ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN - ĐIỆN TỬ

TIỂU LUẬN MÔN
THỊ GIÁC MÁY

ĐỀ TÀI

NHẬN DẠNG CHỮ SỐ TRÊN THẺ TÍN DỤNG

GV hướng dẫn: NGUYỄN ĐỨC THÀNH

Nhóm sinh viên thực hiện:

Họ và tên	*MSSV*
Hồ Văn Cón	1610326
Nguyễn Phan Hải Phú	1612612
Trương Trí Lạc	1611736
Nguyễn Hữu Toàn	1613598

TP. Hồ Chí Minh, 2019

Mục lục

Danh sách hình vẽ

Listings

Lời Nói Đầu

Thị giác máy tính trong những năm gần đây phát triển với tốc độ rất nhanh. Nó được ứng dụng hầu như trên mỗi lĩnh vực của đời sống hằng ngày của chúng ta. Nó bao gồm các ứng dụng từ đơn giản đến phức tạp bằng cách kết hợp giữa các thuật toán cũng như là các phần cứng chẳng hạn như camera,...

Nhờ vào sự phát triển nhanh chống đó, công việc của chúng ta trở nên một cách đơn giản hơn khá nhiều trong các nghành sản xuất, tiêu dùng, dự đoán thị trường chứng khoáng, ngân hàng,... Và đề tài của nhóm em cũng không nằm ngoại lệ đó là nhận dạng số thẻ được ghi trên thẻ tín dụng.

Trong đề tài này, nhóm chúng em tập trung nhận dạng trên các loại thẻ phổ biến như Mastercard, Visa hay American Express,... Ngoài ra các thẻ khác như JCB, HSBC,... sẽ được nhóm nghiên cứu trong thời gian tới.

Bằng cách kết hợp giữa các thuật toán về xử lý ảnh cũng như thuật toán KNN trong Machine Learning đã giải quyết cụ thể yêu cầu của tiểu luận.

Đây là một nghiên cứu nhỏ trong việc tiếp cận về trí tuệ nhân tạo mà nhóm đặt ra. Đó cũng là thành quả đầu mà nhóm đạt được và tiểu luận này cũng giúp được một phần nào cho việc nghiên cứu xâu hơn về lĩnh vực Trí tuệ nhân tạo và Computer Science.

