cs512 Project Report

Yiran Zhang
A20407655

1 Problem statement

1. Captcha image preproessing
   - Circle detection
   - Remove interference circle
   - Denoising
   - Binarization
2. Character segmentation
   - Identify the conglutination type
   - Segmentation of not conglutination characters
   - Segmentation of part conglutination characters
   - Segmentation of high conglutination characters
3. Automatic captcha recognition
   - Build CNN model
   - Training CNN model
   - predict the captcha

2 Proposed solution

1. Circle detection
   - In order to detect circle, we need to preprocess the image first, we need to convert the image to gray and using Gaussion blur to process the captcha image.
   - We use hough transfer algorithm to detect the circle interference.
   - Denoising the image with median filtering algorithm

- convert the iamge to a binary image

2. the main algorithm to Character segmentation by improved vertical projection

   - Add up the element of the same column in captcha image 0-1 matrix, get a one-dimensional array, named as col[ ], each element value of the array corresponding to the total number of black pixels in each column.
   - Iterate through array col[ ], the sequence number of elements value 0 represent for blank columns corresponding abscissa, record it to a one-dimensional array, named as p[ ].
   - Iterate through p[ ], orderly do p[j]-p[j-1], getting the distance of the most close two points of projection valuing 0, and compare with the average width d of characters. If p[j]-p[j-1]¡d, it suggests that these two points are in the same blank area. If 2d¿ p[j]-p[j-1]¿d, it suggests that the points in position p[j] and p[j-1] belong to two different blank areas, then (p[j-1], p[j]) determines one character area and is the segmentation point of not conglutination characters.
   - If p[j]-p[j-1]¿2d, it suggests that there is point of projection not valuing 0 exiting the two characters width area, there is no blank gap between adjacent characters, namely these adjacent characters are conglutination characters. Iterate through the value of col[i], (i(p[j-1], p[j])), find out the maximum value max, compare the value of i corresponding to max minus p[j-1] with character width d, if they are nearly equal, then i will be the segmentation point of edge tangent conglutination characters.
   - Iterate through the value of col[i], (i(p[j-1], p[j])), find out the minimum value min, compare the value of i corresponding to min minus p[j-1] with character width d, if they are nearly equal, then i will be the segmentation point of edge horn conglutination characters.
   - If the above two cases are not set up, it will be high conglutination character, then adopt the method of average cutting, reserve the characteristics of single character information to the most degree

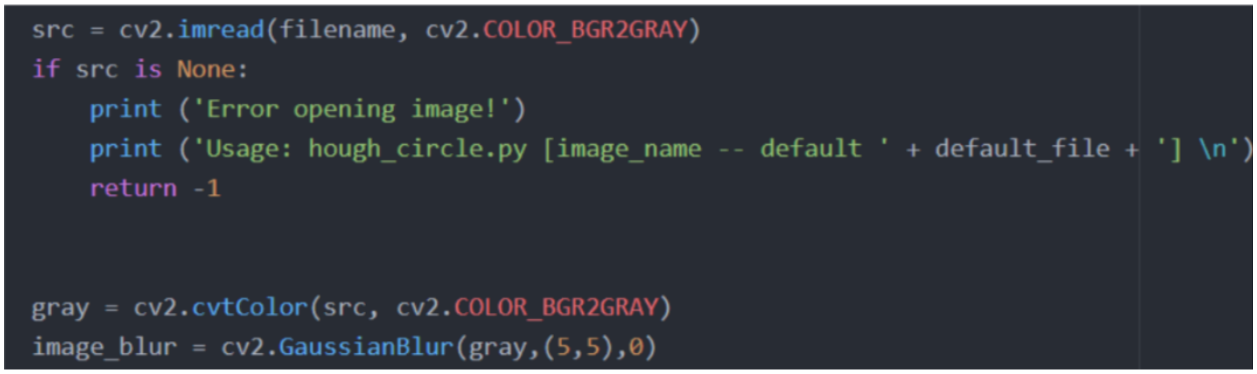
3. Character recognition

   - Build a CNN to predict the captcha image.
   - Collect a dataset to train the model

3 Implementation details

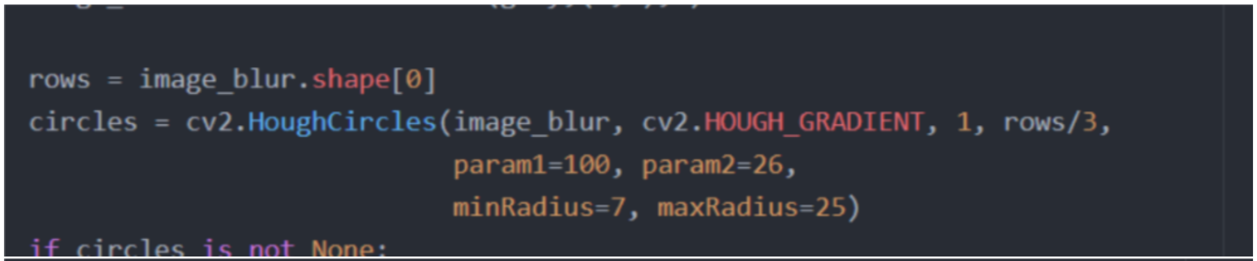
1. Preprocessing image

   Convert to gray and blur the image.

