T.C.

DÜZCE ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BM480 - DERİN ÖĞRENME

Final Projesi

Furkan SEVGİLİ - 161001043

Mehmed Akif AY - 161002212

Yusuf Sina BAKAN- 161001061

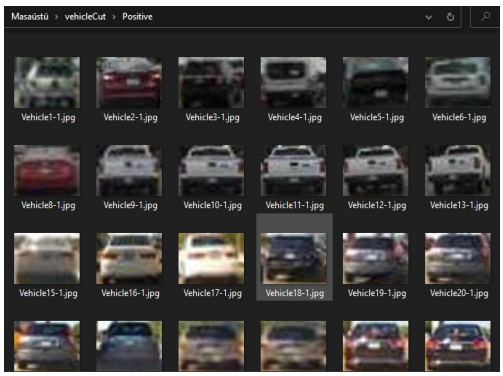
Veri setinin hazırlanması

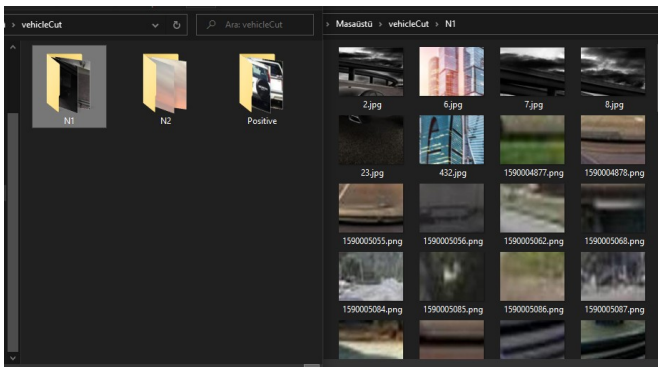
Haziran 2020

Matlab üzerinde erişilen veri seti indirildi, sıkıştırılmış arşiv den çıkarıldı. Daha sonra iki tane iç içe for döngüsü ile veri seti ile beraber gelen GroundTruth matriksinde bulunan değerler referans alınarak resimler kesildi. Aşağıdaki kod da bu kısım bulunmuyor fakat veri setinde bulunan tüm kesilmiş resimlerin genişlik ve yüksekliklerinin ortalamaları 120 ile 91 olduğu hesaplandı. Kesme işleminden sırasında, tüm resimler eşit boyutlarda olması için bu boyutlarda kaydedildi.

```
unzip vehicleDatasetImages.zip
data = load('vehicleDatasetGroundTruth.mat');
vehicleDataset = data.vehicleDataset;
range=size(vehicleDataset);
for i=1:range
    Im=vehicleDataset{i,1};
    bolge=cell2mat(vehicleDataset{i,2});
    s=size(bolge);
        for a=1:s
            I=imread(cell2mat(Im));
            rows=bolge(a,1):bolge(a,1)+bolge(a,3);
            cols=bolge(a,2):bolge(a,2)+bolge(a,4);
            K=I(cols,rows,:);
            K=imresize(K,[91 120]);
            imwrite(K,"./vehicleCut/Vehicle/Vehicle"+int2str(i)+"-"+int2str(a)+".jpg")
        end
end
```

Yukarıdaki kod biraz değiştirilip kesilen 336 araç resminin yanında 336 adet aynı boyutlarda araç olmayan “Negatif” resimler kesildi. Bu “Negatif” resimler N1 ve N2 olmak üzere 2 farklı klasöre paylaştırıldı. Bunu yapmamızın sebebi ileride modeli kullanarak sınıflandırma yaparken, sınıflar için alınan değerlerin binary_classification gibi 1 veya 0 olması yerine yüzdelik değerler olmasını sağlamak.

