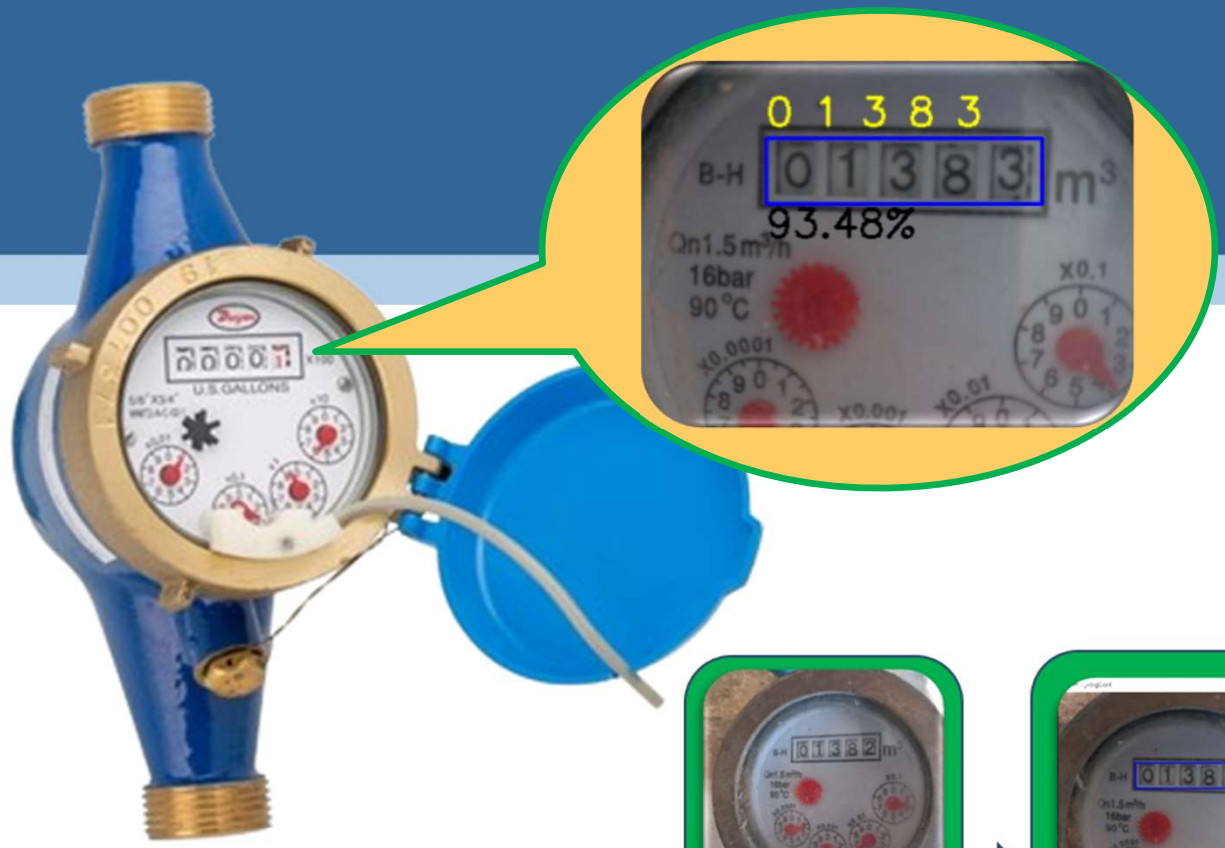
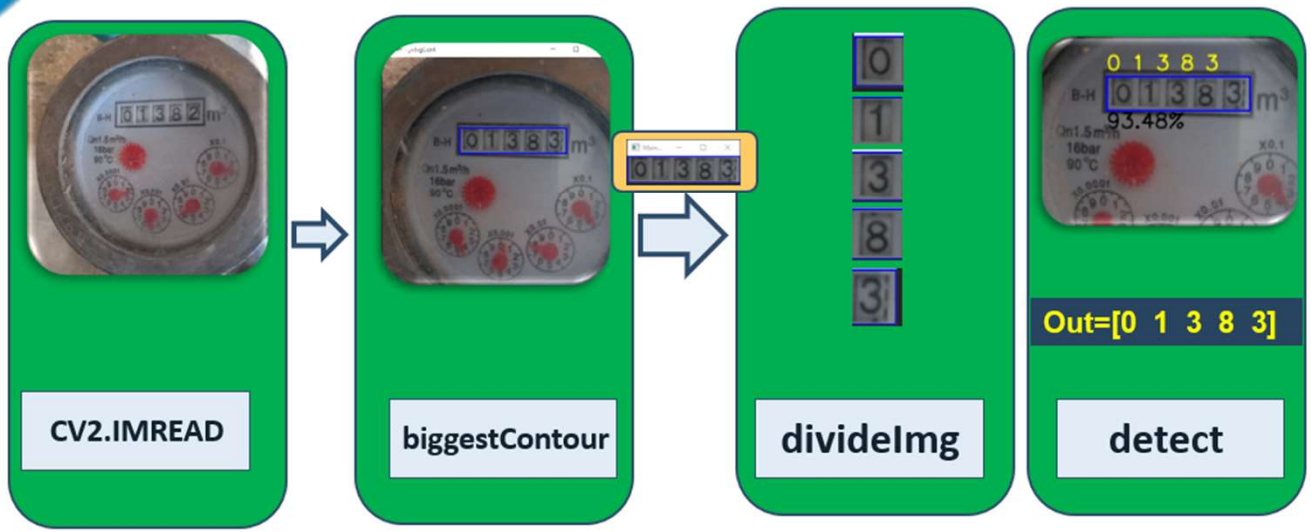
برنامه خوانش هوشمند کنتور آب