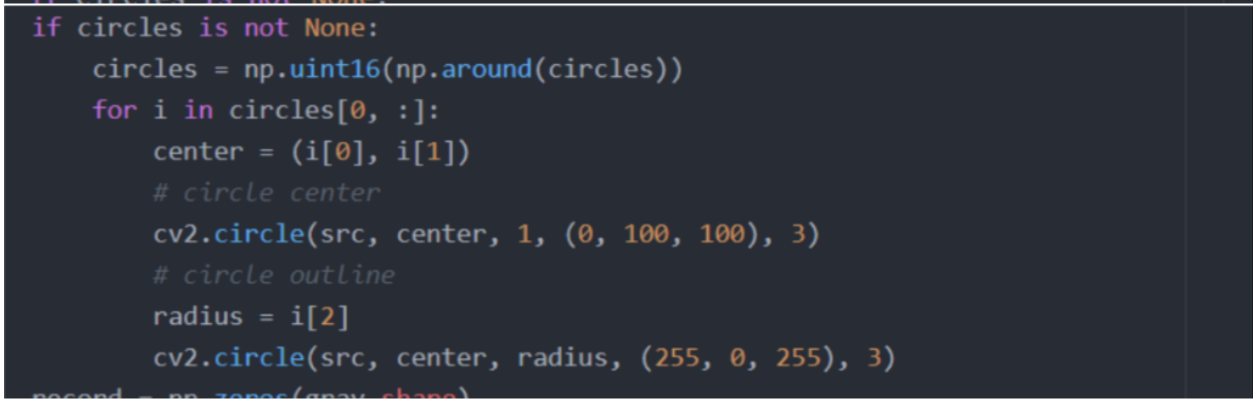


```
src = cv2.imread(filename, cv2.COLOR_BGR2GRAY)
if src is None:
    print ('Error opening image!')
    print ('Usage: hough_circle.py [image_name -- default ' + default_file + '] \n')
    return -1


gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
image_blur = cv2.GaussianBlur(gray,(5,5),0)
```

Using Hough transfer algorithm to find circles and then label the circle

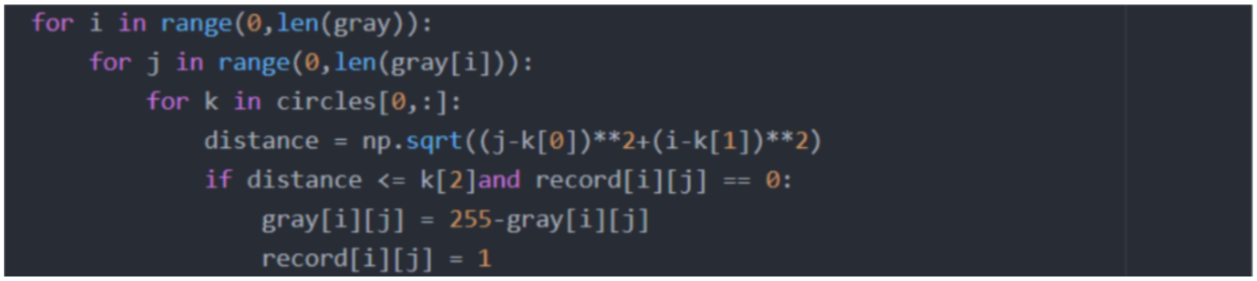


```
rows = image_blur.shape[0]
circles = cv2.HoughCircles(image_blur, cv2.HOUGH_GRADIENT, 1, rows/3,
                           param1=100, param2=26,
                           minRadius=7, maxRadius=25)
if circles is not None:
```



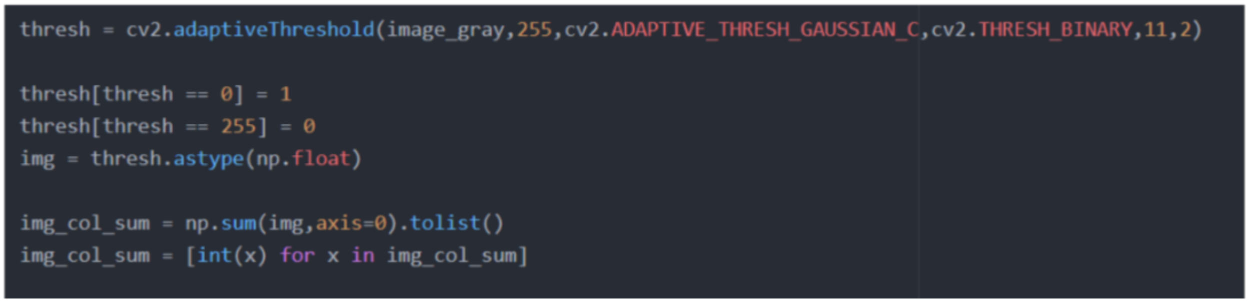
```
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        center = (i[0], i[1])
        # circle center
        cv2.circle(src, center, 1, (0, 100, 100), 3)
        # circle outline
        radius = i[2]
        cv2.circle(src, center, radius, (255, 0, 255), 3)
record = np.zeros(gray.shape)
```