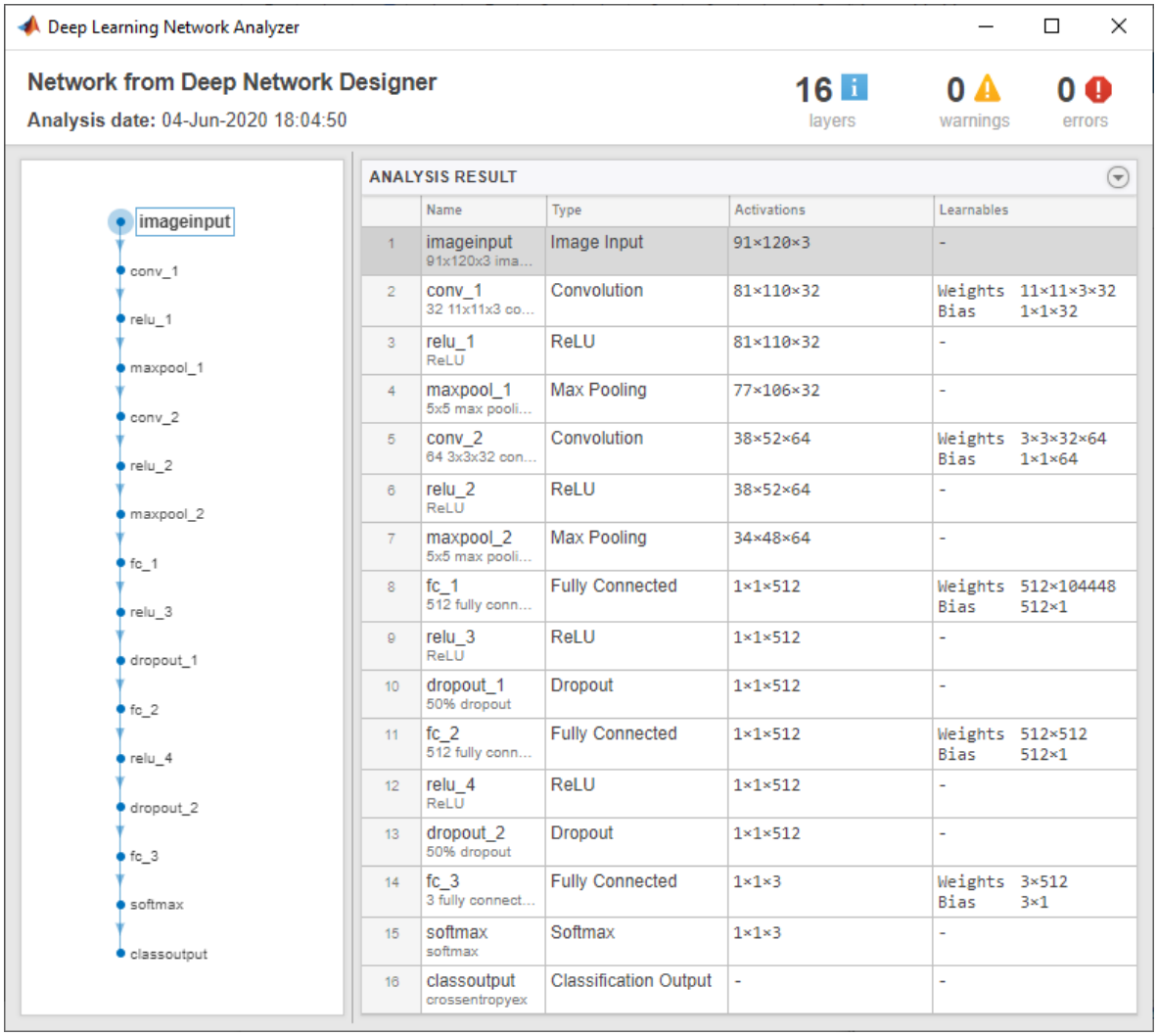




Daha sonra kaydedilen bu resimler aşağıdaki kod parçası kullanılarak okunup eğitim ve test verileri olarak ikiye ayrıldı.

```
allImages=imageDatastore('vehicleCut','IncludeSubfolders',true,'LabelSource','foldernames');
[trainingImages,testImages]=splitEachLabel(allImages,0.8,'randomized');
```