Nhóm 7

Lời Cảm Ơn

1 Nhận diện chữ số trên thẻ Mastercard

1.1 Đề bài

Nhận dạng chữ số trên thẻ tín dụng Mastercard

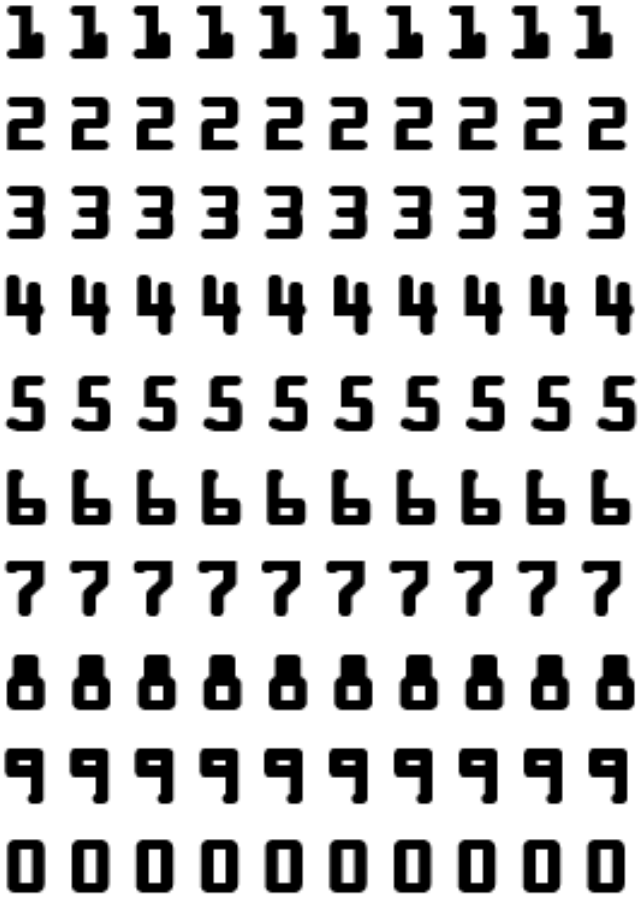
Hình 1: Thẻ tín dụng Mastercard

1.2 Mô hình nhận dạng

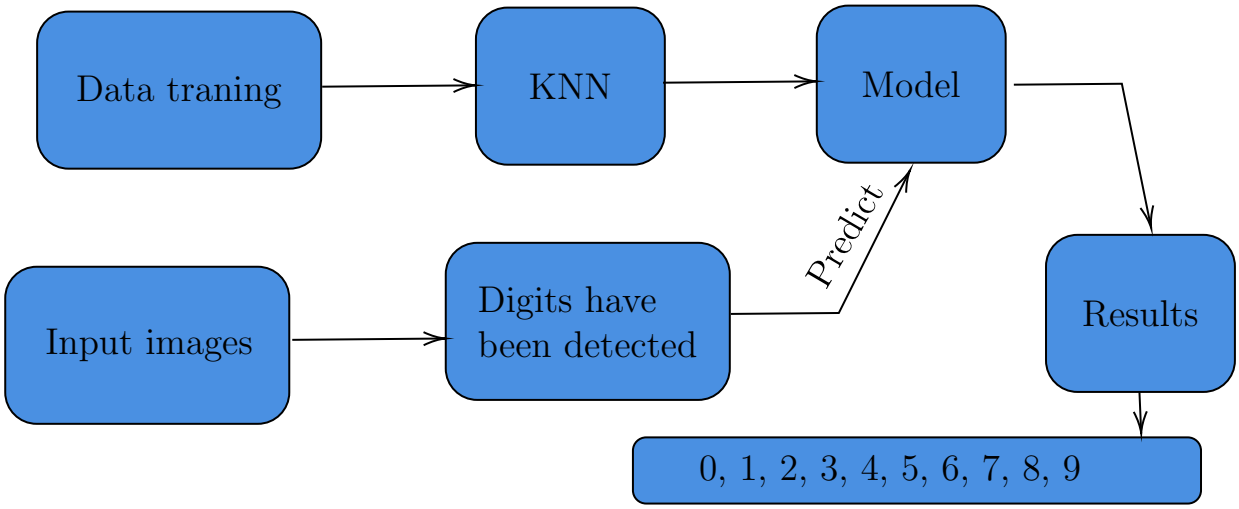
Vì thẻ Mastercard có ký tự chữ số khác với chữ số thường nên ta sẽ thực hiện training với tập dữ liệu riêng.

Ta lên mạng gõ tìm font chữ của thẻ mastercard sau đó cài đặt, vào word gõ 100 ký tự số từ 0 đến 1 và chụp màng hình.

Dữ liệu training là ảnh dưới đây.



