

پروژه درس بینایی کامپیوتر

مدرس : استاد جمشید پیرگری

گروه:

اشکان شیخ انصاری

ابوالفضل عابدینی

دسته بندی اعداد و کاراکترهای فارسی

در این پروژه قصد داریم با استفاده از روش های پردازش تصویر و یادگیری ماشین، پلاک خودروها را شناسایی نمای م و در نهایت متن پلاک در خرجی چاپ شود. بنابر این بر اساس آنچه در کلاس آموزش داده شد. ابتدا باید پردازش های مورد نیاز بر روی تصویر خودرو انجام شود و سپس محل پلاک شناسایی شود. در مرحله بعد با استفاده از روش های سگمنت بندی، بخش های مختلف پلاک از هم جدا شوند. سپس به منظور شناسایی متن پلاک باید از هر بخش ویژگی های مختلف استخراج و عمل دسته بندی با حداکثر دقت انجام شود.

برنامه شما باید شامل توابع زیر باشد:

1- تابع تشخیص محل پلاک

ورودی این تابع تصویر دوربین (شامل یک یا چند خودرو) و خروجی آن تصویر ورودی که در دور پلاک ها مستطیل سبز کشیده شده باشد و ذخیره پلاک ها در محل مناسب می باشد. خروجی نمونه در شکل 1 نشان داده شده است.



شکل 1. خروجی تابع تشخیص محل پلاک

مراحل تشخیص محل پلاک 16 تصویر ماشین فایل cars:

- 1- ابتدا روی تصاویر پیش پردازش انجام داده تا عملیات حذف کانتور آسان تر شود.
- 2- با توجه به ویژگی پلاک از قبیل محیط و مساحت و کانتورهایی که ویژگی آن ها شبیه به پلاک نباشد را حذف می کنیم.
- 3- با توجه به ویژگی مینیمم فاصله ی بین کاراکترهای پلاک و اختلاف زاویه بین آن ها و شبیه پلاک نباشد را حذف می کنیم.

4-کانتورهای باقی مانده پلاک های ما بوده و ما بتید محل پلاک را در عکس پیدا کرده و در فایل plates-car-result ذخیره می کنیم تا بتوانیم از آن مرحله segmentation را انجام بدهیم.
خروجی چند نمونه به صورت زیر است:

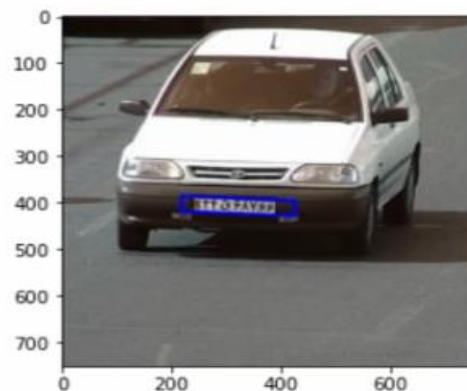
```
for j in range(len(listOfMatchingChars[i])):
    for k in range(len(listOfMatchingChars[i][j])):
        possiblePlate = extractPlate(data_car['data'][i],listOfMatchingChars[i][j])
        licPlate=possiblePlate
        drawRedRectangleAroundPlate(data_car['data'][i],licPlate)
        cv2.imwrite(path + "frame"+str(o)+".jpg",licPlate.imgPlate)
        o=o+1
plt.imshow(data_car['data'][9])
```

Out[303]: <matplotlib.image.AxesImage at 0x1b29f6d5f70>

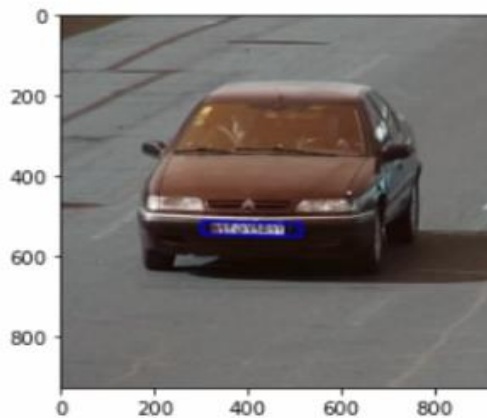


In [305]: plt.imshow(data_car['data'][11])

Out[305]: <matplotlib.image.AxesImage at 0x1b29f267220>



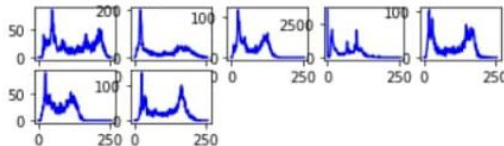
```
In [317]: plt.imshow(data_car['data'][7])
Out[317]: <matplotlib.image.AxesImage at 0x1b29fe9a2e0>
```



2- تابع بخش بندی پلاک

در این بخش شما باید با استفاده از روش های مختلف عمل بخش بندی پلاک و جداسازی کاراکتر ها را انجام دهید. ورودی این بخش تصویر پلاک و خروجی آن بخش های جدا شده پلاک به همراه نرخ صحت روش بخش بندی می باشد. در این قسمت شما باید جدول زیر را تکمیل کنید. برای این جدول باید از 10 تصویر پلاک استفاده کنید و میانگین دقت این 10 تصویر در جدول زیر نوشته شود. روش حد آستانه گذاری و روش مبتنی بر هیستوگرام: ابتدا ما 8 تا از پلاک های استخراج شده از فایل عکس ماشین را انتخاب کرده و نمودار هیستوگرام به شکل زیر است:

```
In [71]: sampel=[12,7,8,15,6,20,5,8]
j=551
for i in range(len(img_seg)):
    if i in sampel:
        hist=cv2.calcHist([img_seg[i]],[0],None,[256],[0,256])
        plt.subplot(j),plt.plot(hist,color="b")
        j=j+1
```



می بینیم که تقریباً همه نمودارها یک پیک بین 40 تا 60 پیکسل را دارند. بنابراین می توان نتیجه گرفت با استفاده از عدد پیک نمودار حد آستانه را مشخص کرد. در روش مبتنی بر حد آستانه threshold را 50 در نظر می گیریم. خروج سگمنت بندی از 10 تا نمونه به صورت زیر است.