Modelin Oluşturulması ve Eğitilmesi

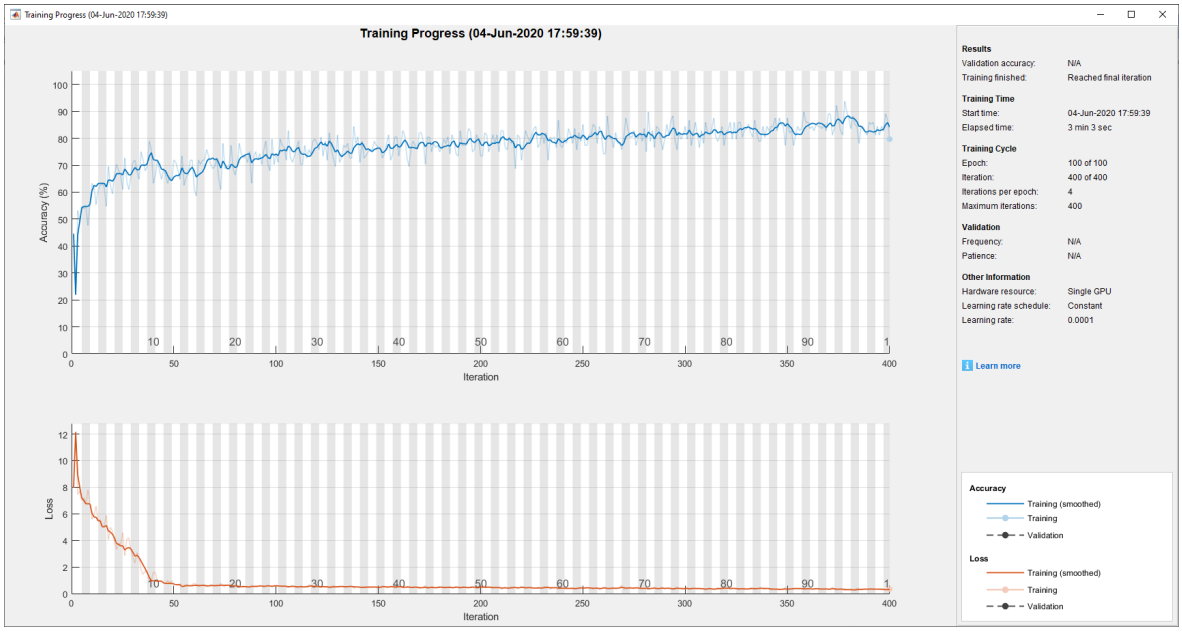


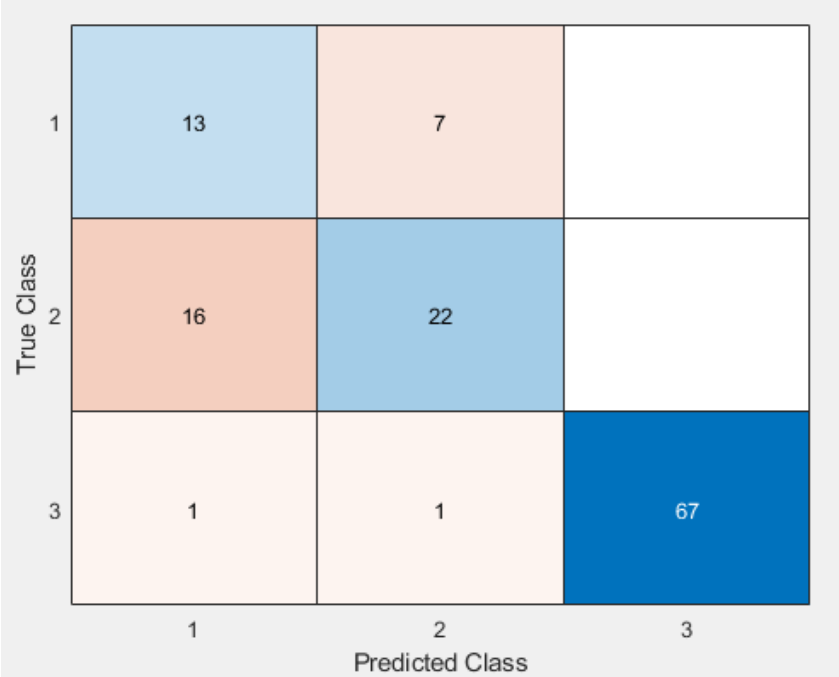
	Name	Type	Activations	Learnables
1	imageinput 91x120x3 ima...	Image Input	91×120×3	-
2	conv_1 32 11x11x3 co...	Convolution	81×110×32	Weights 11×11×3×32 Bias 1×1×32
3	relu_1 ReLU	ReLU	81×110×32	-
4	maxpool_1 5x5 max pooli...	Max Pooling	77×106×32	-
5	conv_2 64 3x3x32 con...	Convolution	38×52×64	Weights 3×3×32×64 Bias 1×1×64
6	relu_2 ReLU	ReLU	38×52×64	-
7	maxpool_2 5x5 max pooli...	Max Pooling	34×48×64	-
8	fc_1 512 fully conn...	Fully Connected	1×1×512	Weights 512×104448 Bias 512×1
9	relu_3 ReLU	ReLU	1×1×512	-
10	dropout_1 50% dropout	Dropout	1×1×512	-
11	fc_2 512 fully conn...	Fully Connected	1×1×512	Weights 512×512 Bias 512×1
12	relu_4 ReLU	ReLU	1×1×512	-
13	dropout_2 50% dropout	Dropout	1×1×512	-
14	fc_3 3 fully connect...	Fully Connected	1×1×3	Weights 3×512 Bias 3×1
15	softmax softmax	Softmax	1×1×3	-
16	classoutput crossentropyex	Classification Output	-	-

