Adult Image Classification

By

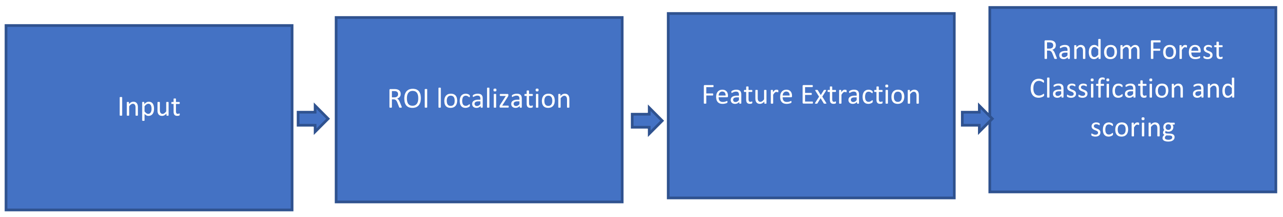
Manikanda Krishnan V (11105EN002)

S Mohana Krishna (11105EN006)

Introduction:

Increasing accessibility of the internet also implies the increasing accessibility to objectionable content. The impact of pornography on the human psyche especially children has been the subject of study for many decades. Psychological research dated back to the 80's stressed out that the exposure of children to pornography affects their behavioural development. The primary source of pornographic information on the Web has traditionally been pornographic images and video. So our project is to design a model which given images from a url can classify them as adult content or not and hence be used to block the website.

Method:



ROI Localization: Each pixel in the image is scored as a skin pixel using a logistic classifier trained on the Compaq skin pixel dataset. The features used to train the logistic classifier are Hue,Cb and Cr and A and B(LAB color space) .These features were chosen to minimize the influence of illumination. Once the scoring is done. We find the median value of pixels whose score was > 0.75 and normalize the pixel values based on distance from the median value . We then assign probability values based on how small the distances are , that is the value that is furthest from the median gets a value of zero and so on. We then filter out only those pixels whose probability > 0.9. This region is our ROI.

Feature Extraction: We extract the following features from the ROI.

- Area of ROI to overall image
- Hu's moments
- Orientation
- Eccentricity
- Area
- Perimeter
- Kurtosis

These features are then used to train a random forest with 300 trees into 4 classes. We make the final classification based on using weighted sum of the scores given to the 4 classes. The use of weights helps us to tune the sensitivity based on different concepts of what is considered as vulgar in different countries.

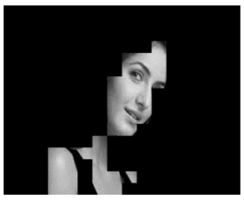
We aim to have fewer false positives as we do not wish to misclassify benign sites and we classify a site as adult if 3 images out of 10 random images from the site are classified as adult.

Data

Training Data: 2679 images

Test Data: 383 images

Conclusion



ROI Localization

Since we had used random forests, the training error was < 2%.

According to the startup(inetclean) who gave us this problem we are getting around 73% accuracy for single image classification. The accuracy for classifying a website is much higher as we take 15 samples and the site is considered blocked if more than 5 images are classified as adult.

Scope for Enhancement:

The ROI is extracted using skin pixels, this is an issue in the case of black and white images or photoshopped images where the colors no longer represent humans. Incorporating shape features might help alleviate these issues.

Video Analytics incorporating speech information might also help improve accuracy.

