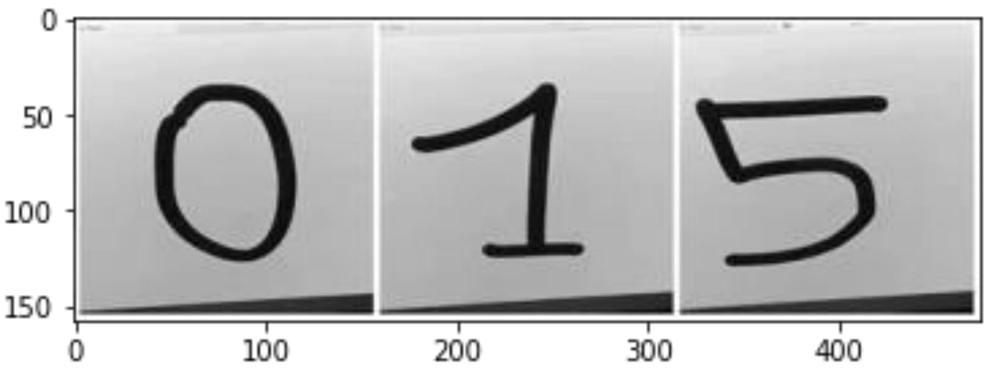
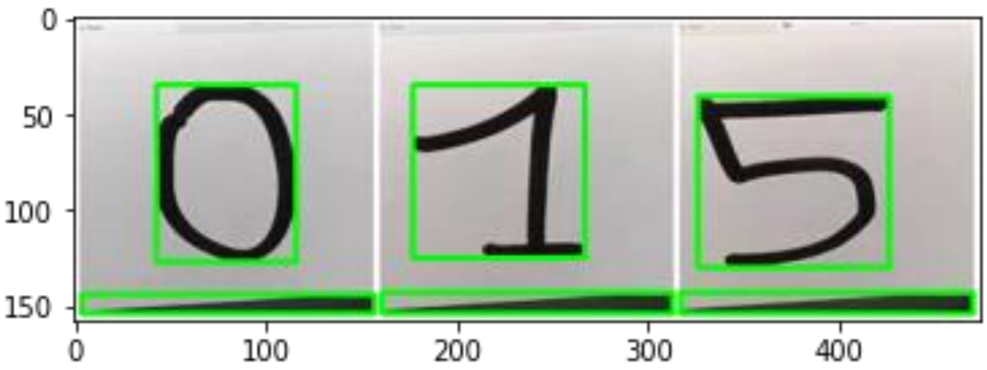
```
UPLOAD_FOLDER = 'C:\\Users\\Samprokshana\\Desktop\\HandwrittenDigitRecognitio
n\\uploads'

app = Flask(__name__)

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def upload():

    if request.method == "POST":

        f = request.files["image"]

        filepath = secure_filename(f.filename)

        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filepath))

        upload_img = os.path.join(UPLOAD_FOLDER, filepath)

        fname=os.path.basename(upload_img)

        imag=cv2.imread(fname)
        grey=cv2.cvtColor(image.copy(),cv2.COLOR_BGR2GRAY)
        plt.imshow(grey,cmap='gray')
        plt.show()
```



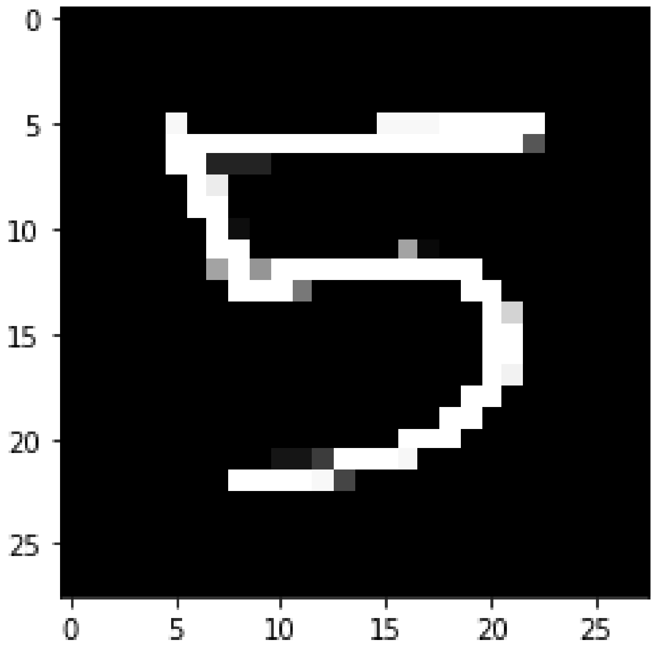
```
      ret,thresh=cv2.threshold(grey.copy(),75,255,cv2.THRESH_BINARY_INV)
      contours,_=cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_A
      PPROX_SIMPLE)
      preprocessed_digits=[]

      for c in contours:
        x,y,w,h=cv2.boundingRect(c)
        cv2.rectangle(image,(x,y),(x+w,y+h),color=(0,255,0),thickness=2)
        digit=thresh[y:y+h,x:x+w]
        resized_digit=cv2.resize(digit,(18,18))

        padded_digit=np.pad(resized_digit,((5,5),(5,5)),"constant",constant_v
      alues=0)
```

```
        preprocessed_digits.append(padded_digit)
        preprocessed_digits.append(padded_digit)
        print("\n\n\n------------PREPROCESSED DIGIT--------------")
        plt.imshow(image,cmap="gray")

         plt.show()
```