Matlab Deep Network Designer kullanılarak yukarıdaki 16 katmanlı ağ yapısı oluşturuldu ve kaydedildi. Daha sonra yanda verilen parametreler kullanılarak model kullanılarak eğitildi.

```
options = trainingOptions('adam', ...
    'MaxEpochs',100,...
    'InitialLearnRate',3e-4, ...
    'ExecutionEnvironment','gpu',...
    'MiniBatchSize',128,...
    'InitialLearnRate',1e-4, ...
    'Verbose',false, ...
    'Plots','training-progress');

mynet = trainNetwork(trainingImages,layers,options);
```





Model %80.31 Accuracy ile eğitimini tamamladı. Yukarıdaki confusion matrix e göre yapılan testlerde model 1. ve 2. sınıflarda çok sayıda hata yapılmış.
Bu durumun sebebi modeli eğitirken kullandığımız iki ayrı “Negatif” veri seti bulunmasıdır. Bu iki veri setinde birbirine benzer resimler bulunduğu için eğitim ve test sırasında çok fazla hata yapılmasına sebep oldu.
Bizim için önemli olan “Pozitif” araç resimlerinin bulunduğu 3. sınıfın doğruluk oranı olduğu için diğer sınıflarda yapılan hataların bir önemi yok. 3. sınıfta yalnızca 2 tane hata yapıldığı için bu model iyi eğitilmiş bir modeldir.

Resim üzerinde arama yapılarak araçların tespiti

Bu bölümde yazılan program, hataların tespiti ve kod üzerinde değişiklik yapmasını kolaylaştırmak amacı ile fonksiyonlar halinde yazılmıştır. Her bir fonksiyon ayrı bir başlık altında incelenmiştir.

detectCarRandom

```
function
[output1,output2]=detectCarRandom(mynet,scene,intersectionThreshold,ra
ndIteration)
s=size(scene);
posArr=[];
for i=1:randIteration
    winc=91*floor((s(1)/91)/(randi(3)+2));
    hinc=120*floor((s(2)/120)/(randi(3)+2));
    wb=randi(s(1)-winc);
    hb=randi(s(2)-hinc);
    ws=wb+winc;
    hs=hb+hinc;
    c=mynet.predict(imresize(scene(wb:ws,hb:hs,:),[91 120]));
    if (c(3)>0.98)
        posArr=[posArr; wb ws hb hs];
    end
end
newPosArr=posArr;
Mx=1;
while(Mx==1)
[newPosArr,ioMx]=mergeIntersections(newPosArr,intersectionThreshold);
    Mx=max(ioMx(:));
end
output2=drawRectangleFromArr(posArr,scene);
output1=drawRectangleFromArr(newPosArr,scene);
count=size(newPosArr(:,1));
```

Araç tespiti için kullanılan ana fonksiyondur.

Parametere olarak sırasıyla cnn ağı, araç için kontrol edilecek resim, çakışma yüzdesi alt sınırı ve kaç tane rastgele arama yapılacagı değerlerini alır.

Verilen resmin içinde bir for döngüsü ile rastgele pozisyon ve boyutlarda kareler de arama yapar.
Bu kareler eğitim sırasında kullanılan 91x120 boyutlarına denk yada bu boyutların katıdır.

Test edilen her kare 91x120 boyutlarına getirilip model tarafından sınıflandırılır.