شرح پروژه

❖ توضیحات:

در این پروژه قصد داریم با استفاده از پردازش تصویراعداد روی کنتور آب را بخوانیم در واقع میتوان آن را یک قسمت ابتدایی از پروژه بزرگ تر دانست که آن اپلیکیشن خوانش هوشمند کنتور آب است.

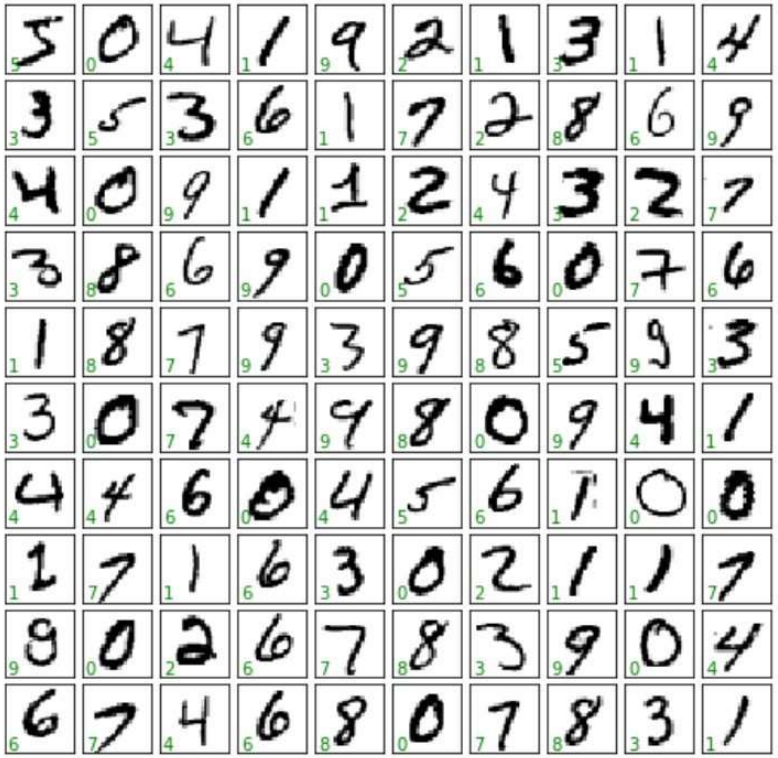
❖ ابزار

در این پروژه با استفاده از برنامه نویسی پایتون یک الگوریتم برای تشخیص اعداد پیدا کردیم طوریکه تصویر هر عدد را از تصویر کنتور استخراج کرده و آن را به مدل دهیم تا عدد موجود در تصویر را پیش بینی کند.

