گزارش تمرین کامپیوتری 3 درس سیگنال‌ها و سیستم‌ها

بابک حسینی محتشم 810101408

محمدسینا پرویزی مطلق 810101394

1403/8

بخش اول)

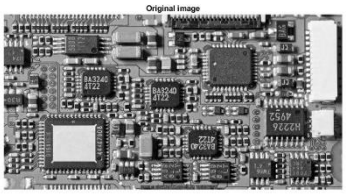
۱- با استفاده از تابع dec2bin مپ ست خواسته شده را تشکیل می‌دهیم:

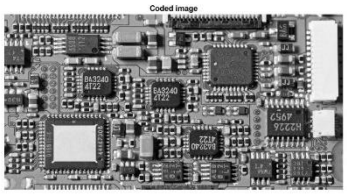
```
len=32;
Mapset=cell(2,len);
for i=1:len
   Mapset{1,i}=char('a'+i-1);
   Mapset{2,i}=dec2bin(i-1,ceil(log2(len)));
end
Mapset{1,27}=char(' ');
Mapset{1,28}=char('.');
Mapset{1,29}=char(',');
Mapset{1,30}=char('!');
Mapset{1,31}=char('"');
Mapset{1,32}=char(';');
save Mapset Mapset;
```

۲- در تابع coding ابتدا با شیفت دادن سپس کم کردن تصویر از تصویر ابتدایی، تغییرات تصویر را پیدا می‌کنیم. سپس تصویر تغییرات را به مربع‌هایی ۵ در ۵ تقسیم می‌کنیم و در هر مربع جمع تغییرات را به دست می‌آوریم.سپس تنها ایندکس نواحی که تغییرات‌شان بیشتر از میانه است را نگه می‌داریم. در نهایت این تغییرات را پیمایش می‌کنیم و بیت کم‌ارزش هر پیکسل را به بیت mام حرف nام پیام تغییر می‌دهیم.

```
function coded_img=coding(msg,img,map)
    shifted_img=img;
    width=size(img,1);
    height=size(img,2);
    shifted_img(1:width-1,1:height-1)=img(2:width,2:height);
    dif_img=abs(shifted_img-img);
    block_len=log2(length(map));
    width=width-mod(width,block_len);
    height=height-mod(height,block_len);
    blocks_sum_dif=zeros([width/block_len,height/block_len],'uint16');
    for i=1:width
        for j=1:height
            blocks_sum_dif(floor((i-1)/block_len)+1,floor((j-1)/block_len)+1)=blocks_sum_dif(floor((i-1)/block_len)+1,floor((j-1)/block_len)+1)+uint16(dif_img(i,j));
        end
    end
    idx=find(blocks_sum_dif(:).'>median(blocks_sum_dif(:)));
    i=1;
    j=1;
    coded_img=img;
    while(j<=strlength(msg))
        if(i>length(idx))
            error('ERROR: The message is too long.');
        end
        [x,y]=ind2sub(size(blocks_sum_dif),idx(i));
        x=(x-1)*block_len+1;
        y=(y-1)*block_len+1;
        i=i+1;
        for k=1:block_len
            c=msg(j);
            bin=map{2,strcmp(map(1, :), c)};
            for m=1:block_len
                color=dec2bin(coded_img(x+k,y+m),block_len);
                color(block_len)=bin(m);
                coded_img(x+k,y+m)=bin2dec(color);
            end
            j=j+1;
            if(j>length(msg))
                break;
            end
        end
    end
end
```

۳- می‌توان دید که تصویر رمزنگاری شده تفاوتی چشمگیر با تصویر اصلی ندارد.





همچنین می‌توان دید که تنها نیاز به تغییر ۲۳ پیکسل برای رمزنگاری این تصویر بودیم:

```
clc;clear;
load Mapset Mapset;
[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
img=imread([path file]);
img=rgb2gray(img);
msg='signal;';
coded_img=coding(msg,img,Mapset);
subplot(1,2,1);
imshow(img);
title('Original image');
subplot(1,2,2);
imshow(coded_img);
title('Coded image');
sum(coded_img-img,'all')
```

Command Window

```
ans =

    23
```

به نظر ما دلیل این که تفاوت دو تصویر آشکار نیست هم همین است زیرا توانستیم تنها با تغییر یک مقدار در تعداد بسیار کمی از پیکسل‌ها پیام را رمزنگاری کنیم.