2

```
2]: plt.imshow(img_seg_thresh[1],cmap='gray')
2]: <matplotlib.image.AxesImage at 0x1e69df09ac0>
```

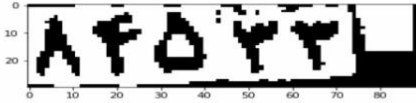
1

```
In [72]: plt.imshow(img_seg_thresh[2],cmap='gray')
Out[72]: <matplotlib.image.AxesImage at 0x1e6a159cd60>
```

4

```
In [80]: plt.imshow(img_seg_thresh[10], cmap='gray')
```

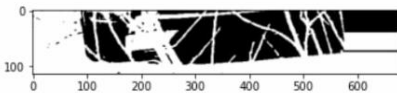
Out[80]: <matplotlib.image.AxesImage at 0x1e6a29d8ee0>



3

```
In [78]: plt.imshow(img_seg_thresh[8], cmap='gray')
```


Out[78]: <matplotlib.image.AxesImage at 0x1e6a1948220>



6

```
In [84]: plt.imshow(img_seg_thresh[14], cmap='gray')
```

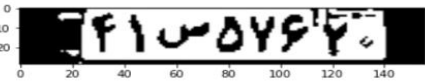
Out[84]: <matplotlib.image.AxesImage at 0x1e6a1681e20>



5

```
In [83]: plt.imshow(img_seg_thresh[13], cmap='gray')
```

Out[83]: <matplotlib.image.AxesImage at 0x1e69e0620d0>



8

```
In [88]: plt.imshow(img_seg_thresh[19], cmap='gray')
```


Out[88]: <matplotlib.image.AxesImage at 0x1e6a2b143a0>



7

```
In [87]: plt.imshow(img_seg_thresh[17], cmap='gray')
```


Out[87]: <matplotlib.image.AxesImage at 0x1e6a16f5dc0>



10

```
In [98]: plt.imshow(img_seg_thresh[28], cmap='gray')
```


Out[98]: <matplotlib.image.AxesImage at 0x1e6a2f16dc0>



9

```
In [94]: plt.imshow(img_seg_thresh[24], cmap='gray')
```

Out[94]: <matplotlib.image.AxesImage at 0x1e6a2d7e7f0>




روش مبتنی بر کانتور: ما در این روش سعی بر این داریم تمام کاراکترهای پلاک را جدا کرده و در فایل سگمنت ذخیره کنیم تا بتوانیم از آن ها ویژگی استخراج کنیم.
خروجی 10 sample به صورت زیر است:

12

```
In [107]: plt.imshow(img_segmentation_contour[24], cmap='gray')
```


Out[107]: <matplotlib.image.AxesImage at 0x1e6a32f78b0>



11

```
In [106]: plt.imshow(img_segmentation_contour[28], cmap='gray')
```


Out[106]: <matplotlib.image.AxesImage at 0x1e6a3294880>



14

```
In [110]: plt.imshow(img_segmentation_contour[18], cmap='gray')
```


Out[110]: <matplotlib.image.AxesImage at 0x1e6a34383a0>



13

```
In [108]: plt.imshow(img_segmentation_contour[19], cmap='gray')
```


Out[108]: <matplotlib.image.AxesImage at 0x1e6a3361be0>



16

```
In [112]: plt.imshow(img_segmentation_contour[13], cmap='gray')
```


Out[112]: <matplotlib.image.AxesImage at 0x1e6a34fef0>

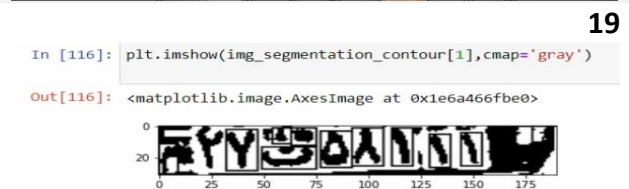
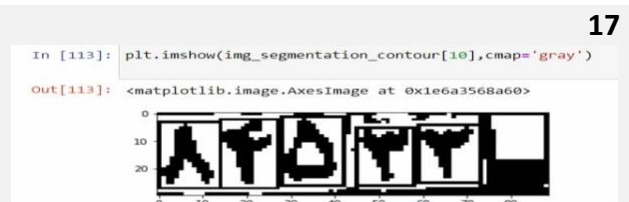
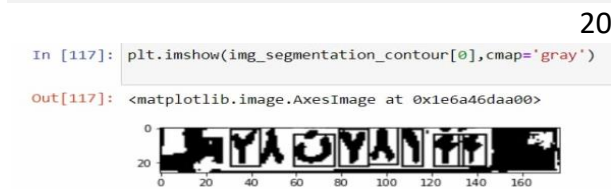
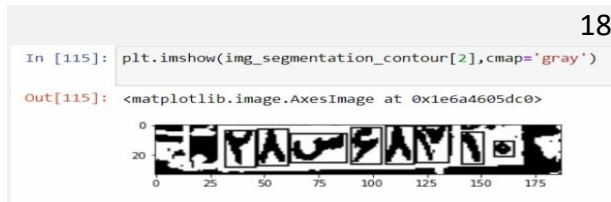