Convert the value in circles, if the distance between the point and a circle less than radius, we will using 255 minus the value to reverse value



```
for i in range(0,len(gray)):
    for j in range(0,len(gray[i])):
        for k in circles[0,:]:
            distance = np.sqrt((j-k[0])**2+(i-k[1])**2)
            if distance <= k[2]and record[i][j] == 0:
                gray[i][j] = 255-gray[i][j]
                record[i][j] = 1
```

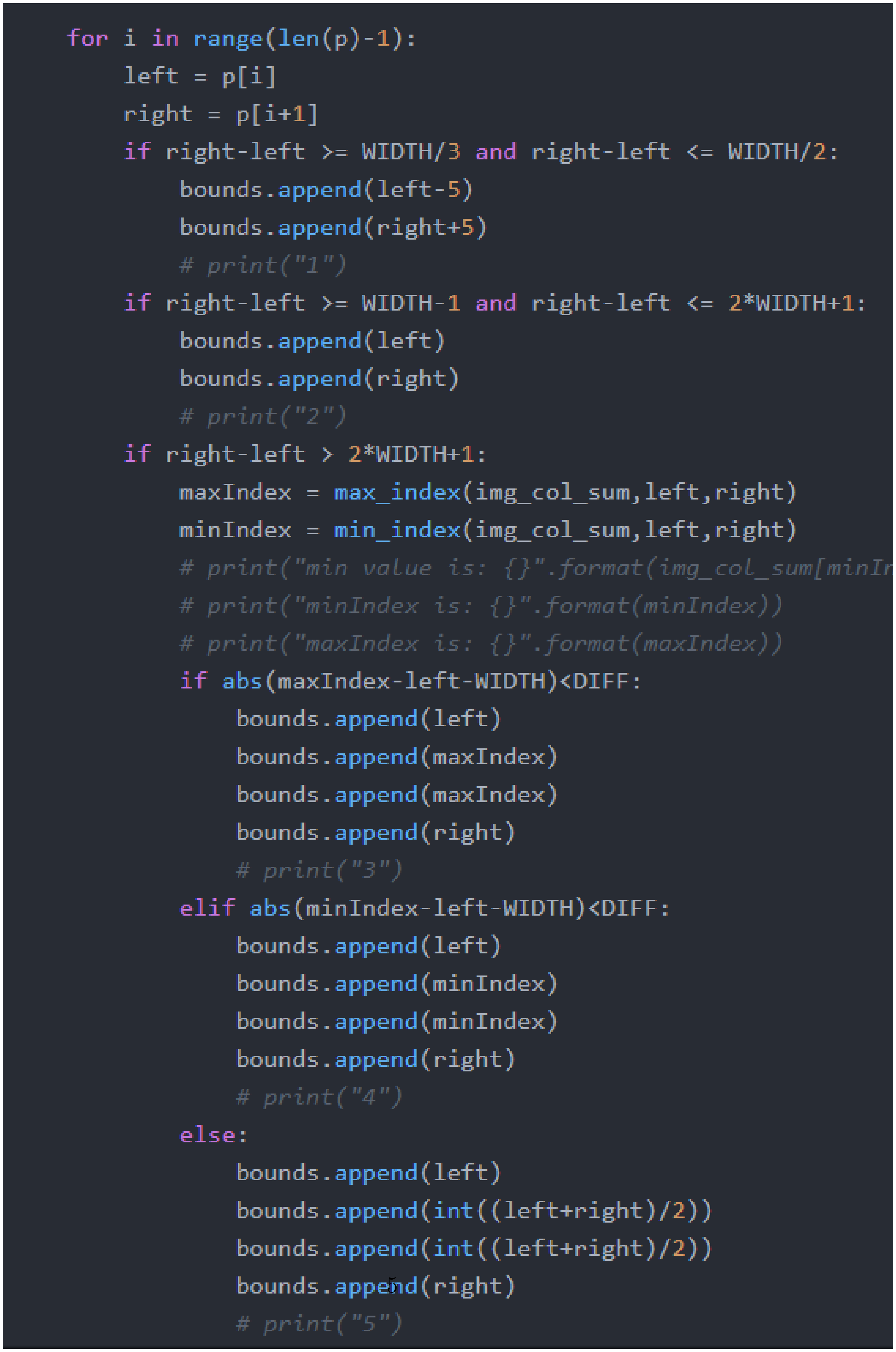
Convert the iamge to binary



```
thresh = cv2.adaptiveThreshold(image_gray,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)

thresh[thresh == 0] = 1
thresh[thresh == 255] = 0
img = thresh.astype(np.float)

img_col_sum = np.sum(img,axis=0).tolist()
img_col_sum = [int(x) for x in img_col_sum]
```

