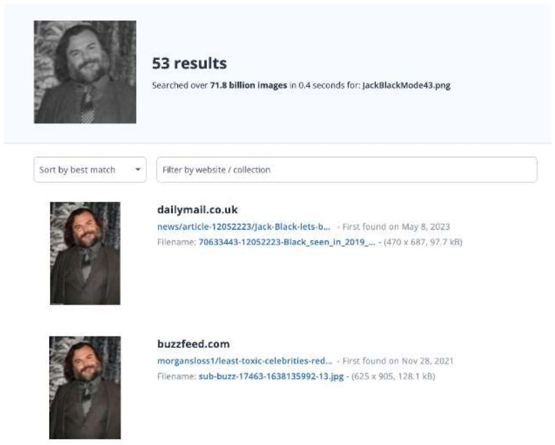
54 results

Searched over **71.8 billion images** in 0.6 seconds for: JackBlackMode345.png

dailymail.co.uk
tvshowbiz/article-12388715/Jack-Black-5... - First found on Aug 9, 2023
Filename: 70633443-12388715-Black_seen_in_2019_... - (634 x 926, 173.5 kB)

dailymail.co.uk
news/article-12052223/Jack-Black-lets-b... - First found on May 8, 2023
Filename: 70633443-12052223-Black_seen_in_2019_... - (470 x 687, 97.7 kB)

Black’s 345 mode image was able to find matches.



53 results

Searched over **71.8 billion images** in 0.4 seconds for: **JackBlackMode43.png**

Sort by best match

Filter by website / collection

dailymail.co.uk

news/article-12052223/Jack-Black-lets-b... - First found on May 8, 2023

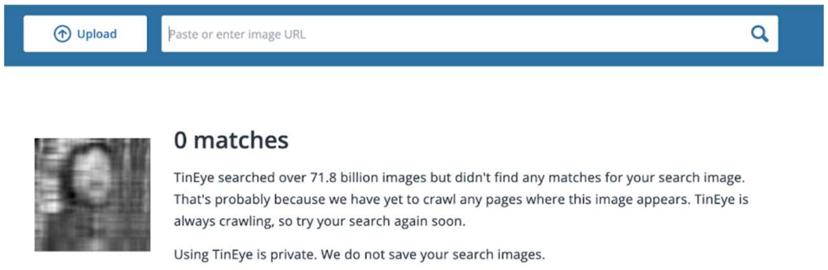
Filename: 70633443-12052223-Black_seen_in_2019_... - (470 x 687, 97.7 kB)

buzzfeed.com

morgansloss1/least-toxic-celebrities-red... - First found on Nov 28, 2021

Filename: sub-buzz-17463-1638135992-13.jpg - (625 x 905, 128.1 kB)

Black's 43 mode image was able to find matches.

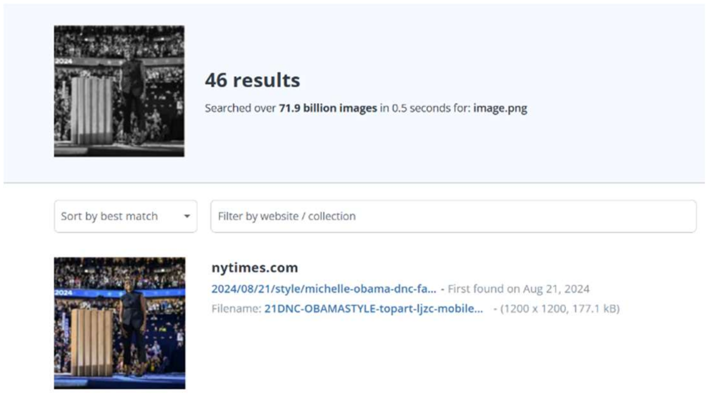


0 matches

TinEye searched over 71.8 billion images but didn't find any matches for your search image. That's probably because we have yet to crawl any pages where this image appears. TinEye is always crawling, so try your search again soon.

Using TinEye is private. We do not save your search images.

Black's 5 mode image was not able to find any matches.