۴- در تابع decoding مشابه تابع coding نواحی مورد نظر را پیدا می‌کنیم. سپس در این تابع برعکس تابع coding که پیام را داخل پیکسل‌ها قرار می‌داد، با جمع کردن بیت کم‌ارزش پیکسل‌ها حروف را تشکیل داده و با چسباندن این حروف کل پیام را به دست می‌آوریم. این کار را تا جایی ادامه می‌دهیم که به حرف ; برسیم.

```
function msg=decoding(img,map,block_len)
    shifted_img=img;
    width=size(img,1);
    height=size(img,2);
    shifted_img(1:width-1,1:height-1)=img(2:width,2:height);
    dif_img=abs(shifted_img-img);
    width=width-mod(width,block_len);
    height=height-mod(height,block_len);
    blocks_sum_dif=zeros([width/block_len,height/block_len],'uint16');
    for i=1:width
        for j=1:height
            blocks_sum_dif(floor((i-1)/block_len)+1,floor((j-1)/block_len)+1)=blocks_sum_dif(floor((i-1)/block_len)+1,floor((j-1)/block_len)+1)+uint16(dif_img(i,j));
        end
    end
    idx=find(blocks_sum_dif(:).'>median(blocks_sum_dif(:)));
    i=1;
    j=1;
    coded_img=img;
    msg='';
    c='';
    end_char=map{1,length(map)};
    while(i<=length(idx) && ~strcmp(c,end_char))
        [x,y]=ind2sub(size(blocks_sum_dif),idx(i));
        x=(x-1)*block_len+1;
        y=(y-1)*block_len+1;
        i=i+1;
        for k=1:block_len
            bin='';
            for m=1:block_len
                color=dec2bin(coded_img(x+k,y+m),block_len);
                bin(m)=color(block_len);
            end
            c=map{1,strcmp(map(2, :), bin)};
            msg(j)=c;
            j=j+1;
            if(c==end_char)
                break;
            end
        end
    end
end
```

می‌توان دید که این تابع به درستی پیام را رمزگشایی کرده است:

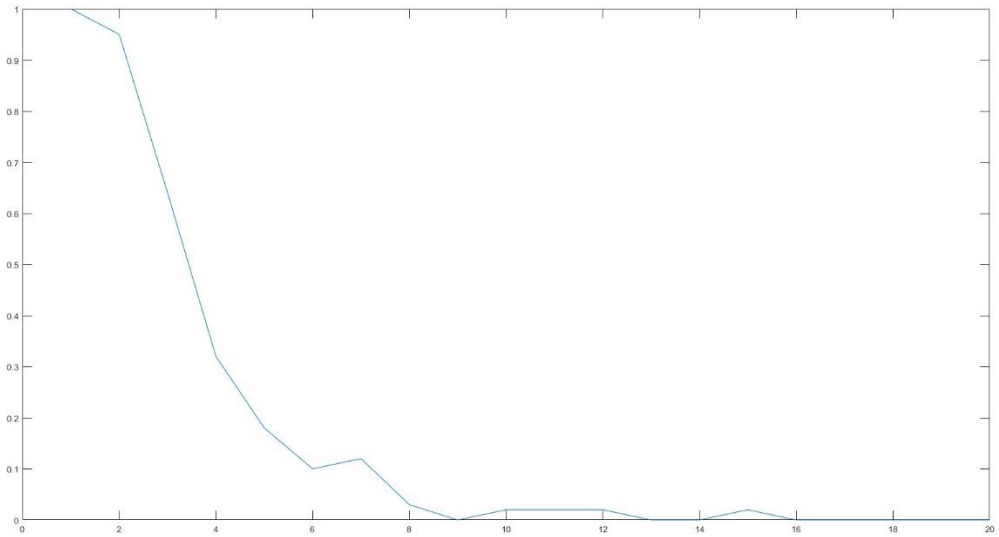
```
clc;clear;
load Mapset Mapset;
[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
img=imread([path file]);
img=rgb2gray(img);
msg='signal;';
coded_img=coding(msg,img,Mapset);
decoding(coded_img,Mapset,ceil(log2(length(Mapset))))
```

```
ommand Window

ans =

    'signal;'
```