15

```
In [111]: plt.imshow(img_segmentation_contour[14], cmap='gray')
```

Out[111]: <matplotlib.image.AxesImage at 0x1e6a349e790>



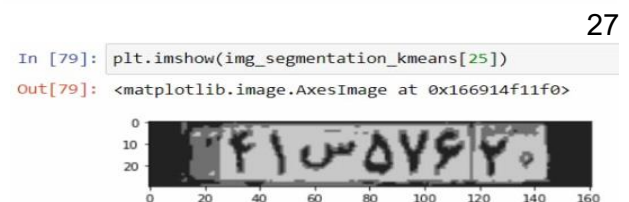
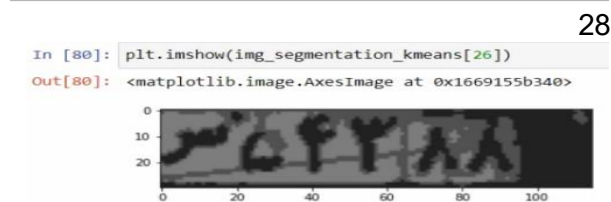
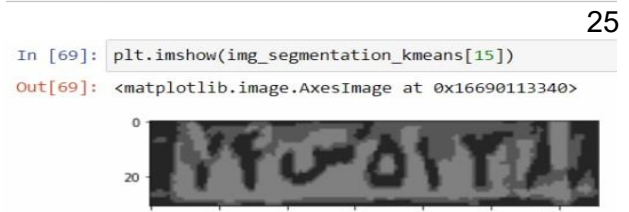
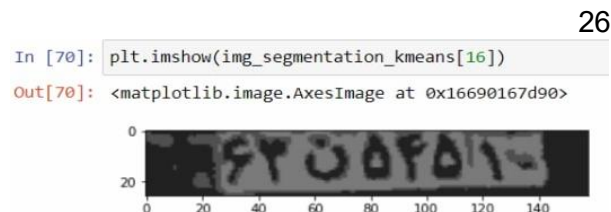
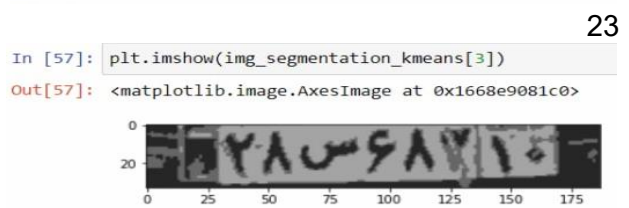
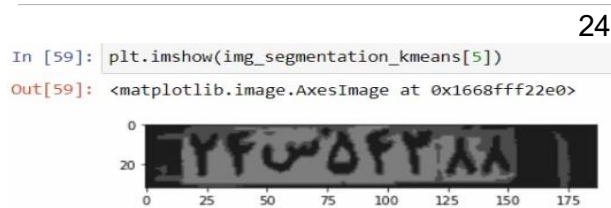
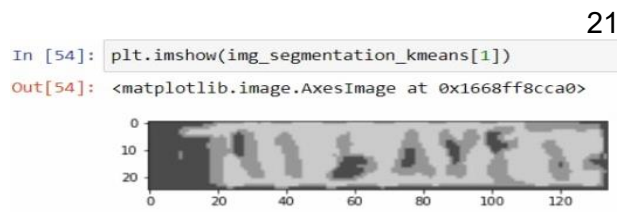
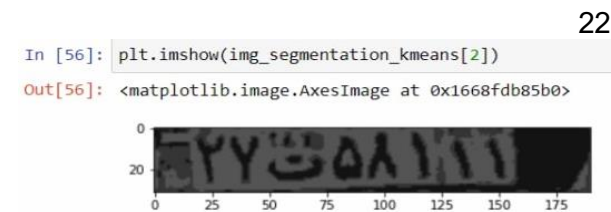


روش پیشنهادی: بخش بندی با استفاده از k-means
مراحل کار به این صورت است که:

1- باید مرحله پیش پردازش انجام دهیم و تصاویر را به فضای rgb ببریم.
- shape تصویر را باید تغییر دهیم و یک بردار 2 بعدی تبدیل کنیم. برای مثال اگر تصویری با shape (100,100,3) باشد تبدیل به (1000,3) می شود.

2- تابع segmentation-K means: این تابع دو آرگومان k و image original به عنوان ورودی می گیرد. مرحله 1 و 2 در این تابع انجام می گیرد و سپس الگوریتم k-means را پیاده سازی می کنیم و خروجی ما یک تصویر سگمنت بندی شده با استفاده از روش k-means است.

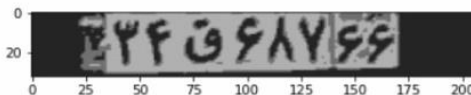
خروجی 10 نمونه به صورت زیر است:




```
In [94]: plt.imshow(img_segmentation_kmeans[22])
Out[94]: <matplotlib.image.AxesImage at 0x16691ae2760>
```



```
In [92]: plt.imshow(img_segmentation_kmeans[19])
Out[92]: <matplotlib.image.AxesImage at 0x16691a0dfd0>
```



جدول 1. بررسی دقت روش های مختلف بخش بندی

روش بخش بندی	دقت بخش بندی کاراکترهای پلاک	تعداد بخش های غیر کاراکتری
روش حد آستانه گذاری و روش مبتنی بر هیستوگرام	دقت بخش بندی کل $= \frac{58}{77} \times 100$ برابر با 75%	پلاک 1: 0 بخش غیر کاراکتری بخش بندی شده است. پلاک 4: 0 بخش غیر کاراکتری بخش بندی شده است. پلاک 5: 0 بخش غیر کاراکتری بخش بندی شده است. پلاک 6: 0 بخش غیر کاراکتری بخش بندی شده است.
	پلاک 1: $\frac{7}{8} \times 100$ برابر با 87.5%	
	پلاک 2: $\frac{8}{8} \times 100$ برابر با 100%	
	پلاک 3: $\frac{0}{8} \times 100$ برابر با 0 (داده پرت)	
	پلاک 4: $\frac{5}{5} \times 100$ برابر با 100%	پلاک 7: 0 بخش غیر کاراکتری بخش بندی شده است.
	پلاک 5: $\frac{7}{8} \times 100$ برابر با 87.5%	پلاک 8: 0 بخش غیر کاراکتری بخش بندی شده است.
	پلاک 6: $\frac{4}{8} \times 100$ برابر با 50%	پلاک 9: 0 بخش غیر کاراکتری بخش بندی شده است.
	پلاک 7: $\frac{8}{8} \times 100$ برابر با 100%	پلاک 10: 0 بخش غیر کاراکتری بخش بندی شده است.
	پلاک 8: $\frac{8}{8} \times 100$ برابر با 100%	
	پلاک 9: $\frac{7}{8} \times 100$ برابر با 87.5%	
	پلاک 10: $\frac{4}{8} \times 100$ برابر با 50%	پلاک 2: 4 بخش غیر کاراکتری بخش بندی شده است. پلاک 3: بیش از 5 بخش غیر کاراکتری بخش بندی شده است.
		20 تا از کاراکترها را به صورت خوب سگمنت بندی نکرده و یک دیتای پرت داریم .
روش مبتنی بر کانتورها	دقت بخش بندی کل $= \frac{65}{77} \times 100$ برابر با 84%	پلاک 11: 0 بخش غیر کاراکتری بخش بندی کرده است. پلاک 12: 0 بخش غیر کاراکتری بخش بندی کرده است.
	پلاک 11: $\frac{7}{8} \times 100$ برابر با 87.5%	پلاک 13: 1 بخش غیر کاراکتری بخش بندی کرده است.
	پلاک 12: $\frac{8}{8} \times 100$ برابر با 100%	پلاک 14: 0 بخش غیر کاراکتری بخش بندی کرده است.
	پلاک 13: $\frac{5}{8} \times 100$ برابر با 62.5%	پلاک 15: 1 بخش غیر کاراکتری بخش بندی کرده است.
	پلاک 14: $\frac{8}{8} \times 100$ برابر با 100%	