46 results

Searched over **71.9 billion images** in 0.5 seconds for: **image.png**

Sort by best match

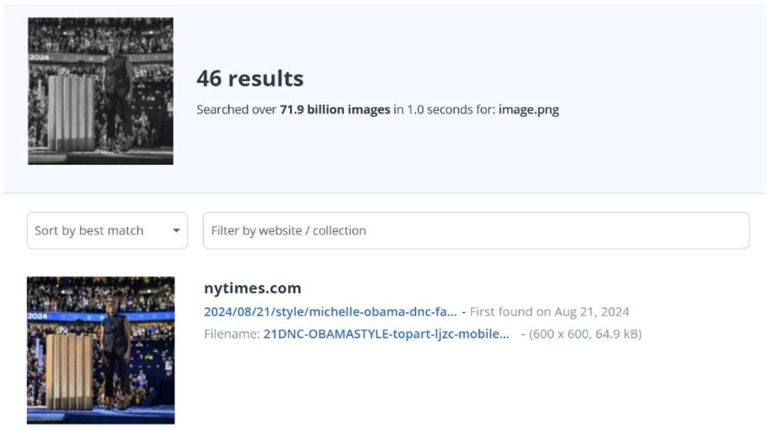
Filter by website / collection

nytimes.com

2024/08/21/style/michelle-obama-dnc-fa... - First found on Aug 21, 2024

Filename: 21DNC-OBAMASTYLE-topart-ljzc-mobile... - (1200 x 1200, 177.1 kB)

Obama's 900 mode image was able to find matches


46 results

Searched over **71.9 billion images** in 1.0 seconds for: **image.png**

Sort by best match

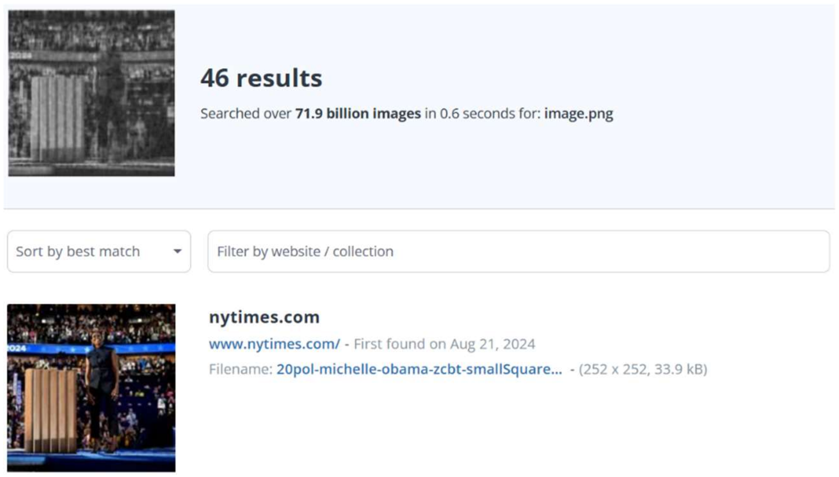
Filter by website / collection

nytimes.com

2024/08/21/style/michelle-obama-dnc-fa... - First found on Aug 21, 2024

Filename: 21DNC-OBAMASTYLE-topart-ljzc-mobile... - (600 x 600, 64.9 kB)

Obama’s 112 mode image was able to find matches.


46 results

Searched over **71.9 billion images** in 0.6 seconds for: **image.png**

Sort by best match

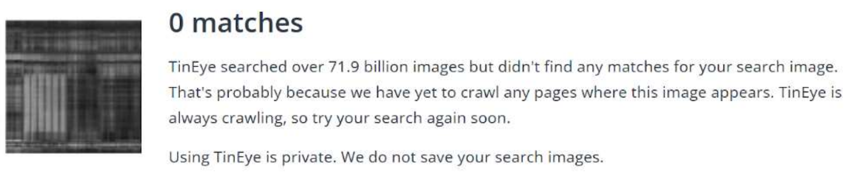
Filter by website / collection

nytimes.com

www.nytimes.com/ - First found on Aug 21, 2024

Filename: 20pol-michelle-obama-zcbt-smallSquare... - (252 x 252, 33.9 kB)

Obama’s 23 mode image was able to find matches.


0 matches

TinEye searched over 71.9 billion images but didn't find any matches for your search image. That's probably because we have yet to crawl any pages where this image appears. TinEye is always crawling, so try your search again soon.

Using TinEye is private. We do not save your search images.

Obama’s 5 mode image failed to find any matches.