2. A Improved Vertical projection algorithm

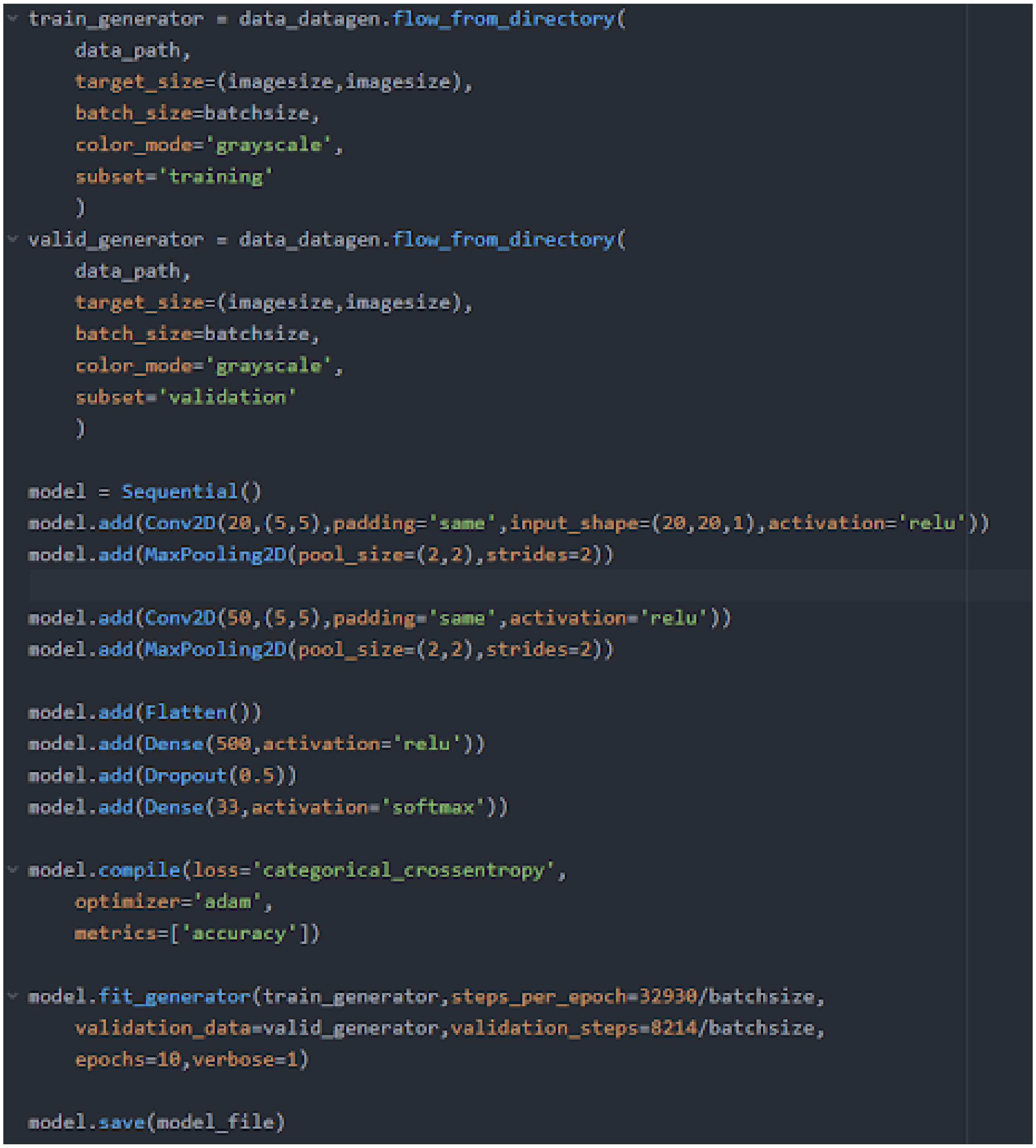
   Using vertical projection to divide the letters in image



```
    for i in range(len(p)-1):
        left = p[i]
        right = p[i+1]
        if right-left >= WIDTH/3 and right-left <= WIDTH/2:
            bounds.append(left-5)
            bounds.append(right+5)
            # print("1")
        if right-left >= WIDTH-1 and right-left <= 2*WIDTH+1:
            bounds.append(left)
            bounds.append(right)
            # print("2")
        if right-left > 2*WIDTH+1:
            maxIndex = max_index(img_col_sum,left,right)
            minIndex = min_index(img_col_sum,left,right)
            # print("min value is: {}".format(img_col_sum[minIn
            # print("minIndex is: {}".format(minIndex))
            # print("maxIndex is: {}".format(maxIndex))
            if abs(maxIndex-left-WIDTH)<DIFF:
                bounds.append(left)
                bounds.append(maxIndex)
                bounds.append(maxIndex)
                bounds.append(right)
                # print("3")
            elif abs(minIndex-left-WIDTH)<DIFF:
                bounds.append(left)
                bounds.append(minIndex)
                bounds.append(minIndex)
                bounds.append(right)
                # print("4")
            else:
                bounds.append(left)
                bounds.append(int((left+right)/2))
                bounds.append(int((left+right)/2))
                bounds.append(right)
                # print("5")
```