Code:

```
% input name - absolute path of image on a unix machine
% output - score - 0 for clean image, 10 for porn with high confidence

function [score]=predict_image(name)
% just to close any open image windows. Mostly during debugging of code
close all
% read the image into an array
image=imread(name);
```

```
% machine learning is carried out at 320x320 size, so all input images are
resized to this.
image=imresize(image,[320,320]);
imshow(image)
% convert into gray image.
grayimage=rgb2gray(image);

% calculate the no. of rows and columns.
[nofrows,nofclmns]=size(grayimage);

%figure,imshow(image)

% calls our own routine for skin detection
SkinPxls=skin(image);

parts=4;                                % 2^parts-1 is the number of parts
each row and coloumn is divided this decides how many squares we process

newgray=zeros(nofrows,nofclmns);           % this is to store the grayscale
version of image satisfying the threshold

%initialize matrix(2^parts,2^parts) with all zeroes.
s=zeros(2^parts,2^parts);

for i=1:2^parts
    for j=1:2^parts
    % in the small region considered the pxls which correpond to skin %
skin pixels in the square considered
        b1(:,:)=SkinPxls((i-1)*(nofrows/2^parts)+1:i*(nofrows/2^parts),(j-
1)*(nofclmns/2^parts)+1:j*(nofclmns/2^parts));

        Noofpxls=sum(b1(:));           % no of skin pxls in the small
region
        totalpxls=(nofrows*nofclmns)/(2^(parts*2));  % total no of pxls in
the region
        SkinPxlsRatio=Noofpxls/totalpxls;
        s(i,j)=SkinPxlsRatio;

        if(s(i,j)>0.4 )
            newgray((i-1)*(nofrows/2^parts)+1:i*(nofrows/2^parts),(j-
1)*(nofclmns/2^parts)+1:j*(nofclmns/2^parts)) = grayimage((i-
1)*(nofrows/2^parts)+1:i*(nofrows/2^parts),(j-
1)*(nofclmns/2^parts)+1:j*(nofclmns/2^parts));
        end
    end
end

b(:,:)=newgray(:,:)>0;          % binary matrix with 0 if pxl is blck else
1
bfill=imfill(b,'holes');        % fills the holes in any region of interest
imshow(grayimage.*uint8(bfill));

f=features(bfill,image);
load forest
load x     %ignores mean color features
load weight_garden_123
load x_weight_123
```

```
score=compute_score(forest,x,f,weight_tree_1,weight_tree_2,weight_tree_3,x_
weight_1,x_weight_2,x_weight_3);

%imn(:,:,1)=bfill.*double(image(:,:,1));
%imn(:,:,2)=bfill.*double(image(:,:,2));
%imn(:,:,3)=bfill.*double(image(:,:,3));
%imshow(uint8(imn));
end

%--------------------------------------------------------------------------
---------------------------------------------------------------------------
-----------
%--------------------------------------------------------------------------
---------------------------------------------------------------------------
-----------

% This function converts pixels from RGB to LAB color space via XYZ color
space,we got the values for conversion from a source,there were multiple
ways to convert RGB to Xyz space and hence we did not use matlab inbuilt
one.
function lab1=rgb2lab(im)
    im1=double(im);
    xyz(:,:,1)=0.412453*im1(:,:,1)+0.357580*im1(:,:,2)+0.180423*im1(:,:,3);
    xyz(:,:,2)=0.212671*im1(:,:,1)+0.715160*im1(:,:,2)+0.072169*im1(:,:,3);
    xyz(:,:,3)=0.019334*im1(:,:,1)+0.119193*im1(:,:,2)+0.950227*im1(:,:,3);

    xref=0.412453*100+0.357580*100+0.180423*100;
    yref=0.212671*100+0.715160*100+0.072169*100;
    zref=0.019334*100+0.119193*100+0.950227*100;