The results of this were rather interesting. The AI was able to detect the photos until it got to the last modes which were the smallest k-value we used. In Devito's it failed at k=10 while Obama's and Black's images failed at k=5. Resolution of the photo definitely affects the reconstruction of the image. Obama's original photo resolution was 1800x1800, knowing that it failed to detect at 5 modes means that out of the 1800 singular vectors from the image rendering only 0.00277 percent were able to be reconstructed. Looking at the next image in Obama's set which is the 23 mode image, it was able to find matches. Out of the 1800 singular vectors only the first 23 were needed for the reconstructed photo to find matches. This would be about 0.0128 percent of the vectors from the original photo to be detected by the AI. Now look at Devito's image set, the 10 mode image failed to find any matches, out of the 300 this was only 0.0333 percent of the original photo's vectors. The next image which was at 38 modes was able to find matches. This accounted for 0.1267 percent of the original vectors to be detected by the AI. Looking at Black's image set now, the 5 image mode failed to find any matches. 5 out of the 2760 original vectors which is 0.0018 percent. Next would be the 43 mode image which was able to find matches accounted for only 0.1558 percent of the original vectors. It is apparent that the AI is able to detect photos with less than 1 percent of the reconstructed photo's matrices. This is good, but we must remember that the matrices at the lower numbers have higher D values meaning the importance at the beginning of the data set starting at 1 are valued more than the data set at 1800 for example.

Conclusion

This paper started off as a way to test image detection on Chatgpt, but we quickly learned that Chatgpt's image detection is still in its infancy age as the AI cannot detect individuals in a photo. Changing over to Tineye we discovered that if we use the beginning parts of the data we are able to reconstruct more of the photo versus reconstructing the photo on the other end of the data set instead of starting at 1. Tineye was able to detect the photos with less than 1 percent of the approximated data from the original photo. The benefits of this method allows us to control the k-values and dictate how much a photo will reconstruct to approximate image A. The con of this method is that we are only evaluating at the starting point of the first set to the specified k-value. As we remember, the values at the beginning of the data set have a bigger impact on the reconstruction of the photo. If we were to discuss alternatives of this problem, I would also want to evaluate the reconstruction of the photo using the other end of the data set and see how successful the AI would detect the photos using the same percentage of k-values used vs the original photo's dimensions.

Peer Review: Laura Pineda

Starting from the abstract, the writer makes it clear as to what the research paper that was to follow will entail and I appreciated that not only was it concise but also quite a great summary when I read it in hindsight. I found it interesting how the AI generator Tineye was introduced since it was one of the few AI search engines that specializes in facial and reverse image search. This eliminates the question as to why the typical AI engines were not considered in the research.

I found the data and method section to be well organized and I valued that the code was embedded with explicit step numbers to easily follow along with the writer's thought process and procedure. Although simple calculations brought the specific k values that were to be noised, the step-by-step tutorial on how these numbers were acquired made the process very clear to follow. The reasoning as to why the D matrix in the SVD analyses of each unique photo was adjusted was well explained though the topic was slightly just brushed over. Although I understood exactly why the D matrix was important, I wonder if a client would think the same.

The results of the research project were quite interesting and the percentages formulated to compare each photo was something I wish I had thought of as well. It mathematically compared and placed the significance of the findings of the entire procedure and was written very convincingly for the client to understand and be intrigued by.

The unique solution to an issue like recognizing grainy images on important surveillances was engaging to learn about and the cons revealed something that clients might be skeptical about but at least they have the full truth on what the service fully entails. Overall, the report made me reflect on my conclusions for the service we created and all in all, the report was very informative and clear to follow.

References

TinEye. TinEye Reverse Image Search, www.tineye.com. Accessed 6 Dec. 2024.