Hình 2: ảnh training Mastercard



Hình 3: Lưu đồ nhận dạng chữ số trên thẻ Mastercard

1.3 Chương trình nhận dạng

```
import numpy as np
#from scipy.misc.pilutil import imresize
import cv2
from skimage.feature import hog
from matplotlib import pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
from sklearn import datasets
from skimage import exposure
import imutils
from imutils import contours

#Khai bao ham xu li pixel de nhan dang
def pixels_to_hog_20(img_array):
    hog_featuresData = []
    for img in img_array:
        fd = hog(img,
                 orientations=10,
                 pixels_per_cell=(5,5),
                 cells_per_block=(1,1),
                 visualize=False)
        hog_featuresData.append(fd)
    hog_features = np.array(hog_featuresData, 'float64')
    return np.float32(hog_features)

#Tao class KNN_MODEL
class KNN_MODEL():
    def __init__(self, k = 3):
        self.k = k
        self.model = cv2.ml.KNearest_create()

    def train(self, samples, responses):
        self.model.train(samples, cv2.ml.ROW_SAMPLE, responses)

    def predict(self, samples):
        retval, results, neigh_resp, dists = self.model.findNearest
   (samples, self.k)
        return results.ravel()

#Khai bao ham xu li anh, nhan dang anh
out_image = {}
def proc_user_img(img_file, model):
    print('loading "%s for digit recognition" ...' % img_file)
    im = cv2.imread(img_file)
    im = imutils.resize(im,width=300)
    imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    rectKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9, 3))
    sqKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    tophat = cv2.morphologyEx(imgray, cv2.MORPH_TOPHAT, rectKernel)
```

```
 gradX = cv2.Sobel(tophat, ddepth=cv2.CV_32F, dx=1, dy=0,ksize
=-1)
 gradX = np.absolute(gradX)
 (minVal, maxVal) = (np.min(gradX), np.max(gradX))
 gradX = (255 * ((gradX - minVal) / (maxVal - minVal)))
 gradX = gradX.astype("uint8")
 gradX = cv2.morphologyEx(gradX, cv2.MORPH_CLOSE, rectKernel)

 thresh = cv2.threshold(gradX, 0, 255,cv2.THRESH_BINARY | cv2.
THRESH_OTSU)[1]
 thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, sqKernel)

 #Phat giac duong bien de dong khung
 _,contours,hierarchy = cv2.findContours(thresh.copy(), cv2.
RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
 locs = []

 #Loai bo cac vung de chon duoc 4 vung anh co 4 so tren ma the
 for i,c in enumerate(contours):
     (x,y,w,h) = cv2.boundingRect(c)
     ar = w/float(h)

     if ar > 2.5 and ar < 4.0:
         if (w > 40 and w < 55) and (h > 13 and h < 20):
             locs.append((x, y, w, h))

 locs = sorted(locs, key=lambda x:x[0])
 output = []
 #Dua vao toa do duong bien cac vung ta thuc hien nhan dang tung
 vung
 j = 1 #Bien dem vong for
 for i, (gX, gY, gW, gH) in enumerate(locs):
     k = 0 #Bien dem vong for nho tu 0 - 3
     groupOutput = []
     group = imgray[gY - 3:gY + gH + 3, gX - 3:gX + gW + 3]
     group = cv2.threshold(group, 0, 255,cv2.THRESH_BINARY | cv2
.THRESH_OTSU)[1]
     _,contours,hierarchy = cv2.findContours(group.copy(), cv2.
RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

     #Thuc hien nhan dang tung anh
     for i,c in enumerate(contours):
         (x,y,w,h) = cv2.boundingRect(c)

         im_digit = group[y-1:y+h+1,x-1:x+w+1]
         im_digit = (255-im_digit)
         im_digit = cv2.resize(im_digit,(57,88))
         hog_img_data = pixels_to_hog_20([im_digit])

         kq = model.predict(hog_img_data)
         string = str(int(kq[0]))
         groupOutput.append(string)
```

```
            out_image[str(j*4 -1 - k)] = im_digit
            k = k +1

        j = j +1
        #Ghi ket qua len cac anh do
        cv2.rectangle(im,(gX - 5, gY - 5),(gX + gW + 5, gY + gH +