<p>پلاک 16: 1 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 17: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 18: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 19: 2 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 20: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>در مجموع 5 بخش غیرکاراكتري سگمنت بندى كرده است.</p>	<p>پلاک 15: $100 \times \frac{3}{8}$ برابر با 37.5%</p> <p>پلاک 16: $100 \times \frac{7}{8}$ برابر با 87.5%</p> <p>پلاک 17: $100 \times \frac{5}{5}$ برابر با 100%</p> <p>پلاک 18: $100 \times \frac{8}{8}$ برابر با 100%</p> <p>پلاک 19: $100 \times \frac{7}{8}$ برابر با 87.5%</p> <p>پلاک 20: $100 \times \frac{7}{8}$ برابر با 87.5%</p>	
<p>پلاک 21: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 22: 2 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 23: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 24: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 25: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 26: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 27: 1 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 28: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 29: 0 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>پلاک 30: 1 بخش غیرکاراكتري بخش بندى كرده است.</p> <p>در مجموع 4 بخش غیرکاراكتري سگمنت بندى كرده است.</p>	<p>دقت بخش بندى كل $= \frac{63}{77} \times 100$ برابر با 81%</p> <p>پلاک 21: $100 \times \frac{0}{8}$ برابر با 0%</p> <p>پلاک 22: $100 \times \frac{8}{8}$ برابر با 100%</p> <p>پلاک 23: $100 \times \frac{7}{8}$ برابر با 87.5%</p> <p>پلاک 24: $100 \times \frac{8}{8}$ برابر با 100%</p> <p>پلاک 25: $100 \times \frac{5}{8}$ برابر با 62.5%</p> <p>پلاک 26: $100 \times \frac{7}{8}$ برابر با 87.5%</p> <p>پلاک 27: $100 \times \frac{8}{8}$ برابر با 100%</p> <p>پلاک 28: $100 \times \frac{4}{5}$ برابر با 80%</p> <p>پلاک 29: $100 \times \frac{8}{8}$ برابر با 100%</p> <p>پلاک 30: $100 \times \frac{8}{8}$ برابر با 100%</p>	<p>روش پیشنهادی: روش مبتنى بر k-means</p>

نتیجه گیری: با توجه به دقت كل 3 روش نتیجه می گیریم كه دقت روش مبتنى بر كانتور < روش مبتنى بر k-means > روش مبتنى بر هیستوگرام و حد آستانه گذارى. بنابراین كاراكترهاى پلاک را به روش كانتور سگمنت بندى می كنیم.

خروجی این مرحله در شكل 2 نشان داده شده است. در این شكل در سطر شماره 1، تعداد كاراكترهاى مربوط به پلاک همگى شناسایی شده اند. بنابر این دقت بخش بندى كاراكترهاى پلاک 100 می باشد. اما آخرین بخش جدا شده مربوط به پلاک نمى باشد بنابر این تعداد بخش هاى غیر كاراكترى يك می باشد. در سطر 2، همه كاراكترها به درستی بخش بندى

شده اند بنابر این دقت بخش بندی کارکترها ی پلاک 100 می باشد. در این مورد تعداد بخش های غیر کاراکتری صفر می باشد. در این قسمت هدف این است که ابتدا همه کاراکترهای پلاک بخش بندی شوند و هیچ بخشی که به عنوان شماره پلاک نیست جداسازی نشود.

1			
2			

شکل 2. خروجی تابع بخش بندی

- تابع استخراج ویژگی

در این مرحله باید از بخش های جدا شده از پلاک ویژگی های مناسب را استخراج نمائید. ورودی ها این تابع بخش های مختلف یک پلاک و خروجی آن بردار ویژگی استخراج شده می باشد. در این قسمت شما باید جدول زیر را تکمیل کنید.

ما در این فاز ابتدا کاراکترهای سگمنت بندی شده ی پلاک های استخراج شده را ویژگی استخراج کرده ایم (feature) این ویژگی ها عبارتند از :تعداد پیکسل های سیاه و سفید ، LBP ، corner shi-tomasi، corner harris ، SIFT و SOBEL و همین ویژگی ها را در دیتای موجود در فایل Data persian بردار ویژگی استخراج کرده ایم.

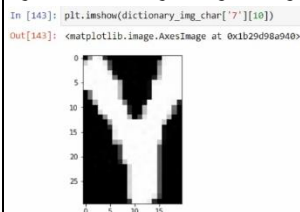
جدول 2. بررسی روش های مختلف استخراج ویژگی

نوع ویژگی	طول بردار ویژگی	توضیحات
-----------	-----------------	---------

ابتدا ما باید عمل پیش پردازش را روی داده ها انجام داده و آن ها را به فضای باینری ببریم تا بتوانیم پیکسل های سیاه و سفید را استخراج کنیم.

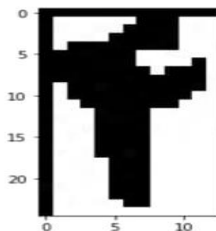
بعد از مرحله پیش پردازش ویژگی استخراج شده به صورت یک vector به طول $(h \times w)$ می باشد.

معایب: این استخراج ویژگی خوب نبوده برای مثال اگر



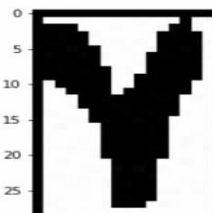
ما بردار ویژگی های این عدد 7 را به مدل دسته بند بدهیم ، کاراکتر عدد 7 که به صورت پس زمینه سیاه باشد را به خوبی پیش بینی نمی کند زیرا بردار ویژگی آن ها متفاوت است.

```
In [21]: plt.imshow(data_char_seg['data'][5])
data_char_seg['data'][5].shape
Out[21]: (25, 13, 3)
```



```
In [142]: print(featurevector_pixel_black_white[5])
[[ 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255
 255 255 0 0 0 255 255 255 0 255 255 255 255 0 0 0
 255 255 255 0 255 255 255 255 0 0 0 0 255 255 255 0 255
 0 0 0 0 0 0 0 0 255 255 255 0 0 0 0 0 0
 255 255 255 255 255 255 0 0 0 0 0 0 255 255 255 255 0
 255 0 0 0 0 0 0 0 0 255 0 0 0 255 0 0 0 0
 0 0 0 0 0 0 0 0 255 0 255 0 0 0 0 0 0 0
 0 0 255 0 255 0 0 0 0 0 0 0 0 0 0 255 255 0
 255 255 0 0 0 0 0 0 255 255 255 0 255 255 255 0 0
 0 0 255 255 255 255 0 255 255 255 0 0 0 0 255 255 255
 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255
 0 255 255 255 255 0 0 255 255 255 255 255 0 255 255 255
 0 0 0 255 255 255 255 255 0 255 255 255 255 0 0 255 255
 255 255 255 0 255 255 255 255 0 0 255 255 255 255 255 0 255
 255 255 255 0 0 0 255 255 255 255 0 255 255 255 255 255
 0 255 255 255 255 0 255 255 255 255 255 255 255 255 255 255]]
```

```
In [134]: plt.imshow(data_char_seg['data'][20])
data_char_seg['data'][18].shape
Out[134]: (21, 12, 3)
```