Full R Code

```
setwd("C:/Main/College Shiiiit/Math 524/data/data")

#Jack Black Photo

# Step 1: Load the grayscale photograph

library(imager)

photo <- load.image("jackBlack.jpg")  # Replace with your photo path

photo = grayscale(photo)

dim(photo)

# [1] 2760 2760    1    1



# Step 2: Perform SVD on the image

svd_photo <- svd(photo)

U = svd_photo$u

D = diag(svd_photo$d)  # Convert singular values into a diagonal matrix

V = svd_photo$v



ktotal = length(svd_photo$d) # find total K values

ktotal
```

```
#[1] 2760


# Step 3: Retain k singular values: divide previous kvalues by doubling the previous divisor

k1 = round(ktotal/2, digits =0) # first k value when we half ktotal

k1 # 1380

U_k1 <- U[, 1:k1]

D_k1 <- D[1:k1, 1:k1]

V_k1 <- V[, 1:k1]


k2 = round(k1/4, digits = 0) # second k value

k2 # 345

U_k2 <- U[, 1:k2]

D_k2 <- D[1:k2, 1:k2]

V_k2 <- V[, 1:k2]


k3 = round(k2/8, digits = 0) # thrid k value

k3 # 43

U_k3 <- U[, 1:k3]
```

```
D_k3 <- D[1:k3, 1:k3]

V_k3 <- V[, 1:k3]



k4 = round(k3/8,digits =0) # fourth k value using /8 because /16 gives us too small of a number
that even we cannot make out what the celebrity is.

k4 # 5

U_k4 <- U[, 1:k4]

D_k4 <- D[1:k4, 1:k4]

V_k4 <- V[, 1:k4]



# Step 4: Reconstruct the photograph with k=50 and plot original photo

dev.off() # prepare plot for photos...



# noised photos

par(mar = c(2,2, 2.5, 2.5), mfrow = c(2, 3))

plot(imrotate(photo,180)

    xlim = c(0, dim(photo)[1]), ylim = c(0, dim(photo)[2]),

    axes = F,

    main = "Grayscale Photo")
```

```
photo_noised1 <- as.cimg(U_k1 %*% D_k1 %*% t(V_k1))

plot(photo_noised1, axes = F, main = "1380 Modes")

photo_noised2 <- as.cimg(U_k2 %*% D_k2 %*% t(V_k2))

plot(photo_noised2, axes = F, main = "345 Modes")

photo_noised3 <- as.cimg(U_k3 %*% D_k3 %*% t(V_k3))

plot(photo_noised3, axes = F, main = "43 Modes")

photo_noised4 <- as.cimg(U_k4 %*% D_k4 %*% t(V_k4))

plot(photo_noised4, axes = F, main = "5 Modes")

dev.off()


#Danny Devito Photo

# Step 1: Load the grayscale photograph

photo <- load.image("dannyDevito.jpg") # Insert photo name

dim(photo) #Last dimension should be 3

#[1] 300 409   1   3

photo = grayscale(photo)

dim(photo) # Last dimension should be 1 now
```

```
#[1] 300 409   1   1


# Step 2: Perform SVD on the image

svd_photo <- svd(photo)

U <- svd_photo$u

D <- diag(svd_photo$d)  # Convert singular values into a diagonal matrix

K = length(svd_photo$d) # find total K values

K

#[1] 300

V <- svd_photo$v


# Step 3a: Adjust the number of singular values to retain

k1 <- 75# divided 400 by 5

U_k1 <- U[, 1:k1]

D_k1 <- D[1:k1, 1:k1]

V_k1 <- V[, 1:k1]


k2 <- 38  # divided 75 by 1/2
```

```
U_k2 <- U[, 1:k2]

D_k2 <- D[1:k2, 1:k2]

V_k2 <- V[, 1:k2]



k3 <- 19  #divided 38 by 1/2

U_k3 <- U[, 1:k3]

D_k3 <- D[1:k3, 1:k3]

V_k3 <- V[, 1:k3]



k4 <- 10  #divided 19 by 1/2

U_k4 <- U[, 1:k4]

D_k4 <- D[1:k4, 1:k4]

V_k4 <- V[, 1:k4]



par(mar = c(2,2, 2.5, 2.5), mfrow = c(2, 3))

plot(imrotate(photo,180),

    xlim = c(0, dim(photo)[1]), ylim = c(0, dim(photo)[2]),

    axes = F,
    main = "Grayscale Photo")


# noised photos

photo_noised1 <- as.cimg(U_k1 %*% D_k1 %*% t(V_k1))

plot(photo_noised1, axes = F, main = "75 Modes")