❑ دیتا ست

❖دیتاست

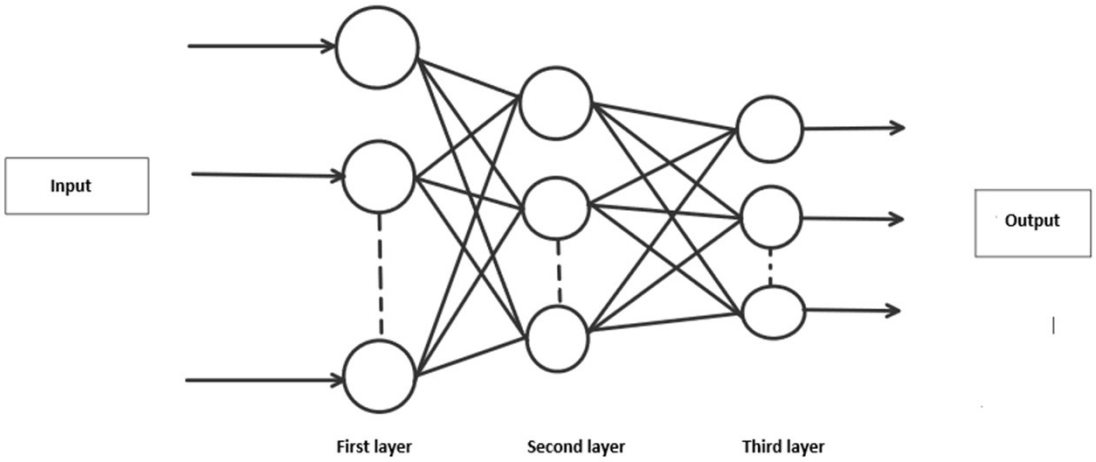
برای این که بتوان اعداد موجود در یک تصویر را تشخیص داد باید مدلی از اعداد ۰ الی ۹ را بصورت یادگیری نظارت شده پردازش کرد.



❑ دیتا ست

❖دیتاست

دیتاست انتخاب شده **Handwritten digit** است که شامل ۷۰۰۰۰ عدد دستنویس است باید به یک شبکه عصبی داده شود تا مدل آن با دقت بالای ۹۹ درصد استخراج شود.



❑ داده آموزشی و ارزیابی

❖ داده آموزشی و ارزیابی

در کتابخانه تنسورفلو این دیتا ست موجود است بعد بارگیری دیتاست باید آن را به دو بخش train وtest تقسیم کرد که بخش اعظم آن را داده آموزشی برای شبکه در برمیگیرد.

Load the data

The **training** dataset consists of 60000 28x28px images of hand-written digits from `0` to `9`.

The **test** dataset consists of 10000 28x28px images.

```
[ ]  mnist_dataset = tf.keras.datasets.mnist
     (x_train, y_train), (x_test, y_test) = mnist_dataset.load_data()
```

```
print('x_train:', x_train.shape)
print('y_train:', y_train.shape)
print('x_test:', x_test.shape)
print('y_test:', y_test.shape)
```

```
x_train: (60000, 28, 28)
y_train: (60000,)
x_test: (10000, 28, 28)
y_test: (10000,)
```

❑ داده آموزشی و ارزیابی

❖ داده آموزشی و ارزیابی

برای استفاده از لایه های کانولوشن، باید داده های خود را تغییر شکل دهیم و یک کانال رنگی به آن اضافه کنیم.

```
[ ]  x_train_with_chanels = x_train.reshape(
         x_train.shape[0],
         IMAGE_WIDTH,
         IMAGE_HEIGHT,
         IMAGE_CHANNELS
     )

     x_test_with_chanels = x_test.reshape(
         x_test.shape[0],
         IMAGE_WIDTH,
         IMAGE_HEIGHT,
         IMAGE_CHANNELS
     )
```

```
[ ]  print('x_train_with_chanels:', x_train_with_chanels.shape)
     print('x_test_with_chanels:', x_test_with_chanels.shape)
```

```
x_train_with_chanels: (60000, 28, 28, 1)
x_test_with_chanels: (10000, 28, 28, 1)
```