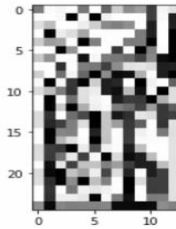


```
In [147]: print(featurevector_pixel_black_white[20])
[[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255
 255 255 255 255 255 255 255 255 255 255 0 255 255 0 0 0 0
 255 255 255 255 255 255 255 255 0 0 255 255 0 0 0 0 0 255
 255 255 255 255 255 0 0 0 0 255 0 0 0 0 0 255 255 255
 255 255 255 0 0 0 0 255 0 0 0 0 0 255 255 255 255 0
 255 0 0 0 255 0 0 0 0 0 0 0 255 255 255 255 0 0
 0 0 255 0 0 0 0 0 0 255 255 255 255 0 0 0 0
 0 255 0 0 0 0 0 0 0 255 255 255 0 0 0 0 0 255
 0 0 0 0 0 0 255 255 0 0 0 0 0 0 255 0 255
 0 0 0 0 255 255 0 0 0 0 0 255 255 0 255 255 255 0
 0 0 0 0 0 0 255 255 255 0 255 255 255 0 0 0 0
 0 0 0 0 255 255 255 0 255 255 255 255 0 0 0 0 0
 0 0 255 255 255 0 255 255 255 255 255 0 0 0 0 0 0
 255 255 255 0 255 255 255 255 255 255 0 0 0 0 0 255 255
 255 255 0 255 255 255 255 255 0 0 0 0 0 255 255 255 255
 255 0 0 0 255 255 255 255 255 0 255 255 255 255 255 0
 0 0 255 255 255 255 255 0 255 255 255 255 255 0 0 0
 0 255 255 255 255 0 255 255 255 255 255 0 0 0 0 255]]
```

ویژگی های آماری تعداد پیکسل های سیاه و سفید ...

عدد 4:

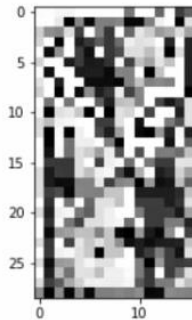
```
In [137]: plt.imshow(img_lbp[5], cmap='gray')
Out[137]: <matplotlib.image.AxesImage at 0x1b29c81b1c0>
```



```
In [147]: featurevector_lbp[5]
Out[147]: array([[125, 254, 247, 255, 119, 255, 103, 207, 143, 159, 63, 247, 227,
254, 255, 251, 0, 250, 0, 242, 255, 255, 255, 62, 251, 32,
223, 191, 255, 255, 239, 207, 139, 129, 144, 255, 32, 255, 34,
255, 0, 239, 223, 167, 255, 255, 255, 255, 124, 32, 255, 2,
205, 158, 167, 255, 3, 129, 255, 255, 124, 124, 2, 255, 2,
255, 255, 3, 149, 255, 255, 255, 56, 252, 4, 130, 253, 2,
201, 148, 255, 255, 17, 248, 124, 120, 108, 70, 139, 13, 2,
255, 255, 49, 255, 126, 120, 64, 144, 228, 231, 223, 31, 35,
209, 255, 112, 248, 104, 0, 142, 11, 16, 177, 255, 30, 0,
251, 0, 248, 56, 4, 207, 31, 39, 255, 0, 244, 254, 98,
221, 190, 125, 52, 226, 239, 31, 16, 189, 18, 249, 0, 224,
255, 0, 252, 112, 224, 245, 255, 30, 60, 58, 4, 178, 225,
253, 2, 201, 0, 176, 241, 252, 126, 108, 68, 158, 57, 96,
221, 50, 255, 111, 73, 0, 200, 24, 48, 231, 223, 56, 64,
223, 40, 64, 128, 223, 3, 199, 159, 56, 96, 255, 52, 227,
255, 4, 135, 131, 255, 34, 199, 255, 60, 242, 253, 122, 225,
255, 6, 183, 227, 253, 0, 195, 253, 62, 251, 253, 254, 227,
221, 62, 57, 0, 237, 82, 243, 253, 30, 191, 255, 255, 227,
```

عدد 7:

```
Out[148]: <matplotlib.image.AxesImage at 0x1b29d9f82b0>
```



```
In [149]: featurevector_lbp[20]
Out[149]: array([[253, 254, 247, 247, 255, 255, 255, 255, 119, 255, 255, 103,
255, 63, 99, 255, 255, 235, 0, 136, 8, 128, 128, 136, 8,
8, 128, 242, 255, 0, 226, 207, 143, 135, 223, 63, 255, 127,
55, 255, 103, 231, 207, 155, 173, 222, 163, 215, 255, 3, 255,
30, 56, 92, 56, 112, 240, 225, 231, 255, 39, 255, 35, 255,
12, 146, 255, 62, 16, 254, 56, 80, 224, 224, 193, 237, 2,
221, 34, 213, 255, 255, 28, 60, 26, 28, 52, 250, 65, 193,
163, 229, 195, 255, 34, 255, 16, 255, 255, 28, 30, 30, 10,
133, 131, 255, 1, 195, 255, 255, 32, 229, 255, 33, 255, 255,
14, 14, 14, 7, 131, 197, 130, 139, 129, 255, 32, 209, 255,
2, 255, 12, 142, 31, 55, 239, 67, 135, 255, 255, 255, 255,
2, 255, 117, 255, 117, 255, 255, 95, 32, 192, 199, 131, 255,
72, 24, 60, 226, 249, 0, 224, 234, 201, 0, 255, 0, 227,
215, 255, 69, 255, 255, 32, 227, 221, 190, 243, 247, 255, 255,
61, 62, 227, 255, 17, 255, 255, 124, 82, 227, 223, 0, 251,
0, 240, 255, 76, 30, 163, 255, 10, 1, 152, 36, 250, 33,
255, 2, 253, 114, 241, 240, 255, 127, 3, 255, 255, 255, 255,
18, 253, 98, 253, 18, 249, 112, 224, 208, 160, 238, 82, 255,
```

ویژگی های LBP

مراحل الگوریتم به این صورت است که 8 تا از همسایه های تمام پیکسل های تصویر را در نظر می گیریم به طوری که اگر همسایگان از آن پیکسل مشخص کوچک تر باشند 1 و اگر بزرگ تر باشند صفر می باشند.