photo_noised2 <- as.cimg(U_k2 %*% D_k2 %*% t(V_k2))

plot(photo_noised2, axes = F, main = "38 Modes")



photo_noised3 <- as.cimg(U_k3 %*% D_k3 %*% t(V_k3))

plot(photo_noised3, axes = F, main = "19 Modes")



photo_noised4 <- as.cimg(U_k4 %*% D_k4 %*% t(V_k4))

plot(photo_noised4, axes = F, main = "10 Modes")



# Step 3b: Adjust the number of singular values to retain (Starting at a different K values)

## Recall total K = 400

k1 <- 2:400
```

```
U_k1 <- U[, k1]

D_k1 <- D[k1, k1]

V_k1 <- V[, k1]



k2 = 3:400 # start of k1 + 1

U_k2 <- U[, k2]

D_k2 <- D[k2, k2]

V_k2 <- V[, k2]



#Lowest k to start is 2

k3 = 2:100 # K/4

U_k3 <- U[, k3]

D_k3 <- D[k3, k3]

V_k3 <- V[, k3]



k4 = 2:25 # K/4

U_k4 <- U[, k4]

D_k4 <- D[k4, k4]
```

```
V_k4 <- V[, k4]


k5 = 2:12 # K/2

U_k5 <- U[, k5]

D_k5 <- D[k5, k5]

V_k5 <- V[, k5]


# Step 4: Reconstruct the photograph with different k values

par(mar = c(2,2, 2.5, 2.5), mfrow = c(2, 3))

photo_noised1 <- as.cimg(U_k1 %*% D_k1 %*% t(V_k1))

photo_noised2 = as.cimg(U_k2 %*% D_k2 %*% t(V_k2))

photo_noised3 = as.cimg(U_k3 %*% D_k3 %*% t(V_k3))

photo_noised4 = as.cimg(U_k4 %*% D_k4 %*% t(V_k4))

photo_noised5 = as.cimg(U_k5 %*% D_k5 %*% t(V_k5))


plot(photo,

    axes = FALSE, main = "Original Grayscale Photo")
```

```
# Plotting noised photos

plot(photo_noised1, axes = FALSE, main = "2:400 Modes")

plot(photo_noised2, axes = FALSE, main = "3:400 Modes")

plot(photo_noised3, axes = FALSE, main = "2:100 Modes")

plot(photo_noised4, axes = FALSE, main = "2:25 Modes")

plot(photo_noised5, axes = FALSE, main = "2:12 Modes")



dev.off()

#Michelle Obama Photo

# Step 1: Load the grayscale photograph

photo <- load.image("michelleObama.jpg") # Insert photo name

dim(photo) #Last dimension should be 3

#[1] 1800 1800   1   3

photo = grayscale(photo)

dim(photo) # Last dimension should be 1 now

#[1] 1800 1800   1   1

# Step 2: Perform SVD on the image

svd_photo <- svd(photo)
```

```
U <- svd_photo$u

D <- diag(svd_photo$d)  # Convert singular values into a diagonal matrix

K = length(svd_photo$d) # find total K values

K

#[1] 1800

V <- svd_photo$v

# Step 3: Adjust the number of singular values to retain (Starting at a different K values)

## Recall total K = 1800

k1 <- 900    #/2 #y

U_k1 <- U[, 1:k1]

D_k1 <- D[1:k1, 1:k1]

V_k1 <- V[, 1:k1]


k2 <- 112    #/16 #y

U_k2 <- U[, 1:k2]

D_k2 <- D[1:k2, 1:k2]

V_k2 <- V[, 1:k2]
```

```
k3 <- 23     #/64 #y

U_k3 <- U[, 1:k3]

D_k3 <- D[1:k3, 1:k3]

V_k3 <- V[, 1:k3]



k4 <- 5      #/256 #y

U_k4 <- U[, 1:k4]

D_k4 <- D[1:k4, 1:k4]

V_k4 <- V[, 1:k4]



# Step 4: Reconstruct the photograph with different k values

par(mar = c(2,2, 2.5, 2.5), mfrow = c(2, 3))

photo_noised1 = as.cimg(U_k1 %*% D_k1 %*% t(V_k1))

photo_noised2 = as.cimg(U_k2 %*% D_k2 %*% t(V_k2))

photo_noised3 = as.cimg(U_k3 %*% D_k3 %*% t(V_k3))

photo_noised4 = as.cimg(U_k4 %*% D_k4 %*% t(V_k4))



plot(photo,
    axes = FALSE, main = "Original Grayscale Photo")

# Plotting noised photos

plot(photo_noised1, axes = FALSE, main = "900 Modes")

plot(photo_noised2, axes = FALSE, main = "112 Modes")

plot(photo_noised3, axes = FALSE, main = "23 Modes")

plot(photo_noised4, axes = FALSE, main = "5 Modes")
```