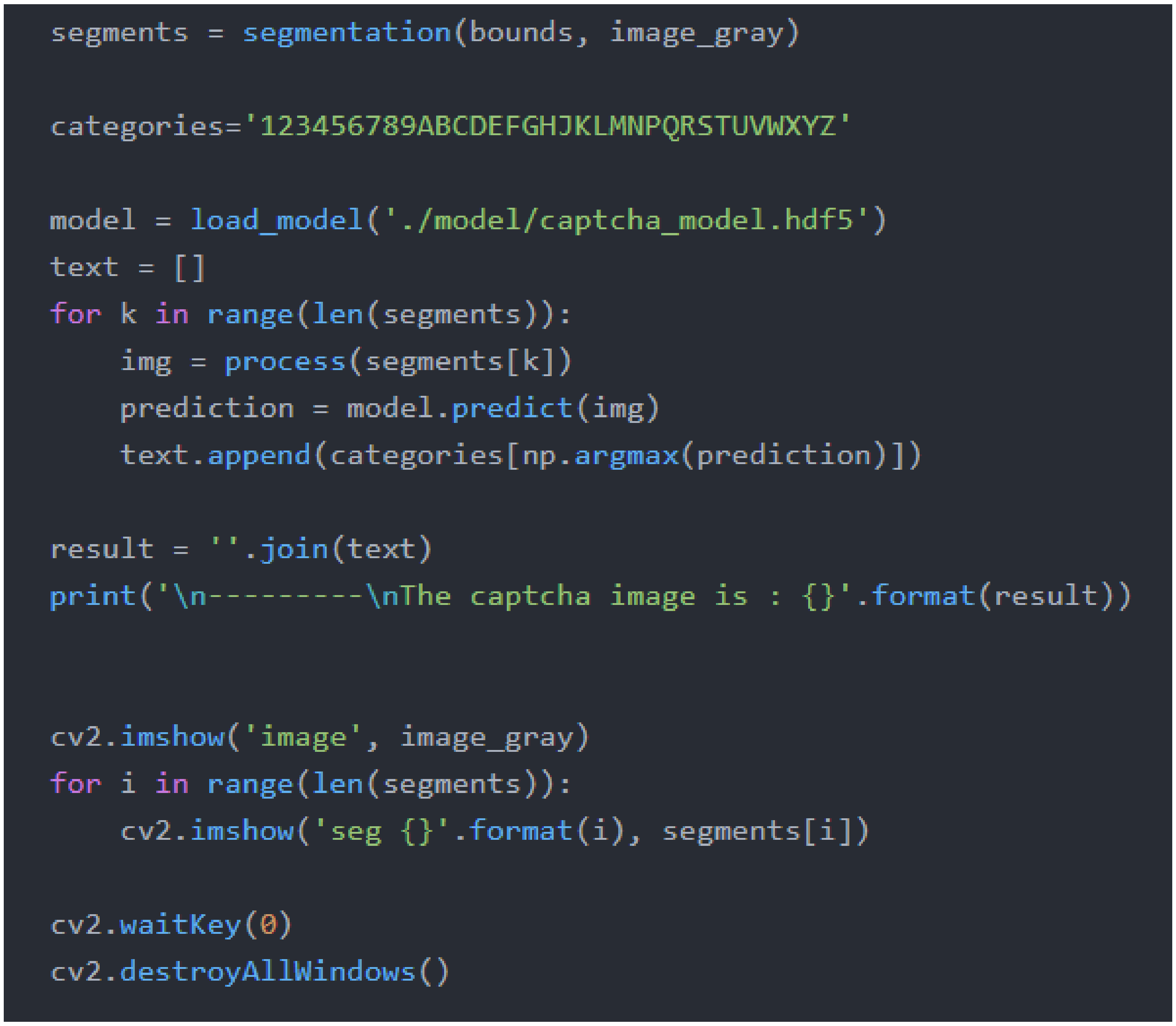
3. Using CNN model to recognition captchas

   Build CNN model



```
train_generator = data_datagen.flow_from_directory(
    data_path,
    target_size=(imagesize,imagesize),
    batch_size=batchsize,
    color_mode='grayscale',
    subset='training'
    )
valid_generator = data_datagen.flow_from_directory(
    data_path,
    target_size=(imagesize,imagesize),
    batch_size=batchsize,
    color_mode='grayscale',
    subset='validation'
    )

model = Sequential()
model.add(Conv2D(20,(5,5),padding='same',input_shape=(20,20,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=2))

model.add(Conv2D(50,(5,5),padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=2))

model.add(Flatten())
model.add(Dense(500,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(33,activation='softmax'))

model.compile(loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy'])

model.fit_generator(train_generator,steps_per_epoch=32930/batchsize,
    validation_data=valid_generator,validation_steps=8214/batchsize,
    epochs=10,verbose=1)

model.save(model_file)
```