lab1(:,:,1)=((xyz(:,:,2)/yref)>0.008856).*((xyz(:,:,2)/yref).^(1/3))+((xyz(
:,:,2)/yref)<=0.008856).*(7.787.*(xyz(:,:,2)/yref)+16/116);

lab1(:,:,2)=((xyz(:,:,1)/xref)>0.008856).*((xyz(:,:,1)/xref).^(1/3))+((xyz(
:,:,1)/xref)<=0.008856).*(7.787.*(xyz(:,:,1)/xref)+16/116)-
((xyz(:,:,2)/yref)>0.008856).*((xyz(:,:,2)/yref).^(1/3))+((xyz(:,:,2)/yref)
<=0.008856).*(7.787.*(xyz(:,:,2)/yref)+16/116);

lab1(:,:,3)=((xyz(:,:,2)/yref)>0.008856).*((xyz(:,:,2)/yref).^(1/3))+((xyz(
:,:,2)/yref)<=0.008856).*(7.787.*(xyz(:,:,2)/yref)+16/116)-
((xyz(:,:,3)/zref)>0.008856).*((xyz(:,:,3)/zref).^(1/3))+((xyz(:,:,3)/zref)
<=0.008856).*(7.787.*(xyz(:,:,3)/zref)+16/116);
    lab1(:,:,1)=116.*lab1(:,:,1)-16;
    lab1(:,:,2)=500.*lab1(:,:,2);
    lab1(:,:,3)=200.*lab1(:,:,3);
    %imtool(uint8(lab1));
end

% outputs a matrix which is 1 if correponding pxl is skin else 0
%360*hsvimage(i,j,1)>220

%........................skin_detection..........................
function SkinPxls=skin(image)

im1=double(image);

for i=1:320
    for j=1:320
```

```
        % random threshold of RED set as 225 and diff of 100
        if(im1(i,j,1)>225 && ((im1(i,j,1)-im1(i,j,2))>100 || (im1(i,j,1)-
im1(i,j,3))>100))
            % blacken out non-skin pixels
            im1(i,j,1)=0; im1(i,j,2)=0; im1(i,j,3)=0;
        end
    end
end

%contains the hcbcrba vector that is used below, it is a vector of weights
used in classifing pixels as skin/non skin using a logistic regression
model.
load compaq_hcbcrba.mat

%rgb2lab function is defined above converts RGB to LAB space for skin
classification
lab=rgb2lab(im1);
ycbcr=rgb2ycbcr(im1);
hsv=rgb2hsv(im1);

    p3(:,:)=1./(1+exp(-
(hcbcrba(1)+hcbcrba(2)*hsv(:,:,1)*360+(hcbcrba(3)*ycbcr(:,:,2))+(hcbcrba(4)
*double(ycbcr(:,:,3))++(hcbcrba(5)*lab(:,:,3))+(hcbcrba(6)*double(lab(:,:,2
)))))));

   im2(:,:,1)=im1(:,:,1).*(p3(:,:)<0.05);
   im2(:,:,2)=im1(:,:,2).*(p3(:,:)<0.05);
   im2(:,:,3)=im1(:,:,3).*(p3(:,:)<0.05);
   im3(:,:,1)=im1(:,:,1).*(p3(:,:)>0.75);
   im3(:,:,2)=im1(:,:,2).*(p3(:,:)>0.75);
   im3(:,:,3)=im1(:,:,3).*(p3(:,:)>0.75);

   bw=p3>0.75;
   se=strel('square',3);
   bw=imopen(bw,se);
   im3(:,:,1)=im1(:,:,1).*bw;
   im3(:,:,2)=im1(:,:,2).*bw;
   im3(:,:,3)=im1(:,:,3).*bw;
  % figure,imshow(uint8(im3))
   lab3=rgb2lab(im3);
   hsv3=rgb2hsv(im3);
   ycbcr3=rgb2ycbcr(im3);
   h=hsv3(:,:,1);
   cb=ycbcr3(:,:,2);
   cr=ycbcr3(:,:,3);
   a=lab3(:,:,2);
   b=lab3(:,:,3);
   l=lab3(:,:,1);
   vec1=median(h((p3(:,:)>0.75)));
   vec2=median(cb((p3(:,:)>0.75)));
   vec3=median(cr((p3(:,:)>0.75)));
   vec4=median(b((p3(:,:)>0.75)));
   vec5=median(a((p3(:,:)>0.75)));
   vec6=median(l((p3(:,:)>0.75)));

  distMat=(lab(:,:,2)-vec5).*(lab(:,:,2)-vec5);
  normFact=max(distMat(:));
  p4=1-distMat/normFact;
  distMat=(lab(:,:,3)-vec4).*(lab(:,:,3)-vec4);
  normFact=max(distMat(:));
  p4=p4.*(1-distMat/normFact);
  distMat=(ycbcr(:,:,2)-vec2).*(ycbcr(:,:,2)-vec2);
  normFact=max(distMat(:));
  p4=p4.*(1-distMat/normFact);
  distMat=(ycbcr(:,:,3)-vec3).*(ycbcr(:,:,3)-vec3);
  normFact=max(distMat(:));
  p4=p4.*(1-distMat/normFact);
  distMat=(hsv(:,:,1)-vec1).*(ycbcr(:,:,1)-vec1);
  normFact=max(distMat(:));
  p4=p4.*(1-distMat/normFact);

   SkinPxls=p4(:,:)>0.9;

end

%...........................................................features.......
..........................................................................
...................%