   5), (255, 0, 255), 2)
        cv2.putText(im, "".join(groupOutput[::-1]), (gX, gY - 15),
   cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 0, 255), 2)
        output.extend(groupOutput)

    return im

#Khai bao ham lay so tu anh tham chieu (reference)
def get_digits(contours, hierarchy):
    hierarchy = hierarchy[0]
    bounding_rectangles = [cv2.boundingRect(ctr) for ctr in
   contours]
    final_bounding_rectangles = []
    u, indices = np.unique(hierarchy[:,-1], return_inverse=True)
    most_common_heirarchy = u[np.argmax(np.bincount(indices))]
    for r,hr in zip(bounding_rectangles, hierarchy):
        x,y,w,h = r
        if ((w*h)>250) and (10 <= w <= 200) and (10 <= h <= 200)
   and hr[3] == most_common_heirarchy:
            final_bounding_rectangles.append(r)
    return final_bounding_rectangles

#Khai bao ham sap xep ccacso tham chieu trong anh va danh so thu tu
def get_contour_precedence(contour, cols):
    return contour[1] * cols + contour[0]

#Khai bao ham load anh tham chieu va xu li
def load_digits_custom(img_file):
    train_data = []
    train_target = []
    start_class = 1
    im = cv2.imread(img_file)
    imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(imgray,11,255,cv2.THRESH_BINARY_INV)[1]

    _,contours,hierarchy = cv2.findContours(thresh.copy(), cv2.
   RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    digits_rectangles = get_digits(contours,hierarchy)
    digits_rectangles.sort(key=lambda x:get_contour_precedence(x,
   im.shape[1]))

    for index,rect in enumerate(digits_rectangles):
        x,y,w,h = rect
        cv2.rectangle(im,(x-8,y-8),(x+w+8,y+h+8),(0,255,0),2)
        im_digit = thresh[y:y+h,x:x+w]
```

```
            im_digit = (255-im_digit)

            im_digit = cv2.resize(im_digit,(57,88))
            train_data.append(im_digit)
            train_target.append(start_class%10)

            if index>0 and (index+1) % 10 == 0:
                start_class += 1

    return np.array(train_data), np.array(train_target)

#------------------chuan bi du lieu-----------------
#Anh tham chieu va anh nhan dang
TRAIN_IMG = 'images\ocr-reference-1.png'
DETECT_IMG = 'images\mastercard.png'

digits, labels = load_digits_custom(TRAIN_IMG)

#In thong tin anh tham chieu
print('train data shape',digits.shape)
print('test data shape',labels.shape)

#Thuc hien sap xep ngau nhien va chuan bi du lieu de training
digits, labels = shuffle(digits, labels, random_state=256)  #Xao
   tron du lieu
train_digits_data = pixels_to_hog_20(digits)
X_train, X_test, y_train, y_test = train_test_split(
   train_digits_data, labels, test_size=0.7)

#------------------training va nhan dang anh-----------------

#Thuc hien training va in ket qua % do chinh xac
model = KNN_MODEL(k = 5)
model.train(X_train, y_train)
preds = model.predict(X_test)
print('Accuracy: ',accuracy_score(y_test, preds))

#Thuc hien nhan dang anh
model = KNN_MODEL(k = 5)
model.train(train_digits_data, labels)
im = proc_user_img(DETECT_IMG, model)

#------------------xuat ket qua-----------------

#Xuat ket qua ra dang anh va imshow
cv2.imwrite("results\Ket_qua_mastercard.png",im)
cv2.namedWindow("Ket_qua_mastercard",cv2.WINDOW_AUTOSIZE)
cv2.imshow("Ket_qua_mastercard", im)

#Xuat tung so truoc khi nhan dang ra plot show
titles = {}
photo = {}
```

```
for so in range(0,16):
    titles[str(so)] = str(so)
    photo[str(so)] = out_image[str(so)]

for i in range(0,16):
  plot.subplot(4,4,i+1), plot.imshow(photo[str(i)],'gray')
  plot.title(titles[str(i)])
  plot.xticks([]), plot.yticks([])
plot.show()

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

Listing 1: Recognition Mastercard

1.4 Đánh giá kết quả và kết luận

- Đầu tiên ta load hai ảnh gồm ảnh tham chiếu và ảnh thẻ tín dụng