❑ داده آموزشی و ارزیابی

❖ نرمالیزه کردن دیتا

برای این که داده پیکسل های تصویر را به شبکه عصبی بدیم باید آن ها را نرمالایز کنیم.

▾ Normalize the data

Here we're just trying to move from values range of `[0...255]` to `[0...1]`.

```
[ ]  x_train_normalized = x_train_with_chanels / 255
     x_test_normalized = x_test_with_chanels / 255
```

❑ داده آموزشی و ارزیابی

❖ داده آموزشی و ارزیابی

پس از آن شبکه عصبی را طراحی کرده و داده **train** را به آن میدهیم. کامپایل کرده و آن را به فرمت **h5.** ذخیره میکنیم.

```
model = keras.Sequential([
    keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
])

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

model.fit(x_train_flattened, y_train, epochs=10 )
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖پردازش اعداد کنتور

برای اینکه بتوانیم اعداد روی کنتور رو تفکیک کنیم باید ابتدا کانتوری که اعداد رو شامل میشه پیدا کنیم.
باید از تابع زیر استفاده کنیم:

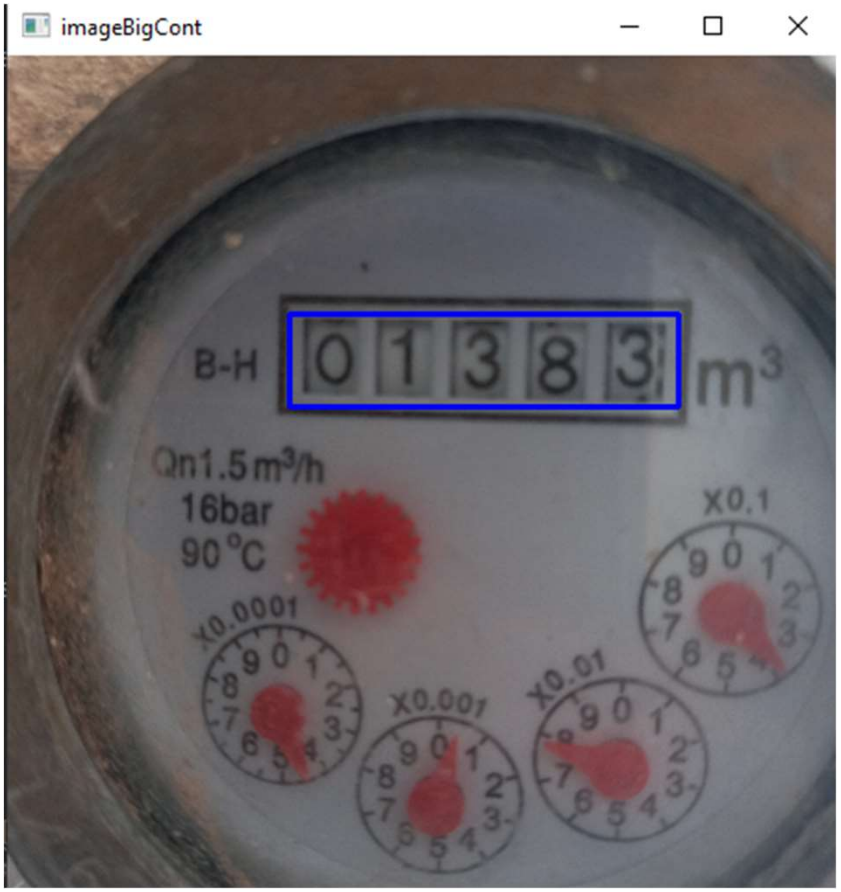
```
def biggestContour(contours):
    global biggest
    biggest = np.array([])
    max_area = 0
    for i in contours:
        area = cv2.contourArea(i)
        if area > 50:
            peri = cv2.arcLength(i, True)
            approx = cv2.approxPolyDP(i, 0.02 * peri, True)
            if area > max_area and len(approx) == 4:
                biggest = approx
                max_area = area
    return biggest, max_area
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