این 8 عدد به صورت یک vector می باشد که عدد دسیمال آن ، مقدار ناحیه آن پیکسل است.

معایب: این استخراج ویژگی منعطف نبوده و در برابر تغییر سایز عکس تغییر می‌کند. به این صورت که اگر تصویری بزرگ تر شود ناحیه‌هایی که گوشه هستند را به عنوان گوشه انتخاب نمی‌کند.

[illegible]

A 19x19 grid representing a Go board. The grid is mostly black, with a few white squares forming a pattern in the center. The axes are labeled from 0 to 18.

```
out[199]: array([[0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 1,  
1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,  
0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 255,  
0, 255, 0, 255, 0, 255, 0, 255, 1, 1, 1, 0, 0,  
255, 252, 252, 252, 255, 255, 255, 0, 255, 0, 255,  
0, 255, 0, 255, 0, 255, 2, 2, 2, 2, 2, 2, 1,  
1, 1, 2, 2, 2, 0, 255, 0, 255, 255, 255,  
251, 251, 251, 0, 0, 0, 255, 255, 254, 254, 254,  
255, 255, 255, 4, 4, 4, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 251, 251, 251, 255, 255, 255, 255, 255,  
255, 1, 1, 1, 253, 253, 253, 255, 255, 255, 255,  
0, 0, 0, 2, 2, 0, 0, 0, 4, 4, 0, 0,  
0, 255, 255, 255, 252, 252, 252, 255, 255, 0, 0,  
0, 255, 255, 255, 255, 255, 252, 252, 252, 252, 3, 3,  
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 2,  
2, 2, 255, 255, 252, 252, 252, 1, 1, 1, 254, 254,  
254, 252, 252, 2, 2, 2, 1, 1, 1, 0, 0, 0,  
2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 255,
```

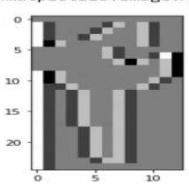
ویژگی های گوشه

هدف از استخراج ویژگی
sobel این است که نقاط لبه را
به عنوان ویژگی به مدل دسته
بند بدهیم تا متوجه شویم ویژگی
خوبی است یا خیر.

ما در این استخراج ویژگی ابتدا
تمامی تصویرها را به فضای
خاکستری می بریم و عملیات لبه
یابی sobel را برای تمامی
تصاویر انجام داده ایم. ویژگی
استخراج شده به صورت یک
vector به طول $(1 \times h \times w)$
می باشد.

```
In [183]: plt.imshow(img_sobel[5], cmap='gray')
```

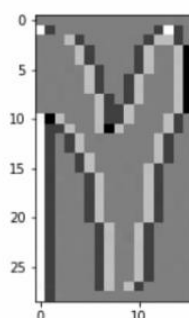
Out[183]: <matplotlib.image.AxesImage at 0x1b29df3ecd0>



```
In [213]: featurevector_sobel[5]
```

Out[213]: array([[0., -1., 1., 0., 1., -2., 2., -1., 0.,
-1., 2., -2., 2., 510., -255., 0., 0., -1.,
2., -256., 255., 0., 255., -256., 2., -2., 502.,
-249., -2., -4., 7., -253., 246., 5., -2., 255.,
-256., 5., -6., 510., -255., -3., 6., -258., 255.,
0., 0., 3., 249., -253., 2., -2., 502., -504.,
253., 2., -1., -4., 3., 3., -6., 258., -255.,
-1., 2., 0., 0., 2., -4., 2., 0., 255.,
-255., -3., 5., -2., 1., -2., 0., -1., 1.,
2., -3., 2., 250., -249., -2., 0., -253., 506.,
-506., 0., 1., -2., 2., -2., 1., 0., 253.,
-505., 251., 1., 255., -510., -4., 5., -6., 3.,
0., 3., -6., 4., -2., 1., 0., 253., -506.,
506., -507., 254., 5., -9., 3., 1., 0., 0.,
0., 4., 246., -500., 510., -508., 251., 2., 1.,
-2., 3., -4., 2., 1., 251., -250., -4., 506.,
-251., -257., 256., -1., 1., -3., 2., 3., 248.,
-251., 257., -2., 508., -255., 1., -250., 246., 4.,
1., 252., -253., 1., -2., 2., -2., 504., -249.,
-3., -255., 258., -4., -1., 257., -255., -1., 1.,

```
Out[211]: <matplotlib.image.AxesImage at 0x1b29dbe9400>
```



```
In [212]: featurevector_sobel[20]
```

Out[212]: array([[-2., 1., 1., -2., 1., 0., 0., 0., 2.,
-4., 2., 0., 0., 4., -8., 8., 508., -254.,
0., -2., 4., -2., 0., 0., -3., 6., -3.,
0., -255., 507., -250., -4., 2., -2., 3., 251.,
-253., 0., -3., 6., -3., 0., -4., -245., 250.,
250., -250., -2., -2., 4., -6., 3., 251., -247.,
-4., 0., 0., -2., -251., 255., -4., 2., 255.,
-510., 4., -2., -2., 2., 255., -257., 2., 2.,
-4., 4., -256., 253., 1., 3., 248., -502., -2.,
1., 5., -10., 5., 255., -255., -1., 2., -5.,
-245., 247., 6., -8., 259., -510., 6., -6., 3.,
4., -8., 256., -249., -3., -1., -253., 256., -2.,
-2., 2., 254., -508., -2., 4., -6., 3., 1.,
253., -254., 0., -2., -250., 251., 1., 1., -2.,
256., -510., -4., 5., -6., 5., -2., -2., 257.,
-257., 4., -257., 255., 0., 0., 0., 255., -510.,
4., -4., 3., -2., 1., 0., 255., -255., -255.,
258., -6., 4., 0., -3., 257., -510., 498., -501.,
252., 0., 1., -3., 257., -255., -252., 249., 4.,
-2., 1., 255., -256., 2., 510., -255., -254., 253.,