% calculates the features of image for predicting whether the image should
% be blocked or not
function f =features(bfill,imagenew)

%computes the areas and Centroids of skin regions.
[bo,l]=bwboundaries(bfill);
st=regionprops(l,'Area','Centroid');
labels=length(st);
Ar=zeros(labels,1);
Centroids=zeros(labels,2);
for s=1:labels
    Ar(s)=st(s).Area;
    Centroids(s,:)=st(s).Centroid;
end

%chooses the.largest 3 region as ROI.
f=zeros(1,22);
f(22)=labels;
if(labels>3)
    labels=3;
end
[weights,index]=sort(Ar,'descend');
bw=bfill.*0;
    x_centre=zeros(labels,1);
    y_centre=zeros(labels,1);
    Ar_centre=zeros(labels,1);

for ind=1:labels
    bw=bw+(l==index(ind));
    x_centre(ind)=Centroids(ind,1);
    y_centre(ind)=Centroids(ind,2);
    Ar_centre(ind)=weights(ind);
end

%Calculates weighted variance of the Centroids based on the top 3 regions
by area.  Weight is 1 if it one of the top 3 regions else zero.
f(16)=var(x_centre,Ar_centre);
f(17)=var(y_centre,Ar_centre);
bw=bwconvhull(bw);           % joins all the regions of interest
```

```
% pxlcnt=sum(bw(:));
% if((pxlcnt/(320*320))*100<=10)
%     bw=zeros(320,320);
% end

image(:,:,1)=bw.*double(imagenew(:,:,1));   % since we got the region of
interest we can extract only the pxls in the region and store them in
imnew for further processing
image(:,:,2)=bw.*double(imagenew(:,:,2));
image(:,:,3)=bw.*double(imagenew(:,:,3));
image=uint8(image);

%imshow(image);

%sees if skin was detected or not
[m,n,k]=size(image);
if(sum(image(:)~=0))
    counter=1;

%gets all skin region values
    notNullPositions1=(image(:,:,1)>0)+(image(:,:,2)>0)+(image(:,:,3)>0);
    notNullPositions=notNullPositions1(:,:)>0;
    pixelVals1=image( :,:,1);
    clmn1=pixelVals1(notNullPositions);
    pixelVals2=image( :,:,2);
    clmn2=pixelVals2(notNullPositions);
    pixelVals3=image( :,:,3);
    clmn3=pixelVals3(notNullPositions);

    b=skin(image);
    hsv=rgb2hsv(image);
    h1=hsv(:,:,1);
    h=h1(:);
    g=rgb2gray(image);

    %imshow(bw);
    totalcount=sum(bw(:)~=0);
    count=sum(b(:)~=0);
    f(1)=count/totalcount;                          % skin pxls ratio
    f(2)=sum(clmn1(:))/count;                       % mean of red
    f(3)=sum(clmn2(:))/count;                       % mean of green
    f(4)=sum(clmn3(:))/count;                       % mean of blue
    f(5)=var(double(clmn1));                        % variance of red
    f(6)=var(double(clmn2));                         % variance of green
    f(7)=var(double(clmn3));                         % variance of blue
    hu=husmoments(g);
    f(8)=hu(1);
    f(9)=hu(2);
    f(10)=hu(3);
    f(11)=hu(4);
    f(12)=hu(5);
    f(13)=hu(6);
    f(14)=hu(7);

    st=regionprops(bw,'Orientation','Area','Perimeter','Eccentricity');
    f(15)=st.Orientation;                             % orientation of image
with x axis
    f(18)=st.Area/((st.Perimeter)*(st.Perimeter));
    f(19)=st.Eccentricity;
    bins=.05:.05:1;
    val=hist(h(h>0),bins);
    f(20)=kurtosis(val);
    f(21)=st.Area/(320*320);
else
    f=-1*ones(1,22);
end
end

%.................................husmoments...........................

% Calculates the 7 husmoments of an image
function hu=husmoments(image)
[m,n]=size(image);
[x,y]=meshgrid(1:n,1:m);

x=x(:);
y=y(:);
image=image(:);
image=double(image);
m00=sum(image);
if(m00==0)
    m00=eps;
end

m10=sum(x.*image);