برای اینکه بتوانیم اعداد روی کنتور رو تفکیک کنیم باید ابتدا کانتوری که اعداد رو شامل میشه پیدا کنیم.

حاصل:



❑ استفاده از مدل برای تشخیص اعداد کنتور

❖پردازش اعداد کنتور

برای اینکه بتوانیم اعداد روی کنتور رو تفکیک کنیم باید ابتدا کانتوری که اعداد رو شامل میشه پیدا کنیم.

پس از آن باید این چهار ضلعی را تبدیل به مستطیل کرد.

تابع مورد استفاده:

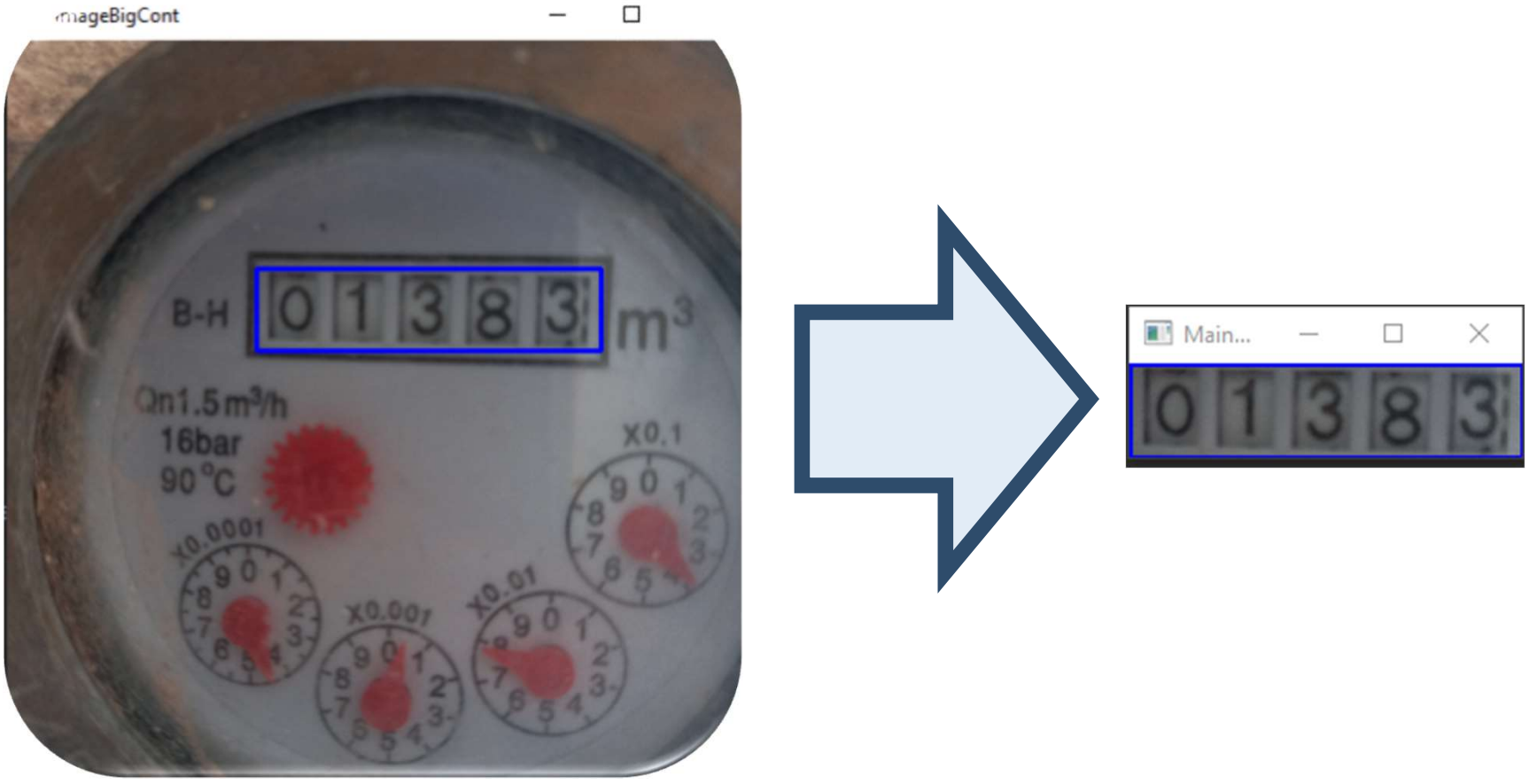
```python
def four_point_transform(image, pts):
    rect = order_points(pts)
    (tl, tr, br, bl) = rect

    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]], dtype="float32")
    # compute the perspective transform matrix and then apply it
    M = cv2.getPerspectiveTransform(rect, dst)
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
    # return the warped image
    return warped
# ==========
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖پردازش اعداد کنتور

برای اینکه بتوانیم اعداد روی کنتور رو تفکیک کنیم باید ابتدا کانتوری که اعداد رو شامل میشه پیدا کنیم.

شکل حاصل:



❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

برای اینکه بتوانیم اعداد روی کنتور رو تفکیک کنیم باید ابتدا کانتوری که اعداد رو شامل میشه پیدا کنیم.

اکنون باید این مستطیل شامل پنج عدد مساوی را تبدیل به پنج تصویر هر کدام شامل عدد کنیم:

ورودی این تابع ۱ الی پنج است و خروجی آن تصویر یکی از حاضر در مستطیل است.