```
    #Anh tham chieu va anh nhan dang
    TRAIN_IMG = 'images\ocr-reference-1.png'
    DETECT_IMG = 'images\mastercard.png'
    digits, labels = load_digits_custom(TRAIN_IMG)
```

- Hàm load_digits_custom để xử lí cắt 100 số trong ảnh tham chiếu ra và thực hiện xử lí nó thành mảng các số
- Sau đó ta test nó bằng các xáo trộn dữ liệu, thực hiện xử lí và đưa ra phần trăm chính xác

```
    #Thuc hien sap xep ngau nhien va chuan bi du lieu de training
    digits, labels = shuffle(digits, labels, random_state=256)  #
   Xao tron du lieu
    train_digits_data = pixels_to_hog_20(digits)
    X_train, X_test, y_train, y_test = train_test_split(
   train_digits_data, labels, test_size=0.7)

    #------------------training va nhan dang anh----------------
    #Thuc hien training va in ket qua % do chinh xac
    model = KNN_MODEL(k = 5)
    model.train(X_train, y_train)
    preds = model.predict(X_test)
    print('Accuracy: ',accuracy_score(y_test, preds))
```

- Sau đó thực hiện nhận dạng ảnh

```
    #Thuc hien nhan dang anh
    model = KNN_MODEL(k = 5)
    model.train(train_digits_data, labels)
    im = proc_user_img(DETECT_IMG, model)
```

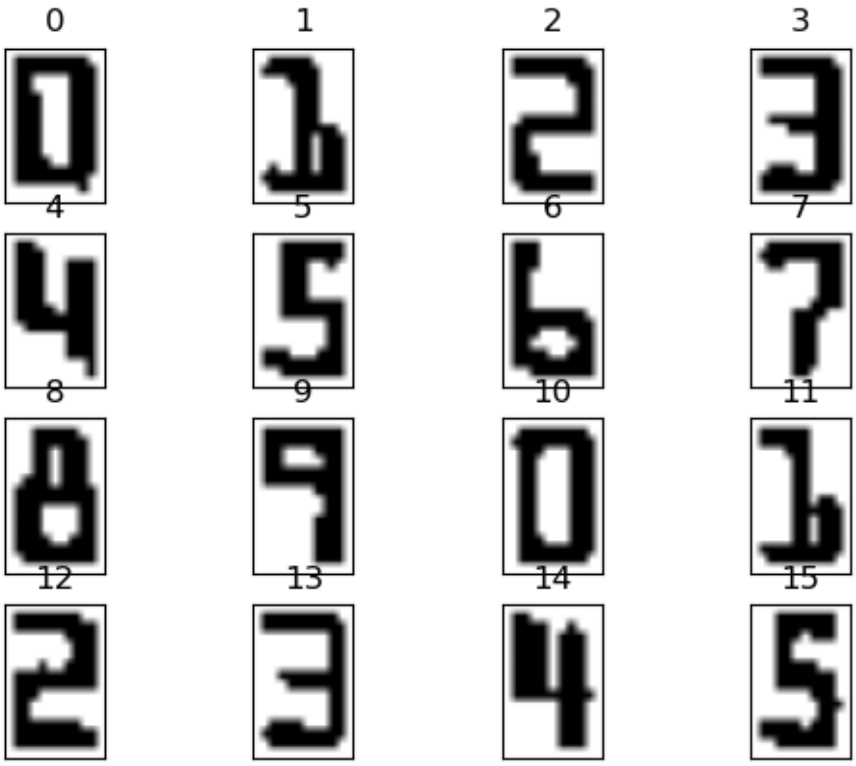
- Hàm proc_user_img() để tiền xử lí ảnh nhận dạng và nhận dạng nó
- Sau khi nhận dạng xong ta thực hiện imshow từng số trước nhận dạng và kết quả, đồng thời cũng xuất ảnh kết quả ra file .png

```
    #Xuat ket qua ra dang anh va imshow
```

```
cv2.imwrite("results\Ket_qua_mastercard.png",im)
cv2.namedWindow("Ket_qua_mastercard",cv2.WINDOW_AUTOSIZE)
cv2.imshow("Ket_qua_mastercard", im)

#Xuat tung so truoc khi nhan dang ra plot show
titles = {}
photo = {}
for so in range(0,16):
    titles[str(so)] = str(so)
    photo[str(so)] = out_image[str(so)]

for i in range(0,16):
  plot.subplot(4,4,i+1), plot.imshow(photo[str(i)],'gray')
  plot.title(titles[str(i)])
  plot.xticks([]), plot.yticks([])
plot.show()