ویژگی SOBEL

<p>استخراج ویژگی SIFT به این صورت است که نقاط کلیدی (keypoint) ها را به عنوان ویژگی استفاده می کنیم</p> <p>نقاط کلیدی ویژگی هایی از نظیر مختصات (x,y) ، اندازه همسایگی معنی دار ، زاویه ای جهت آن را مشخص می کند و قدرت نقاط کلیدی را مشخص می کند را به عنوان ویژگی استخراج می کنیم.</p>	<pre>In [259]: plt.imshow(img_sift[5])</pre> <p>Out[259]: <matplotlib.image.AxesImage at 0x1b29f9554f0></p>  <pre>In [260]: plt.imshow(img_sift[20])</pre> <p>Out[260]: <matplotlib.image.AxesImage at 0x1b29f8ba3d0></p> 	<p>ویژگی های SIFT</p>
<p>مراحل استخراج ویژگی ARR:</p> <p>1- برای استخراج ویژگی ARR تصویر را به فضای باینری می بریم تا پیکسل های تصویر 0 یا 255 شوند.</p> <p>2- تابع crop_image: این تابع تصویر باینری شده را می گیرد و به 12 قسمت تبدیل می کند بطوریکه قسمت های تصویر height بر 4 تقسیم شده و width بر 3 تقسیم شده است و خروجی لیستی از 12 قسمت تصویر است.</p> <p>3- تابع calculation_white_pixel: در این تابع لیستی از 12 قسمت مرحله قبل را به عنوان ورودی را دریافت کرده و نسبت پیکسل های سفید را به تعداد کل پیکسل های آن ناحیه تقسیم می کنیم. عدد بدست آمده ویژگی آن ناحیه است.</p>	<p>برای مثال عدد 5:</p> <pre>In [202]: plt.imshow(img_datapersian_black_white[5][2], cmap='gray')</pre> <p>Out[202]: <matplotlib.image.AxesImage at 0x2315a355340></p>  <p>توجه: در ویژگی ARR تصویر به 12 قسمت تقسیم می شود که هر 12 قسمت را به دست آورده ایم اما در خروجی پایتون فقط 9 قسمت نمایش داده می شود (پایتون بیشتر از 9 قسمت نمایش نمی دهد). بنابراین خروجی 12 قسمت تصویر به صورت زیر است:</p>  <pre>In [211]: def calculation_white_pixel(listt): h=0 m=0 for s in listt[0]: if s==255: h=h+1 return h</pre> <p>بردار ویژگی عدد 5 به صورت زیر است که شامل 12 قسمت است.</p> <p>هر قسمت محاسبه شده تعداد پیکسل های سفید نسبت به کل پیکسل های آن ناحیه است.</p>	<p>ویژگی های ARR</p>

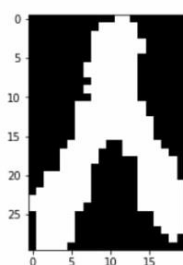
```
In [222]: final_featurevector_arr_datapersian[5][2]
```

```
Out[222]: [0.23809523809523808,
0.7551020408163265,
0.061224489795918366,
0.5,
0.4107142857142857,
0.5357142857142857,
0.6428571428571429,
0.0,
0.5714285714285714,
0.6666666666666666,
0.6964285714285714,
0.7678571428571429]
```

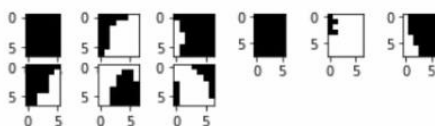
برای مثال عدد 8:

```
In [225]: plt.imshow(img_datapersian_black_white[8][2], cmap='gray')
```

```
Out[225]: <matplotlib.image.AxesImage at 0x2315c9f2fa0>
```



توجه: در ویژگی ARR تصویر به 12 قسمت تقسیم می شود که هر 12 قسمت را به دست آورده ایم اما در خروجی پایتون فقط 9 قسمت نمایش داده می شود (پایتون بیشتر از 9 قسمت نمایش نمی دهد). بنابراین خروجی 12 قسمت تصویر به صورت زیر است:



```
In [211]: def calculation_white_pixel(listt):
h=0
```

بردار ویژگی عدد 8 به صورت زیر است که شامل 12 قسمت است.

هر قسمت محاسبه شده تعداد پیکسل های سفید نسبت به کل پیکسل های آن ناحیه است.

```
In [227]: final_featurevector_arr_datapersian[8][2]
```

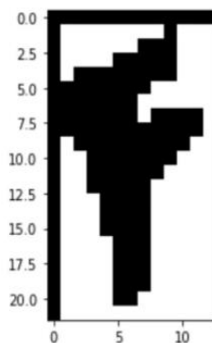
```
Out[227]: [0.0,
0.6530612244897959,
0.16326530612244897,
0.0,
0.8928571428571429,
0.25,
0.35714285714285715,
0.5102040816326531,
0.673469387755102,
0.8541666666666666,
0.08928571428571429,
0.5357142857142857]
```


<p>تابع transfer_row: این تابع 3 آرگومان ورودی : 1 تصویر باینری شده 2 ارتفاع 3 عرض تصویر را می گیرد. که مختصات هر انتقال به صورت افقی را در لیستی ذخیره کرده و به عنوان خروجی برمی گرداند.</p> <p>تابع transfer_col: این تابع 3 آرگومان ورودی : 1 تصویر باینری شده 2 ارتفاع 3 عرض تصویر را می گیرد. که مختصات هر انتقال به صورت عمودی را در لیستی ذخیره کرده و به عنوان خروجی برمی گرداند.</p> <p>تابع max_min_distance_r: ow: این تابع مختصات لیستی که انتقال سیاه و سفیدش به صورت افقی باشد را به عنوان ورودی می گیرد و بزرگ ترین و کوچک ترین فاصله آن را به همراه مختصات به عنوان خروجی برمی گرداند.</p> <p>تابع max_min_distance_c: ol: این تابع مختصات لیستی که انتقال سیاه و سفیدش به صورت عمودی باشد را به عنوان ورودی می گیرد و بزرگ ترین و کوچک ترین فاصله آن را به همراه مختصات به عنوان خروجی برمی گرداند.</p>	<p>برای مثال عدد 5:</p> <pre>In [201]: plt.imshow(listof_char[59], cmap='gray')</pre> <p>Out[201]: <matplotlib.image.AxesImage at 0x1b7d64a95e0></p>  <p>مختصات انتقال سیاه به سفید و سفید به سیاه به صورت افقی و عمودی:</p> <pre>In [197]: h,w=listof_char[59].shape coordinate_row=transfer_row(listof_char[59],7,w) coordinate_col=transfer_col(listof_char[59],h,8) print(coordinate_row) print(coordinate_col)</pre> <pre>[[(0, 7), (4, 7)], [(4, 7), (11, 7)]] [[(8, 0), (8, 3)], [(8, 3), (8, 12)], [(8, 12), (8, 18)], [(8, 18), (8, 24)]]</pre> <p>بزرگ ترین و کوچک ترین انتقال سیاه به سفید و سفید به سیاه:</p> <pre>In [198]: y=max_min_distance_row(coordinate_row) x=max_min_distance_col(coordinate_col) print(y) print(x)</pre> <pre>((7, [(4, 7), (11, 7)]), (4, [(0, 7), (4, 7)])) ((9, [(8, 3), (8, 12)]), (3, [(8, 0), (8, 3)]))</pre>	<p>ویژگی های انتقال سفید به سیاه بالعکس</p>
---	--	---