Eğer sınıflandırma sonucunda 3. sınıf için alınan değer 0.98 den büyük ise bulunan bölgenin koordinatlarını bir dizi içine kaydeder.

Tabiki rastgele arama yapıldığı için karelerin bazıları birbirleri ile çalışacaktır. Bu sorunu çözmek için pozitif test sonuçlarını tutan dizi bir while döngüsü içinde mergeIntersections fonksiyonuna verilir. Bu while döngüsü fonksiyondan dönen intersectionMatrix dizisi içinde 1 değeri olduğu sürece tekrar eder.

detectCarRandom fonksiyonu return değeri olarak. Araçları gösteren karelerin bulunduğu output1 resmini, yine bu karelerin birbirleri ile birleştirilmemiş halde oldukları halini tutan output2 resimini ve tespit edile araç sayısını tutan count değerini döndürür.

mergeIntersections

```
function [narr,ioMx]=mergeIntersections(posArr,intersectionThreshold)
ioMx=detectIntersection(posArr,intersectionThreshold);
newPosArr=[];
visitedIdx=[];
for i=1:size(ioMx,1)
    for j=1:size(ioMx,1)
if((ismember(i,visitedIdx)==0)&&(i~=j)&&(ioMx(i,j)==1||ioMx(j,i)==1))
            visitedIdx=[visitedIdx; j i];

newPosArr=[newPosArr;min(posArr(i,1),posArr(j,1)) ,max(posArr(i,2),pos
Arr(j,2)),min(posArr(i,3),posArr(j,3)),max(posArr(i,4),posArr(j,4))];
        end
    end
end
for i=1:size(ioMx,1)
    if(ismember(i,visitedIdx)==0)

newPosArr=[newPosArr;posArr(i,1) ,posArr(i,2),posArr(i,3),posArr(i,4)]
;
    end
end
narr=newArr;
```

Bu fonksiyon çakışan karelerin birleştirilmesi için kullanılır.

Parametre olarak karelerin bulunduğu pozisyonları tutan matrisi ve çakışma yüzdelerinin alt sınırı olarak kullanılacak değeri alır.

ilk önce detectIntesection fonksiyonu çağrılıp çakışma matrisi oluşturulur.

Bu matris graf teorisindeki komşuluk matrisine benzer bir yapıdadır. Örneğin matrisin 1. satır 7. sütununda bulunan değer 1.karenin 7. kareye çakışma yüzdesini tutar.

Daha sonra 2 adet for döngüsü ile bu matris içinde dolaşılır. Dolaşılan indexler birbiri ile çakışıyor ise bu iki indexte bulunan kareler birleştirilir ve pozisyon dizisine eklenir. Bu esnada aynı çakışmaların tekrar yazılmasını engellemek için kontrol edilen index lerden biri visitedIdx dizisine eklenir.

en son tek bir for döngüsü ile visitedIdx içinde bulunmayan indexler pozisyon dizisine eklenir ve return olarak bu dizi döndürülür.

detectIntersection

```
function iarr=detectIntersection(arr,intersectionThreshold)
arrSize=size(arr);
intersectionMatrix=zeros(arrSize(1));
for i=1:arrSize(1)
    for j=1:arrSize(1)
        if i~=j
            iwb=arr(i,1);
            iws=arr(i,2);
            ihb=arr(i,3);
            ihs=arr(i,4);
            jwb=arr(j,1);
            jws=arr(j,2);
            jhb=arr(j,3);
```

```
            jhs=arr(j,4);
            dx=min(iws,jws)-max(iwb,jwb);
            dy=min(ihs,jhs)-max(ihb,jhb);
            if(dx>0 &&dy >0)
                x=(iws-iwb);
                y=(ihs-ihb);
                intersectionMatrix(i,j)=(dx*dy)/(x*y);
            end
        end
    end
end
iarr=round(intersectionMatrix*100)>intersectionThreshold;
```

iki adet for döngüsü ile parametre olarak gelen pozisyon matrisi içinde dolaşır.Matrisin içindeki köşe değerlerinin karşılaştırır. Eğer çakışma varsa, yüzde kaç çakışma olduğunu hesaplar ve çakışma matrisindeki yerine yazar.