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

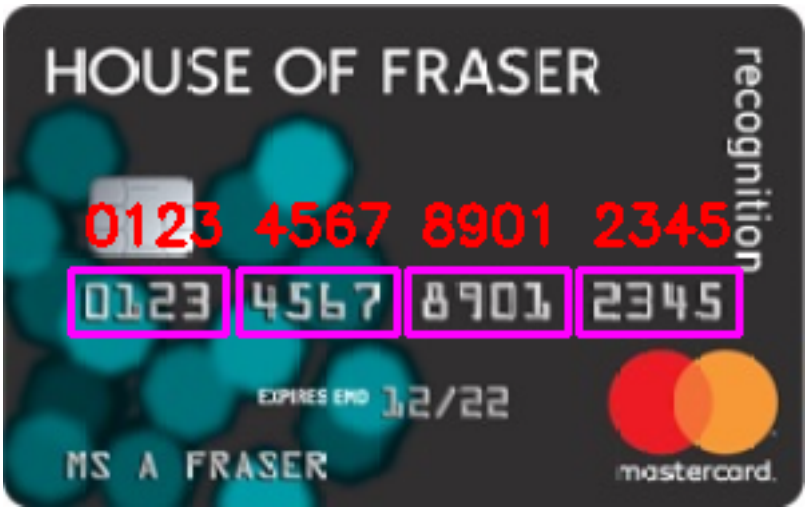
KẾT QUẢ NHẬN ĐƯỢC

- Các chử số trên thẻ sau khi được detect



Hình 4: Các chử số sau khi phát hiện trên thẻ Mastercard

- Kết quả cuối cùng sau khi nhận dạng thẻ Mastercard

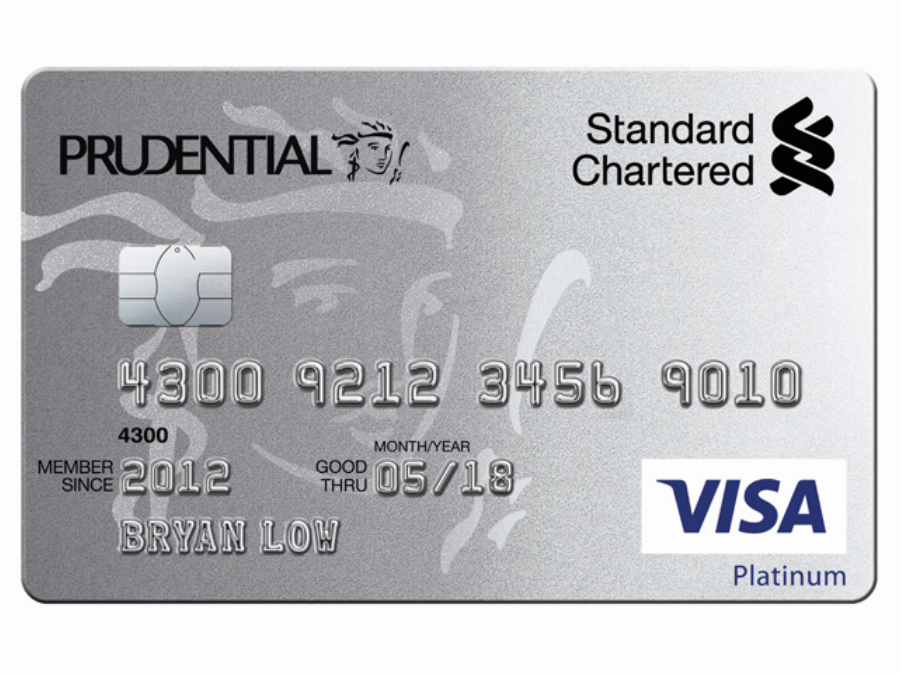


Hình 5: Kết quả sau khi nhận dạng Mastercard

2 Nhận diện chữ số trên thẻ Visa