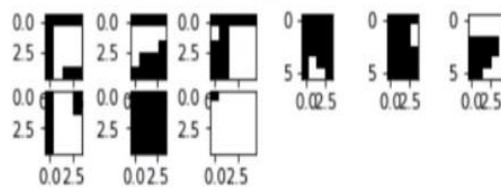
ویژگی های DTW

برای مثال عدد 4:

```
In [200]: plt.imshow(listof_char[18],cmap='gray')
Out[200]: <matplotlib.image.AxesImage at 0x1b7d643dfd0>
```



توجه: در ویژگی DTW تصویر به 24 قسمت تقسیم می شود که هر 24 قسمت را به دست آورده ایم اما در خروجی پایتون فقط 9 قسمت نمایش داده می شود (پایتون بیشتر از 9 قسمت نمایش نمی دهد). بنابراین خروجی 24 قسمت تصویر به صورت زیر است:



example

بردار ویژگی عدد 4 به صورت زیر است که شامل 24 قسمت است. هر قسمت از تصویر شامل بزرگ ترین و کوچک ترین فاصله حاشیه تا نویسه به صورت افقی و عمودی است.

تابع

:crope_image_dtw

این تابع یک تصویر باینری شده از ورودی می گیرد و آن را به 24 قسمت به صورت 6 سطر و 4 ستون تقسیم می کند.

تابع

:dis_row_dtw_black

این تابع قسمت crop شده تصویر را به عنوان ورودی می گیرد و فاصله ی حاشیه تا نویسه هم از چپ به راست و هم از راست به چپ به صورت افقی در یک لیستی به عنوان خروجی ذخیره می کند.

تابع

:dis_col_dtw_black

این تابع قسمت crop شده تصویر را به عنوان ورودی می گیرد و فاصله ی حاشیه تا نویسه هم از بالا به پایین و هم از پایین به بالا به صورت عمودی در یک لیستی به عنوان خروجی ذخیره می کند.

تابع **:max_min_**

در این تابع لیست های خروجی 2 تابع بالا را به عنوان ورودی گرفته و کوچک ترین و بزرگ ترین هر کدام را محاسبه می کند و به عنوان خروجی برمی گرداند.

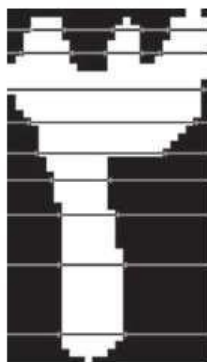
	<pre>In [94]: featurevector_dtw[18]</pre> <pre>Out[94]: [[(2, 0), (2, 0)], [(0, 0), (2, 2)], [(1, 0), (2, 0)], [(3, 0), (2, 0)], [(2, 0), (2, 0)], [(2, 0), (1, 0)], [(2, 0), (2, 0)], [(3, 3), (2, 2)], [(2, 0), (2, 0)], [(0, 0), (0, 0)], [(0, 0), (1, 0)], [(3, 1), (2, 0)], [(2, 2), (0, 0)], [(1, 0), (1, 0)], [(1, 0), (2, 0)], [(0, 0), (0, 0)], [(2, 2), (0, 0)], [(2, 1), (2, 0)], [(1, 1), (0, 0)], [(0, 0), (0, 0)], [(2, 2), (0, 0)], [(2, 2), (1, 1)]]</pre> <p>0%</p>	
		ویژگی های پیشنهادی شما

فاصله تا دیوار (DTW) Distance to Walls : برای بدست آوردن این ویژگی ،ناحیه کاراکتر به 24 قسمت تقسیم می شود.(6 سطر و 4 ستون) در هر ناحیه فاصله عمودی و افقی حاشیه تصویر تا حاشیه نویسه بدست می آید. برای هر ناحیه بزرگترین و کوچکترین فاصله عمودی و افقی ذخیره می شود .



شکل 3. فاصله تا دیوار

انتقال سفید به سیاه: با توجه به ویژگی نویسه های ایرانی، برای هر سطر و هر ستون از تصویر تعداد انتقال از سیاه به سفید و از سفید به سیاه محاسبه می شود. علاوه بر این، طول این انتقال از سیاه به سفید و سفید به سیاه نیز محاسبه می شود. سپس بزرگترین و کوچکترین طول به همراه محلشان برای سطر و ستون ذخیره می شود.



شکل 4. انتقال سفید به سیاه

نسبت ناحیه فعال (ARR) (Active Region Ratio):

برای محاسبه این ویژگی سطرها و ستون های فعال به چهار و سه ناحیه تقسیم می شوند. در هر ناحیه تعداد پیکسل های سفید بر تعداد کل پیکسل ها هر ناحیه تقسیم می شود.



شکل 5. نسبت ناحیه فعال





4- تابع دسته بندی

در این قسمت، اعداد و کاراکترهای فارسی به استفاده از رو های دسته بندی مختلف به صورت صحیح دسته بندی شوند. برای این منظور ابتدا بر اساس داده های موجود مدل های مختلف دسته بندی آموزش و ارزیابی می شود. سپس کاراکترها و اعداد با استفاده از مدل ها تشخیص داده می شوند. شما در این مرحله جدول زیر را باید کامل کنید.

جدول 3. بررسی دقت روش های مختلف دسته بندی بر اساس ویژگی های مختلف

Method	ویژگی های آماری				ویژگی های LBP				ترکیب ویژگی ها			
	Accuracy	Specificity	Sensitivity	time	Accuracy	Specificity	Sensitivity	time	Accuracy	Specificity	Sensitivity	time
RF												
SVM												
MLP												
KNN												
DT												

در نهایت خروجی برنامه شما باید تصویر ورودی را بگیرد و شماره پلاک در در خروجی چاپ نماید. خروجی نمونه در شکل 6 نشان داده شده است.

	33 ق 884- 44		63 ن 545-10
	ط 189-66		25 65 و 778-11

شکل 6. خروجی نهایی

با آرزوی بهترین ها.