m01=sum(y.*image);
m11=sum(x.*y.*image);
m20=sum(x.^2.*image);
m02=sum(y.^2.*image);
m30=sum(x.^3.*image);
m03=sum(y.^3.*image);
m12=sum(x.*y.^2.*image);
m21=sum(x.^2.*y.*image);

%------------------central moments-----------------------------------%

xbar=m10/m00;
ybar=m01/m00;

eta11=(m11-ybar*m10)/m00^2;    %corr of xy
eta20=(m20-xbar*m10)/m00^2;    %variance of x
eta02=(m02-ybar*m01)/m00^2;    %variance of y
eta30=(m30-3*xbar*m20+2*xbar^2*m10)/m00^2.5; % skewness of x
eta03=(m03-3*ybar*m02+2*ybar^2*m01)/m00^2.5; % skewness of y
eta21=(m21-2*xbar*m11-ybar*m20+2*xbar^2*m01)/m00^2.5;
eta12=(m12-2*ybar*m11-xbar*m02+2*ybar^2*m10)/m00^2.5;

%-----------------------------------------------------------------------
--%

hu(1)=eta02+eta20;
hu(2)=(eta20-eta02)^2+4*eta11^2;
hu(3)=(eta30-3*eta12)^2+(3*eta21-eta03)^2;
hu(4)=(eta30+eta12)^2+(eta21+eta03)^2;
hu(5)=(eta30-3*eta12)*(eta30+eta12)*((eta30+eta12)^2-
3*(eta21+eta03)^2)+(3*eta21-eta03)*(eta21+eta03)*(3*(eta30+eta12)^2-
(eta21+eta03)^2);
hu(6)=(eta20-eta02)*((eta30+eta12)^2-
(eta21+eta03)^2)+4*eta11*(eta30+eta12)*(eta21+eta03);
```

```
hu(7)=(3*eta21-eta03)*(eta30+eta12)*((eta30+eta12)^2-
3*(eta21+eta03)^2)+(3*eta12-eta30)*(eta21+eta03)*(3*(eta30+eta12)^2-
(eta21+eta03)^2);

end

% computes the final score based on the features.
function[score] =
compute_score(forest,x,data,weight_tree_1,weight_tree_2,weight_tree_3,x_wei
ght_1,x_weight_2,x_weight_3)
m=length(forest);

vote=zeros(size(data,1),m);
for i=1:m
    % i denotes the tree and using x,fe has the features which
    fe=data(:,x(i,:));
    vote(:,i)=predict(forest{i},fe);
end

%voting scheme is as follows benign=0,benign_gray=1,adult_gray=2,adult =3
%this will generate features for stage 2 classification.
%computes the no of tree who have voted as benign, adult,adult_gray,
benign_gray
output_benign=sum(vote==0,2) ;% this has sum of the votes of all the trees
i.e no of ones if they are more than a threshold then site can be blocked,
threshold can be set
output_adult=sum(vote==3,2) ;
output_benign_gray=sum(vote==1,2);
output_adult_gray=sum(vote==2,2);

% features for stage 2 classification
output=[output_benign,output_benign_gray,output_adult_gray,output_adult]

vote_weight_1=zeros(size(data,1),size(x_weight_1,1));
vote_weight_2=zeros(size(data,1),size(x_weight_2,1));
vote_weight_3=zeros(size(data,1),size(x_weight_3,1));

% for i=1:size(x_weight_1,1)
%    % i denotes the tree and using x,fe has the features which
%     fe=output(:,x_weight_1(i,:));
%     vote_weight_1(:,i)=predict(weight_tree_1{i},fe);
% end

%classify the image as block or non block based on the output vector
calculated above,each tree classifies based on 2 features taken at a
time,the features that needs to be considered are given in x_weight _2
for i=1:size(x_weight_2,1)
    % i denotes the tree and using x,fe has the features which
    fe=output(:,x_weight_2(i,:));
    vote_weight_2(:,i)=predict(weight_tree_2{i},fe);
end

% for i=1:size(x_weight_3,1)
%     % i denotes the tree and using x,fe has the features which
%     fe=output(:,x_weight_3(i,:));
%     vote_weight_3(:,i)=predict(weight_tree_3{i},fe);
% end
%(sum(vote_weight_1==1,2)+sum(vote_weight_2==1,2)+sum(vote_weight_3==1,2))
```

```
block=
(sum(vote_weight_2==1,2)+sum(vote_weight_1==1,2)+sum(vote_weight_3==1,2))>2
;

%this will convert 0/1 to 0/10 score
score=block*10;
end
```