2.1 Đề bài

Nhận dạng chữ số trên thẻ tín dụng Visa

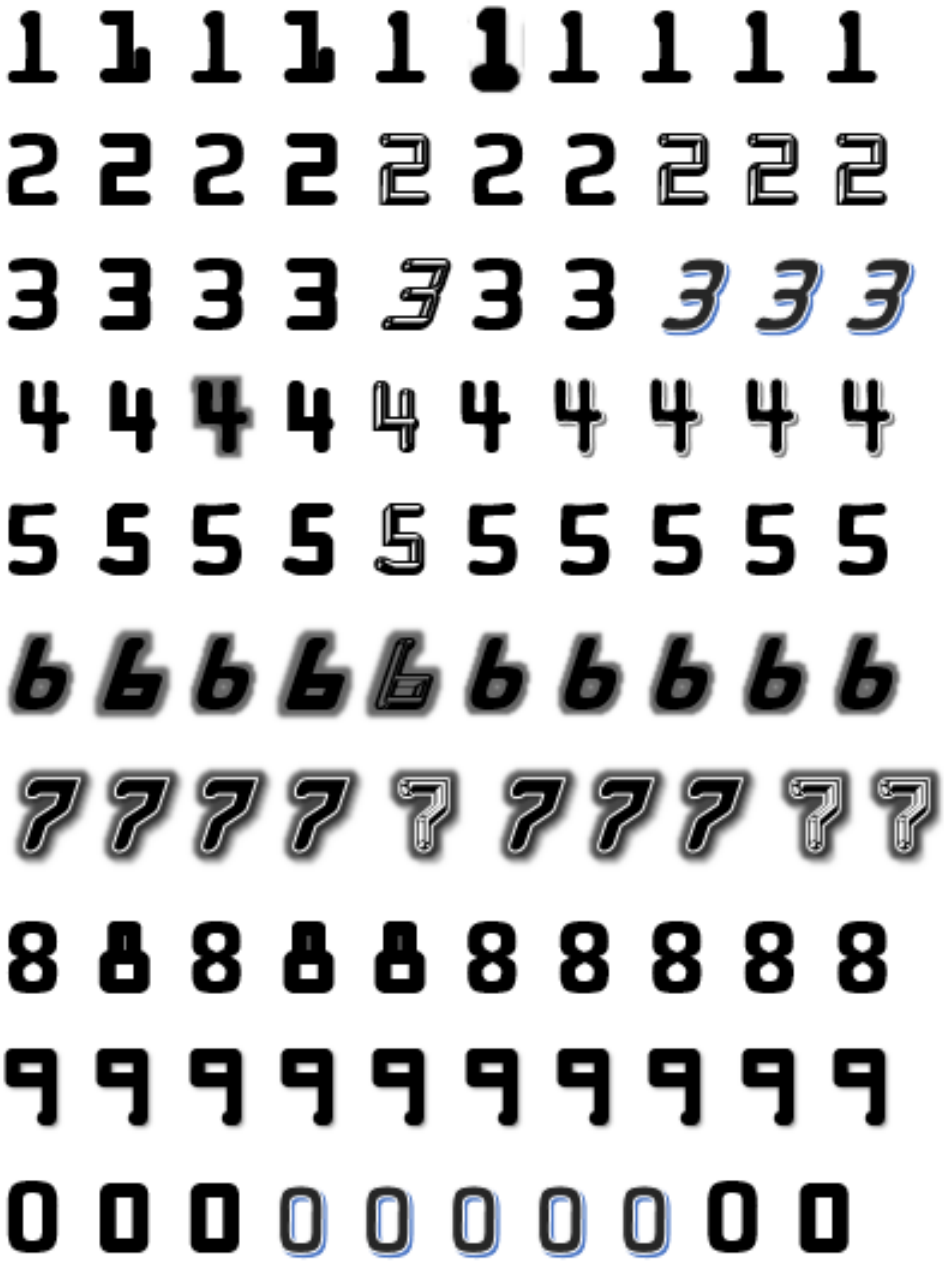


2.2 Mô hình nhận dạng

Vì thẻ Visa có ký tự chữ số khác với chữ số thường nên ta sẽ thực hiện training với tập dữ liệu riêng.

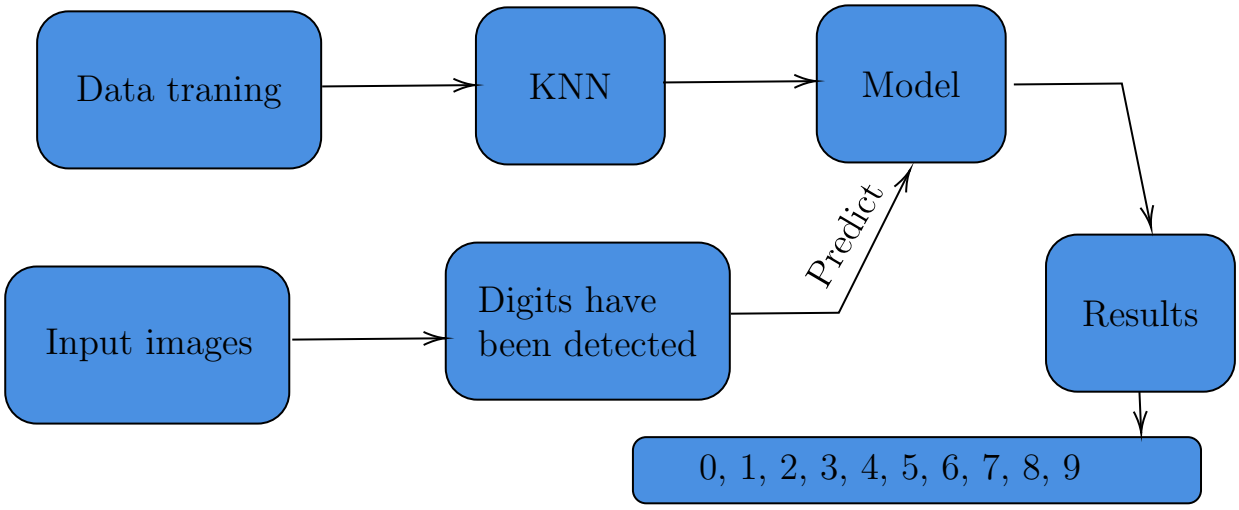
Ta lên mạng gõ tìm font chữ của thẻ visa sau đó cài đặt, vào word gõ 100 ký tự số từ 0 đến 1 và chụp màng hình.