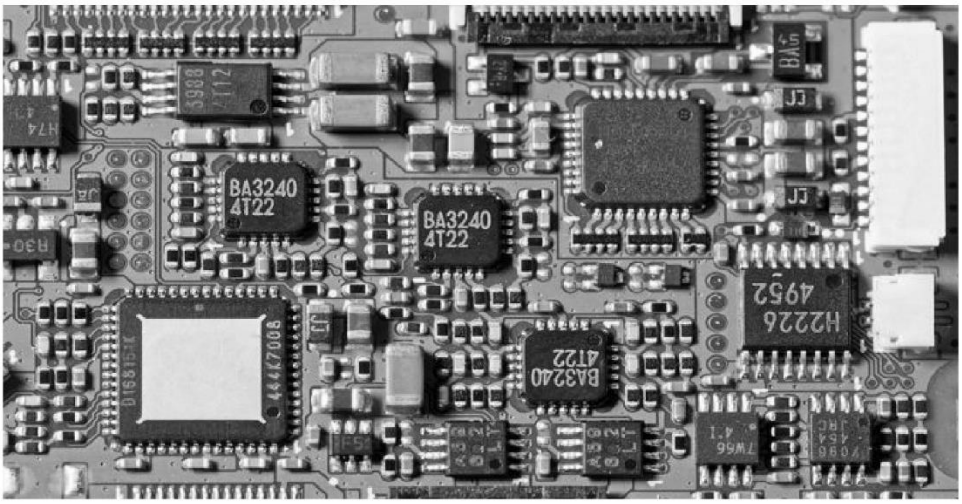
۵- حدس می‌زنیم که با اضافه کردن نویز بیشتر باعث تغییر تعداد بیشتری از پیکسل‌ها شویم و در نتیجه پیام گرفته شده نادرست شود. برای تست این فرض تصویر را برای نویز نرمال از مقدار صفر تا نویز ۲ برابر رمزنگاری و سپس رمزگشایی می‌کنیم و درصد دفعاتی که متن را به درستی رمزنگاری کردیم به دست می‌آوریم:

```
clc;clear;
load Mapset Mapset;
[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
img=imread([path file]);
img=rgb2gray(img);
msg='signal;';
coded_img=coding(msg,img,Mapset);
n=100;
m=20;
correct=zeros(1,m);
first_wrong_img=-1;
first_wrong_msg=-1;
last_wrong_img=-1;
last_wrong_msg=-1;
for j=1:m
    for i=1:n
        noisy_img=coded_img+uint8(0.1*j*randn(size(coded_img)));
        decoded_msg=decoding(noisy_img,Mapset,ceil(log2(length(Mapset))));
        if(strcmp(msg,decoded_msg))
            correct(j)=correct(j)+1;
        else
            if(first_wrong_img==-1)
                first_wrong_img=noisy_img;
                first_wrong_msg=decoded_msg;
            end
            last_wrong_img=noisy_img;
            last_wrong_msg=decoded_msg;
        end
    end
    correct(j)=correct(j)/n;
end
figure;
plot(1:m,correct);
figure;
imshow(first_wrong_img);
title(first_wrong_msg);
figure;
imshow(last_wrong_img);
title(last_wrong_msg);
```