```
def divideImg(value):
    g = 2  # set number boxes
    if value == 1:  # first number
        icon = Rectimg[0 + 4 * g:32 - 2 * g, 0 + 3 * g:32 + g]
    if value == 2:  # second number
        icon = Rectimg[0 + 2 * g:32 - 2 * g, 32 + 3 * g:64 - g]
    if value == 3:  # third number
        icon = Rectimg[0 + 2 * g:32 - 2 * g, 64 + 3 * g:96 - g]
    if value == 4:  # forth number
        icon = Rectimg[0 + 2 * g:32 - 2 * g, 96 + 3 * g:128 - g]
    if value == 5:  # fifth number
        icon = Rectimg[0 + 2 * g:32 - 2 * g, 128 + 2 * g:160 - 3 * g]

    return icon
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

در حلقه اصلی برنامه میبینیم که تصویر گرفته شده پیش پردازش شده و به تابع تشخیص دهنده داده میشود و خروجی آن ۵ عدد روی کنتور آب است.

```
while True:
    # get images
    img = getImg(readData)
    # preProcessing
    biggest = preProcess(img)
    # detect digits
    p = detect(biggest)
    print(p)
    # see results
    drawResults()
# cv2.imshow('number', resdyForPred(cv2.resize(divideImg(4), (50, 50))))
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

در تابع **preprocess** تابع **biggesstContours** برای پیداکردن مکان اعداد وجود دارد.

```
def preProcess(img):
    global  imgThreshold, imgContours, imgBigContour
    img = cv2.resize(img, (450, 450))
    imgContours = img.copy()
    imgBigContour = img.copy()
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (5, 5), 1)
    imgThreshold = cv2.adaptiveThreshold(imgBlur, 255, 1, 1, 11, 2)
    #cv2.imshow('imgThreshold', imgThreshold)

    contours, hierarchy = cv2.findContours(imgThreshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(imgContours, contours, -1, (0, 255, 0), 3)
    biggest, maxArea = biggestContour(contours)
    mainShape(biggest, imgBigContour)