Recognize image, because the main dataset that we used is from a captcha website called wordpress, thay think some letters like ’1’ and ’I’ are too similer to recgonize by people, so they remove it. so our dataset just have 33 categories

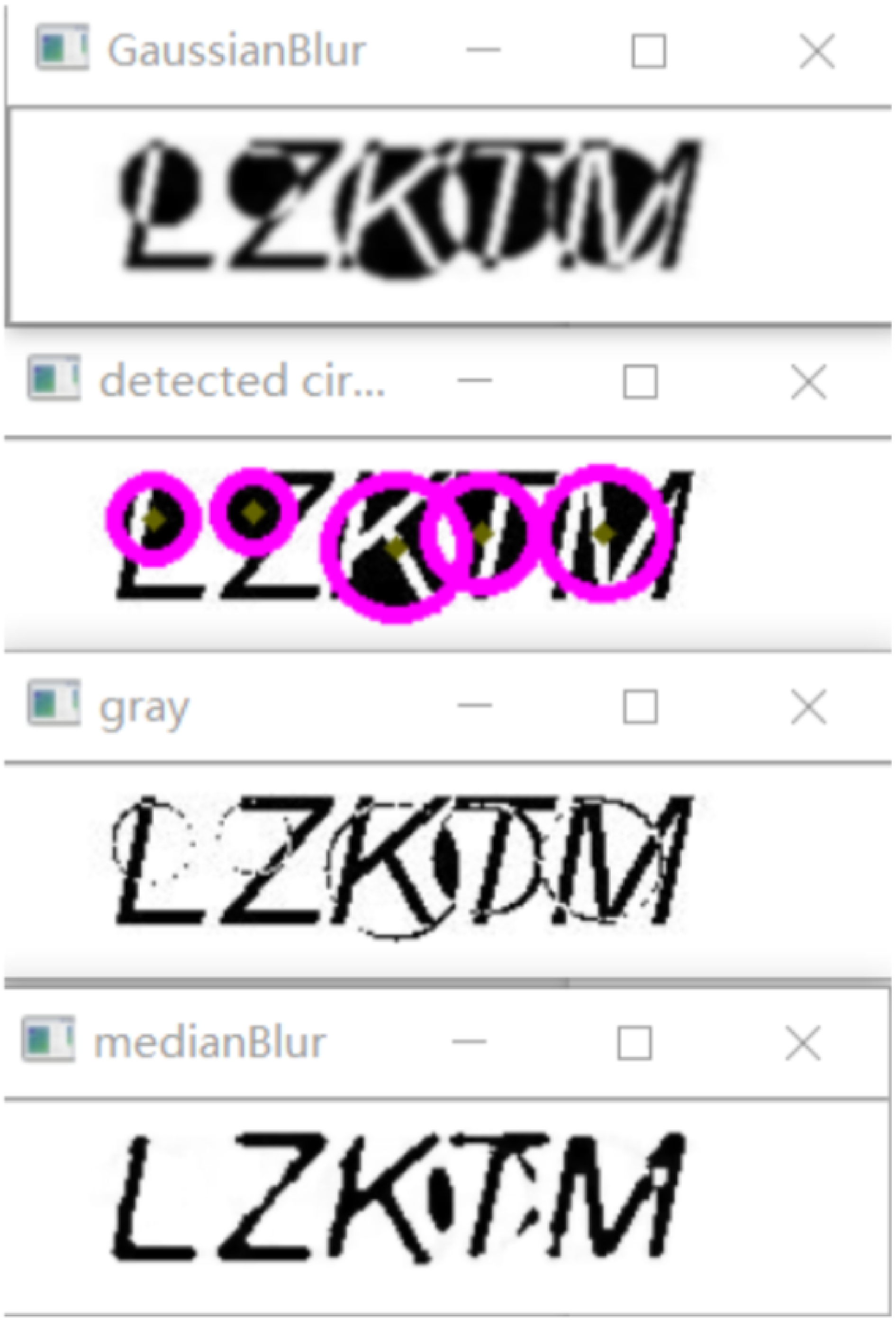


```
segments = segmentation(bounds, image_gray)

categories='123456789ABCDEFGHJKLMNPQRSTUVWXYZ'

model = load_model('./model/captcha_model.hdf5')
text = []
for k in range(len(segments)):
    img = process(segments[k])
    prediction = model.predict(img)
    text.append(categories[np.argmax(prediction)])

result = ''.join(text)
print('\n---------\nThe captcha image is : {}'.format(result))


cv2.imshow('image', image_gray)
for i in range(len(segments)):
    cv2.imshow('seg {}'.format(i), segments[i])

cv2.waitKey(0)
cv2.destroyAllWindows()
```