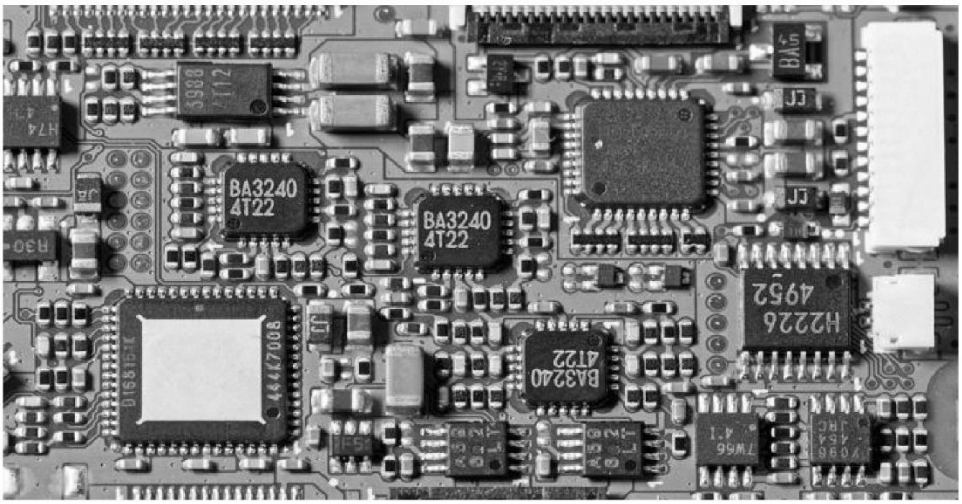
می‌توان دید همان طور که انتظار داشتیم با افزایش نویز، دقت مدل پایین می‌آید.



This is the first wrong decoded message with 0.2x noise: sagnal;



This is the last wrong decoded message with 2.0x noise: cihnaltvr!atibfap"dtkn o!m;



می‌توان دید گه حتی با نویز ۲ برابر هم تصویر متفاوت دیده نمی‌شود همچنین می‌توان مشاهده کرد که چقدر پیام رمزگشایی شده با پیام ابتدایی با افزایش نویز تفاوت دارد.

بخش دوم)

در تابع ICrecognition ابتدا تصاویر را سیاه و سفید می‌کنیم. سپس تصویر ۱۸۰ درجه چرخش یافته IC را هم تولید می‌کنیم.

```
function img=ICrecognition(pcb,ic)
    pcb=rgb2gray(pcb);
    ic=rgb2gray(ic);
    box_width=size(ic,1);
    width=size(pcb,1);
    box_height=size(ic,2);
    height=size(pcb,2);
    img=pcb;
    ic=double(ic);
    pcb=double(pcb);
    norm2=sum(ic.^2,"all");
    correlation=zeros((width - box_width + 1),(height - box_height + 1));
    rot_correlation=zeros((width - box_width + 1),(height - box_height + 1));
    rot_ic=rot90(ic,2);
    rot_norm2=sum(ic.^2,"all");
```