    return biggest
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

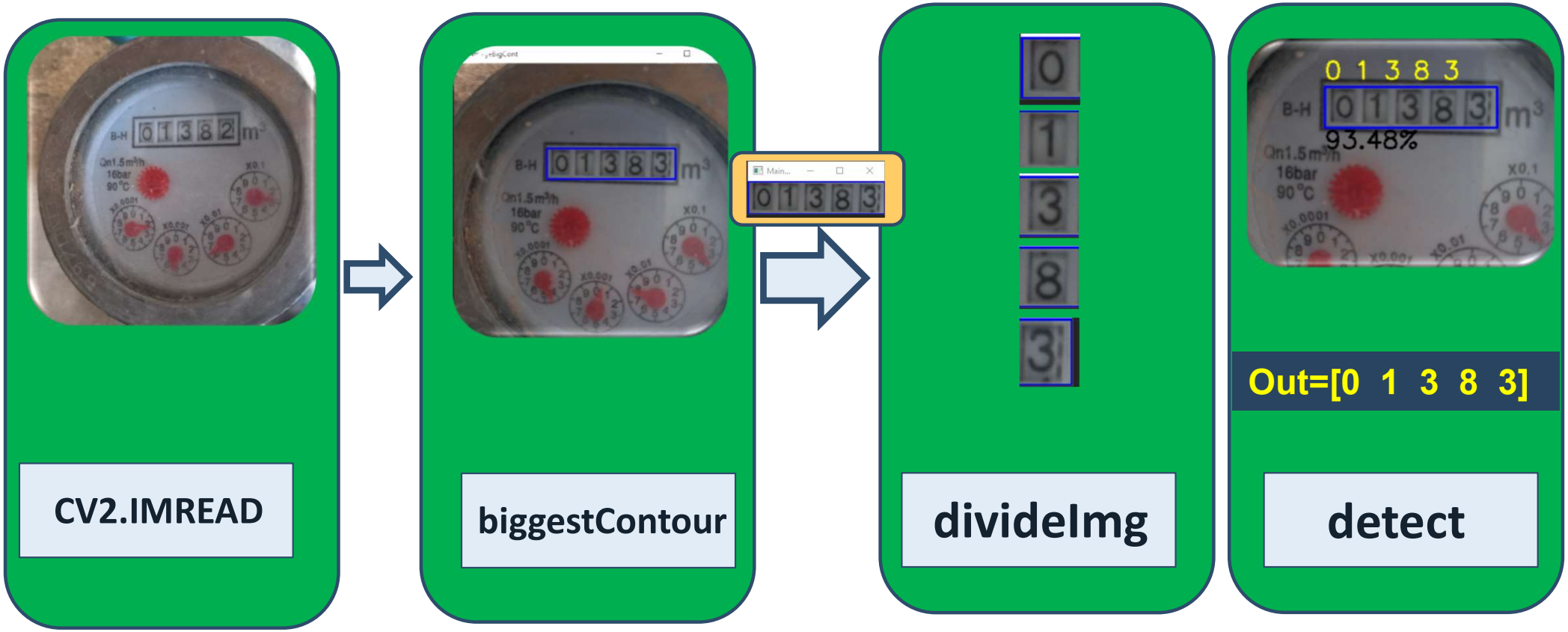
پس از آن کانتور شامل اعداد را به تابع **detect** میدهیم تا پنج عدد را در قالب یک ماتریس به بدهد.

```
def detect(biggest):
    for i in range(1, 6):
        if biggest is not None:
            img = cv2.resize(divideImg(i), (32, 32))
            img1 = resdyForPred(img)
            img = img1.reshape(1, 32, 32, 1)
            cv2.imshow('resdyForPred', img1)
            #cv2.waitKey(500)
            predictions = model.predict(img)

            probVal = np.amax(predictions)

            max_index = predictions.argmax()
            if probVal > 0.5:
                p[i - 1] = max_index
                pval[i - 1] = probVal

    return p
```

❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

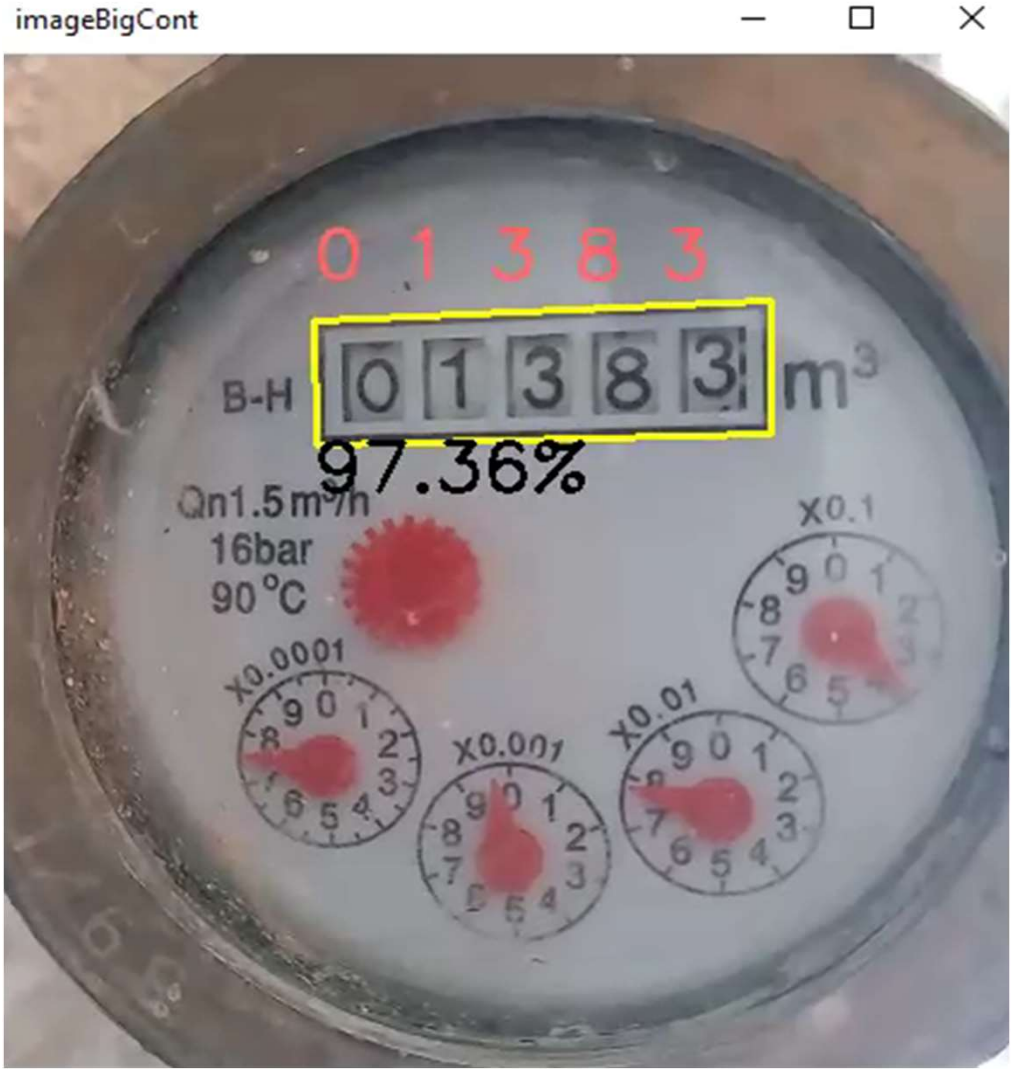
پس از آن کانتور شامل اعداد را به تابع **detect** میدهیم تا پنج عدد را در قالب یک ماتریس به بدهد.



❑ استفاده از مدل برای تشخیص اعداد کنتور

❖ پردازش اعداد کنتور

تشخیص نهایی:



آدرس های ارتباطی

github.com/SmFaraji

Projects channel -> t.me/EngineeringLab
aparat channel -> www.aparat.com/EngineeringLab
YouTube channel -> https://www.youtube.com/@sm_faraji