4 Results and discussion

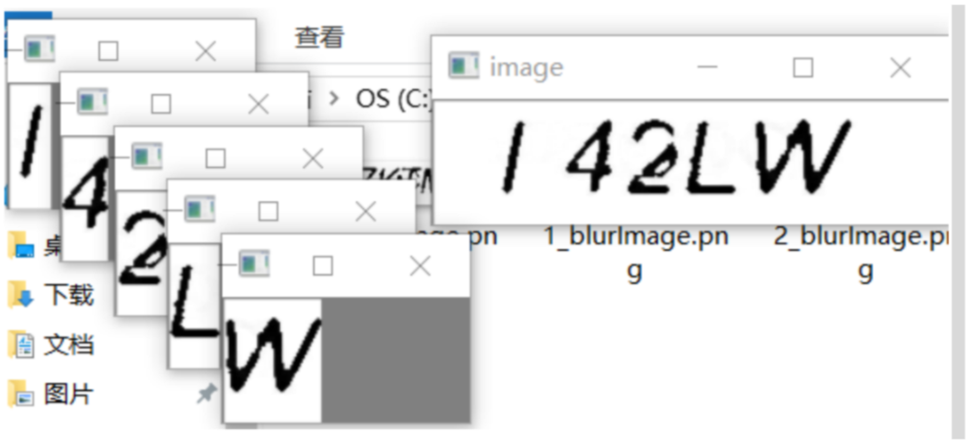
1. Preprocessing captcha

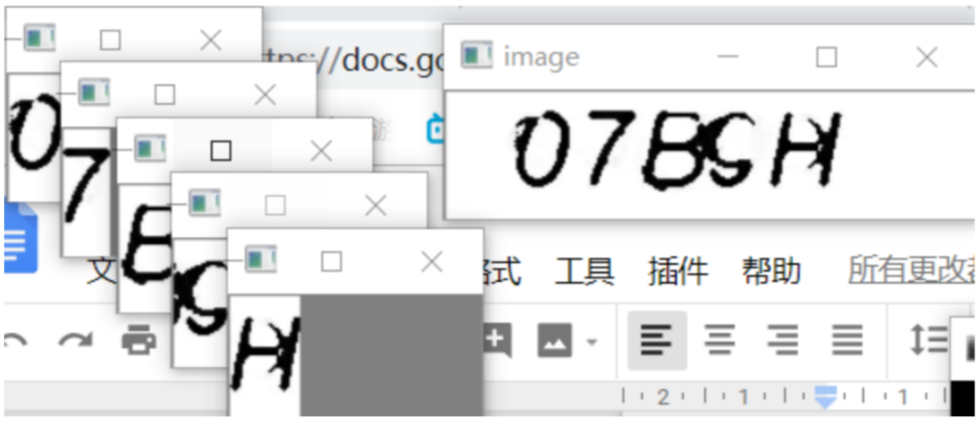
   The preprocessing are not always good enough, because the circle are different in every image, it is hardly to find all circles in every captchas with just one group of parameters. If we want to get a better result, we might look for another way to detect circles. Although the paper saids we can detect circle using hough transfer directly, I add a gaussian blur before the hough tranfer algorithm, it get a better result.