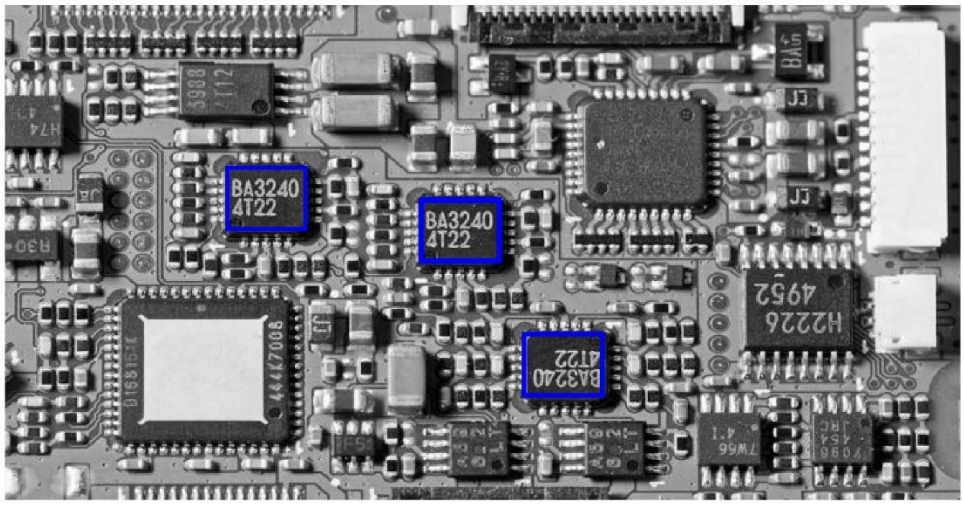
سپس تصویر هر دو IC را در تصویر PCB پیمایش می‌کنیم و دو مجموعه correlation را حساب می‌کنیم. برای هر مجموعه correlation، با فرض نرمال بودن correlationها، threshold را برابر میانگین به علاوه ۳ برابر انحراف معیار ادر نظر می‌گیریم. سپس تمام پیکسل‌ها را پیمایش کرده و آن‌هایی که correlation بیشتر از threshold داشتند را به عنوان ICهای یافت شده در PCB در نظر می‌گیریم. پس correlation این نقاط را صفر می‌کنیم تا دوباره آن‌ها در پیمایش آن‌ها را به عنوان IC دیگر حساب نکنیم و ایندکس گوشه را ذخیره می‌کنیم. پس از پایان پیمایش مستطیل‌هایی به طول و عرض تصویر IC با گوشه‌های یافت شده ترسیم می‌کنیم.

```
    for i = 1:(width - box_width + 1)
        for j = 1:(height - box_height + 1)
            subpcb = pcb(i:i + box_width - 1, j:j + box_height - 1,:);
            norm1=sum(subpcb.^2,"all");
            norm=sqrt(norm1*norm2);
            cor = subpcb .* ic ./ norm;
            cor=sum(cor,'all');
            correlation(i,j)=cor;
            rot_norm=sqrt(norm1*rot_norm2);
            cor = subpcb .* rot_ic ./ rot_norm;
            cor=sum(cor,'all');
            rot_correlation(i,j)=cor;
        end
    end
    thresh=mean(correlation(:))+3*std(correlation(:));
    rot_thresh=mean(rot_correlation(:))+3*std(rot_correlation(:));
    for i = 1:(width - box_width + 1)
        for j = 1:(height - box_height + 1)
            if(correlation(i,j)>thresh)
                img = insertShape(img, 'Rectangle', [j,i,box_height-1,box_width-1], 'Color', 'b', 'LineWidth', 3);
                correlation(i:i + box_width - 1, j:j + box_height - 1,:)=0;
            end
            if(rot_correlation(i,j)>rot_thresh)
                img = insertShape(img, 'Rectangle', [j,i,box_height-1,box_width-1], 'Color', 'b', 'LineWidth', 3);
                rot_correlation(i:i + box_width - 1, j:j + box_height - 1,:)=0;
            end
        end
    end
end
```

اسکریپت این بخش:

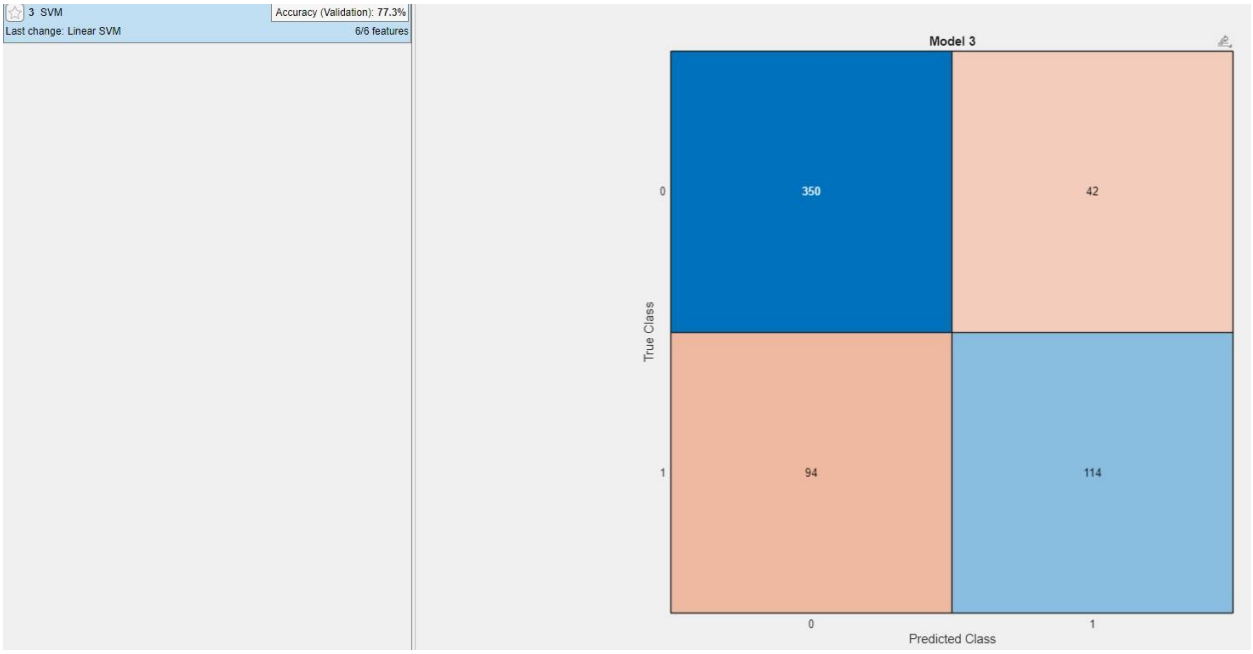
```
clc;clear;[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose the PCB image');
pcb=imread([path file]);
[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose the IC image');
ic=imread([path file]);
img=ICrecognition(pcb,ic);
imshow(img);
```

تصویر خروجی:



بخش سوم)

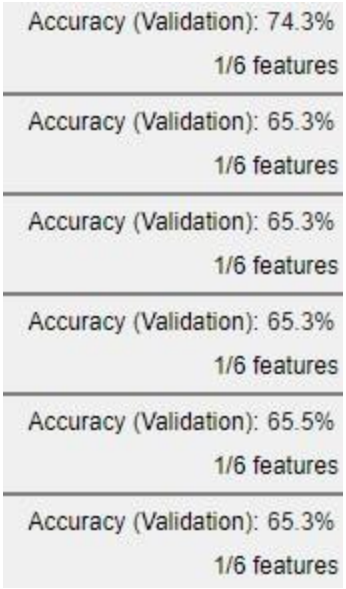
۱- با مدل linear svm به دقت 77.3 درصد با confusion matrix زیر می‌رسیم:



۲- در کل ۶ فیچر وجود دارد:

1	☑	Glucose
2	☑	BloodPressure
3	☑	SkinThickness
4	☑	Insulin
5	☑	BMI
6	☑	Age

۶ مدل مختلف آموزش می‌دهیم به طوری که به ترتیب یکی از ۶ فیچر بالا در هر مدل فعال باشد:



می‌توان دید glucose بیشترین تاثیر را دارد و از طرفی بقیه فیچرها تاثیر کمتر و تقریبا یکسانی دارند.

۳- مدل آموزش دیده شده با همه فیچرها را وارد اسکریپت p3_3 می‌کنیم و می‌توانیم مشاهده کنیم که دقت این مدل روی داده آموزش 77.5 درصد است که برابر دقت قبل است.

```
clc;
predictions=TrainedModel.predictFcn(diabetestraining);
ytrain=diabetestraining.label;
txt=sprintf('The train accuracy is %f',sum(ytrain==predictions)/length(ytrain));
disp(txt);
```

```
Command Window
The train accuracy is 0.775000
```

۴- حال دقت را روی داده تست به دست می‌آوریم. می‌توان دید که به دقت مشابهی یعنی 78% می‌رسیم پس مدل به خوبی آموزش داده و تعمیم‌پذیری بالایی دارد.

```
clc;
predictions=TrainedModel.predictFcn(diabetesvalidation);
ytest=diabetesvalidation.label;
txt=sprintf('The test accuracy is %f',sum(ytest==predictions)/length(ytest));
disp(txt);
```

```
Command Window
The test accuracy is 0.780000
```