En son çakışma matrisinde, parametre olarak verilen çakışma değerinden büyük olan yüzdelere 1 küçük olanlara 0 değerini verir.

drawRectangleFromArr

```
function scn=drawRectangleFromArr(arr,scene)
s=size(arr);
for i=1:s(1)
    scene=drawRectangle(scene,arr(i,1),arr(i,2),arr(i,3),arr(i,4));
end
scn=scene;
```

Parametre olarak gelen, pozisyon matrisi içerisindeki değerleri arasında bir for döngüsü ile gezinir.

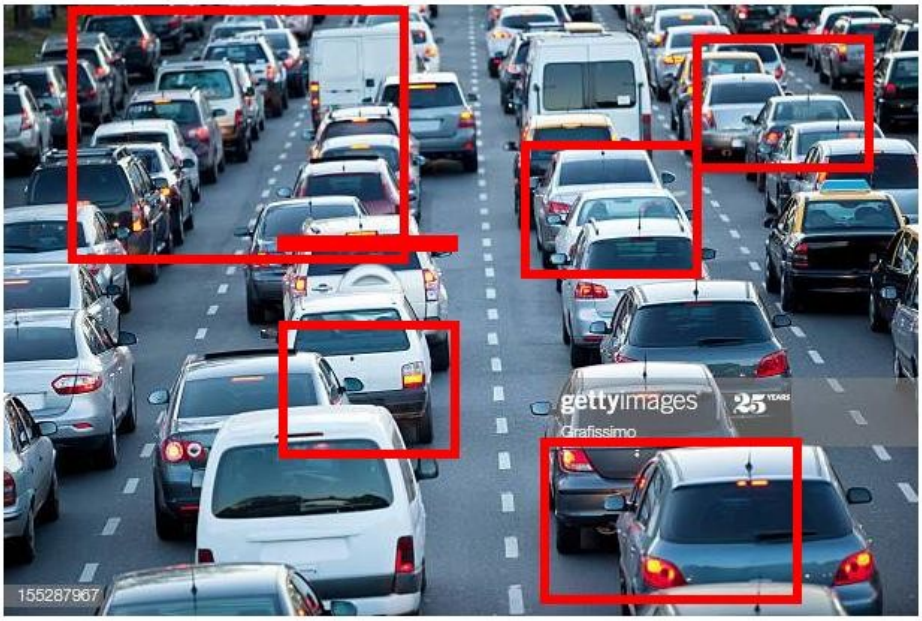
Her bir satır için drawRectangle Fonksiyonunu çağırır.

drawRectangle

```
function s=drawRectangle(scene,wb,ws,hb, hs)
      scene(wb:wb+5,hb:hs,1)=255;
      scene(wb:wb+5,hb:hs,2)=0;
      scene(wb:wb+5,hb:hs,3)=0;
      scene(ws-5:ws,hb:hs,1)=255;
      scene(ws-5:ws,hb:hs,2)=0;
      scene(ws-5:ws,hb:hs,3)=0;
      scene(wb:ws,hb:hb+5,1)=255;
      scene(wb:ws,hb:hb+5,2)=0;
      scene(wb:ws,hb:hb+5,3)=0;
      scene(wb:ws,hs-5:hs,1)=255;
      scene(wb:ws,hs-5:hs,2)=0;
      scene(wb:ws,hs-5:hs,3)=0;
s=scene;
```

Verilen resim üzerinde, verilen köşe değerine uygun 10 piksel genişliğinde kenarlara sahip kırmızı bir kare çizer. Resmi return ile geri gönderir.

Yapılan Testler




KASKOSU YOK AMA
MUSKASI VAR
01 FP 570


DTN 27


SIL 300


PORSCHE
DVQ·66V


ŠKODA
В870РЕ799

Grafik Arayüzü

Arayüzden aldığımız fotoğraf ve daha önceden kayıt edilmiş ağımız parametre olarak araç bulma fonksiyonuna geçilir.

```
function RESMSEButtonPushed(app, event)
            [filename,filepath]=uigetfile({'*.png'},'Bir resim
seçin.');
            fullname = [filepath, filename];
            inputImage = imread(fullname);
            app.Image.ImageSource = fullname;
            load("mynet3out.mat");
            [output1,output2,count] =
detectCarRandom(mynet,inputImage,randi(10),randi(30)+30);
            app.Image.ImageSource = output2;
            pause(2);
            app.Image.ImageSource = output1;
            app.TahminResult.Text = string(count(1));

        end
```