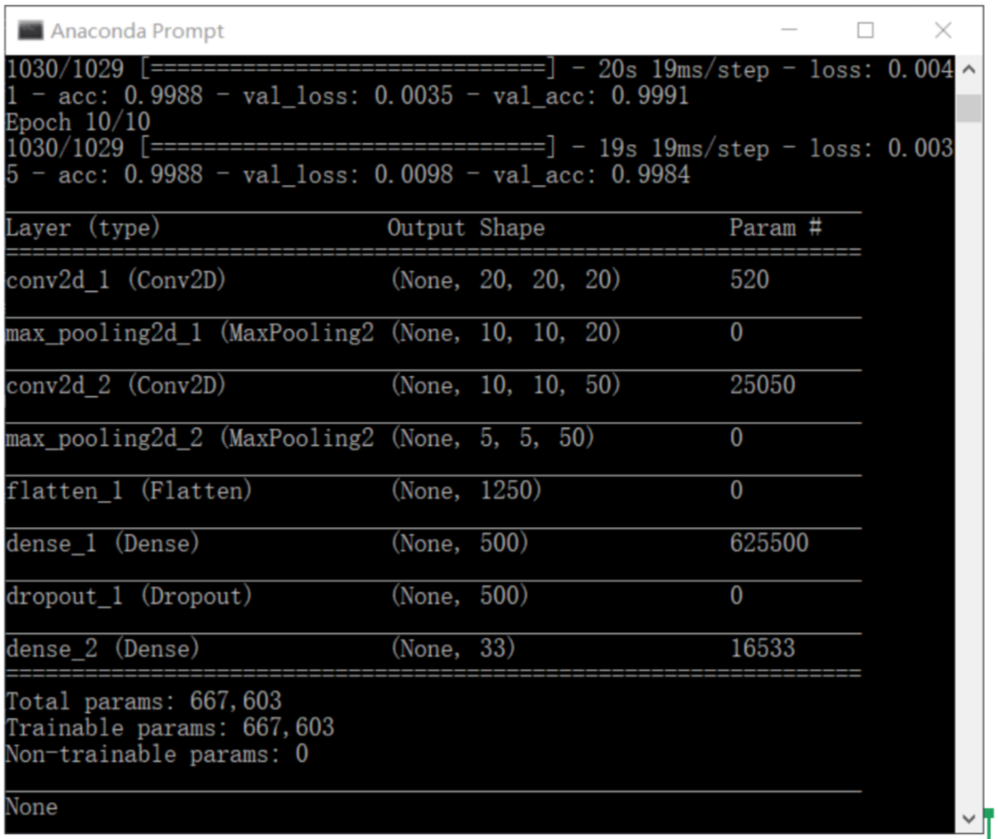
2. Segmentation

   The segmentation algorithm is based on vertical projection. The result is good but it also need to be improved.



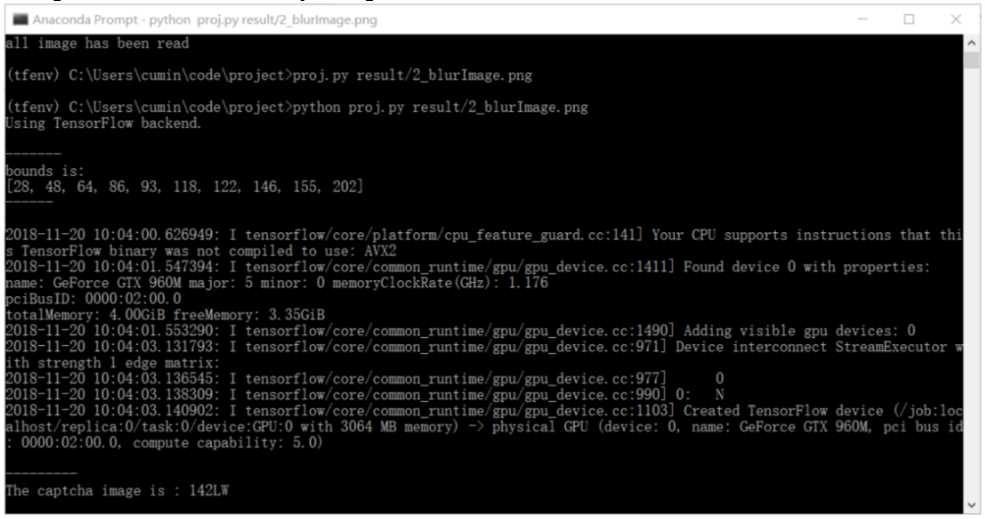


3. recognition Training CNN model



```
1030/1029 [==============================] - 20s 19ms/step - loss: 0.004
1 - acc: 0.9988 - val_loss: 0.0035 - val_acc: 0.9991
Epoch 10/10
1030/1029 [==============================] - 19s 19ms/step - loss: 0.003
5 - acc: 0.9988 - val_loss: 0.0098 - val_acc: 0.9984
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 20, 20, 20)        520
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 20)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 50)        25050
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 50)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1250)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               625500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 33)                16533
=================================================================
Total params: 667,603
Trainable params: 667,603
Non-trainable params: 0
_________________________________________________________________
None
```

The result are good if previous step is good, if we get a clean image we can recognition it easily, but when we have a image with some noise, it always make some small mistake like predict a 'O' as 'Q' or predict 'E' as 'F'



```
all image has been read

(tfenv) C:\Users\cumin\code\project>proj.py result/2_blurImage.png

(tfenv) C:\Users\cumin\code\project>python proj.py result/2_blurImage.png
Using TensorFlow backend.

-------
bounds is:
[28, 48, 64, 86, 93, 118, 122, 146, 155, 202]
------

2018-11-20 10:04:00.626949: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2
2018-11-20 10:04:01.547394: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1411] Found device 0 with properties:
name: GeForce GTX 960M major: 5 minor: 0 memoryClockRate(GHz): 1.176
pciBusID: 0000:02:00.0
totalMemory: 4.00GiB freeMemory: 3.35GiB
2018-11-20 10:04:01.553290: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1490] Adding visible gpu devices: 0
2018-11-20 10:04:03.131793: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] Device interconnect StreamExecutor w
ith strength 1 edge matrix:
2018-11-20 10:04:03.136545: I tensorflow/core/common_runtime/gpu/gpu_device.cc:977]      0
2018-11-20 10:04:03.138309: I tensorflow/core/common_runtime/gpu/gpu_device.cc:990] 0:   N
2018-11-20 10:04:03.140902: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1103] Created TensorFlow device (/job:loc
alhost/replica:0/task:0/device:GPU:0 with 3064 MB memory) -> physical GPU (device: 0, name: GeForce GTX 960M, pci bus id
: 0000:02:00.0, compute capability: 5.0)

---------
The captcha image is : 142LW
```

5 Reference

Lili zhang,Yuxiang Xie, Xidao Luan, Jingmeng He. Captcha Automatic Segmentation and Recognition Based on Improved Vertical Projection