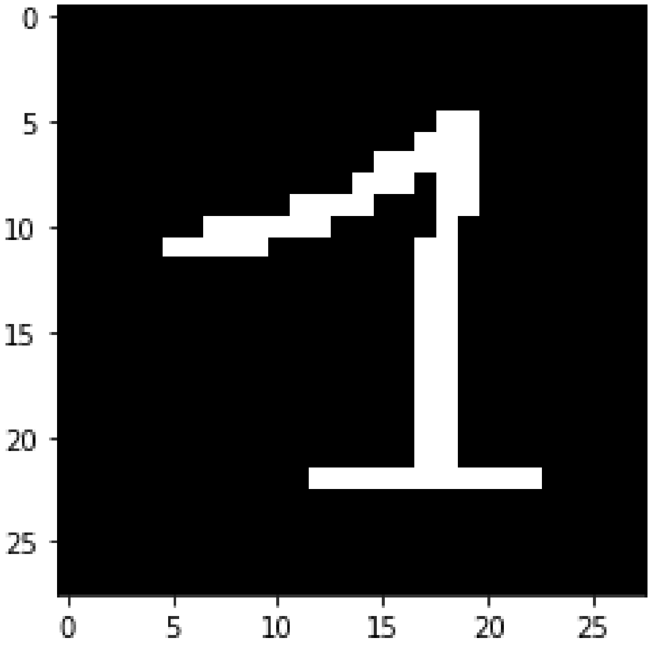
```
      ------------PREPROCESSED DIGIT--------------
```



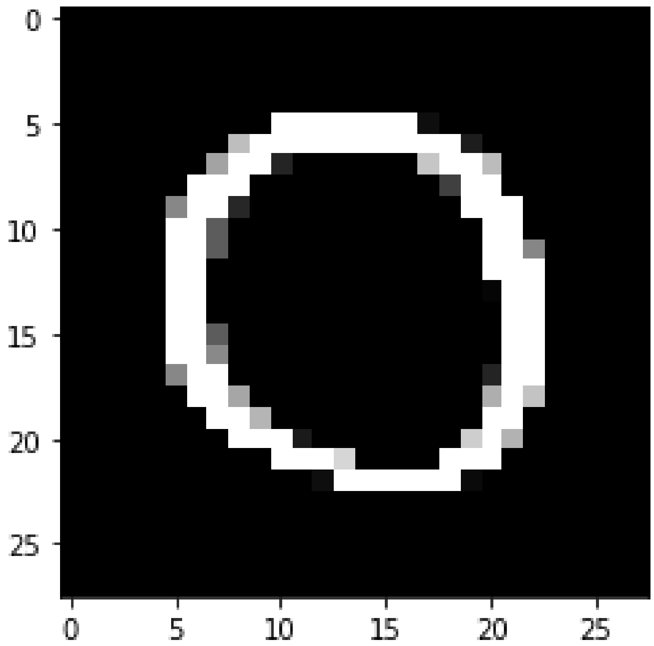
```
inp=np.array(preprocessed_digits)
for digit in preprocessed_digits:
  prediction=model.predict(digit.reshape(1,28,28,1))
  print("\n\n\n------------------------\n\n")
  plt.imshow(digit.reshape(28,28),cmap="gray")
  plt.show()
  print("\n\nFINAL OUTPUT  {}".format(np.argmax(prediction)))
  print("\nsoftmax\n{}".format(prediction))
  hard_maxed_prediction=np.zeros(prediction.shape)
  hard_maxed_prediction[0][np.argmax(prediction)]=1
  print("\nhardmax\n{}".format(hard_maxed_prediction))
  print("\n\n---------------------\n\n")
```



```
FINAL OUTPUT  5

softmax
[[1.7827598e-16 4.7495518e-11 1.8994897e-17 3.3779909e-08 8.7917082e-15
  1.0000000e+00 1.6201093e-09 8.1360667e-13 3.5875628e-12 3.7106801e-21]]

hardmax
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]


---------------------


1/1 [==============================] - 0s 19ms/step



-------------------------
```



```
FINAL OUTPUT  1



softmax
[[7.2860418e-10 9.8727173e-01 5 8.3070006e-03 8.5445590e-06 4.3615545e-03
  3.0203847e-12 1.0111049e-14 5.1127081e-0 1.3068305e-11 6.3537592e-11]]

hardmax
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]


---------------------


1/1 [==============================] - 0s 29ms/step



-------------------------
```



```
FINAL OUTPUT  0

softmax
[[1.0000000e+00 3.6947175e-22 4.7702914e-17 3.9116349e-17 1.0729757e-20
  1.9394410e-14 9.3705656e-19 1.1059809e-19 8.9281733e-17 8.1152592e-15]]

hardmax
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

---------------------
```