Dữ liệu training là ảnh dưới đây.



Hình 6: ảnh training Visa

- Vì lí do ảnh Visa có nền màu trắng khi lấy nhị phân ra ta qua nhiều giai đoạn xử lí nên các số không được nguyên bản, không được đẹp nên ta sẽ thực hiện biến đổi ngẫu nhiên các số từ font chữ góc (in nghiêng, in đậm, bóng mờ, phóng to, thu nhỏ, thêm số từ font khác,. . .) và chọn kết quả cảm thấy tốt nhất



Hình 7: Lưu đồ nhận dạng chữ số trên thẻ Visa

2.3 Chương trình nhận dạng

```
import numpy as np
#from scipy.misc.pilutil import imresize
import cv2
from skimage.feature import hog
from matplotlib import pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
from sklearn import datasets
from skimage import exposure
import imutils
from imutils import contours

#Khai bao ham cat anh
cropped = {}
def crop(im, height, width, k):

    imgwidth = im.shape[1]
    imgheight = im.shape[0]
    for i in range(0,imgheight,height):
        for j in range(0,imgwidth,width):
            box = (j, i, j+width, i+height)
            a = im[i: i+height,j: j+width ]
            cropped[str(k)] = a
            img = ~a
            k +=1
    return k

#Khai bao ham xu li pixel de nhan dang
def pixels_to_hog_20(img_array):
    hog_featuresData = []
```

```
        for img in img_array:
            fd = hog(img,
                     orientations=10,
                     pixels_per_cell=(5,5),
                     cells_per_block=(1,1),
                     visualize=False)
            hog_featuresData.append(fd)
        hog_features = np.array(hog_featuresData, 'float64')
        return np.float32(hog_features)

    #Tao class KNN_MODEL
    class KNN_MODEL():
        def __init__(self, k = 3):
            self.k = k
            self.model = cv2.ml.KNearest_create()

        def train(self, samples, responses):
            self.model.train(samples, cv2.ml.ROW_SAMPLE, responses)

        def predict(self, samples):
            retval, results, neigh_resp, dists = self.model.
    findNearest(samples, self.k)
            return results.ravel()

    #Khai bao ham xu li anh, nhan dang anh
    def proc_user_img(img_file, model):
        print('loading "%s for digit recognition" ...' % img_file)
        im = cv2.imread(img_file)
        im = imutils.resize(im,width=300)
        imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)

        #Tien xu li anh
        rectKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9,
    5))
        sqKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5)
    )
        tophat = cv2.morphologyEx(imgray, cv2.MORPH_TOPHAT,
    rectKernel)
        gradX = cv2.Sobel(tophat, ddepth=cv2.CV_32F, dx=1, dy=0,
    ksize=-1)
        gradX = np.absolute(gradX)
        (minVal, maxVal) = (np.min(gradX), np.max(gradX))
        gradX = (255 * ((gradX - minVal) / (maxVal - minVal)))
        gradX = gradX.astype("uint8")
        gradX = cv2.morphologyEx(gradX, cv2.MORPH_CLOSE, rectKernel
    )

        thresh = cv2.threshold(gradX, 0, 255,cv2.THRESH_BINARY |
    cv2.THRESH_OTSU)[1]
        thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, sqKernel
    )
```

```
        #Phat giac duong bien de dong khung
        _,contours,hierarchy = cv2.findContours(thresh.copy(), cv2.
    RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        locs = []

        #Loai bo cac vung de chon duoc 4 vung anh co 4 so tren ma
    the
        for i,c in enumerate(contours):
            (x,y,w,h) = cv2.boundingRect(c)
            ar = w/float(h)
            if ar > 2.5 and ar < 4.0:
                if (w > 40 and w < 55) and (h > 13 and h < 20):
                    locs.append((x, y, w, h))

        #Sap xep cac vung
        locs = sorted(locs, key=lambda x:x[0])
        output = []
        k = 0
        u = 0

        #Dua vao toa do duong bien cac vung ta thuc hien nhan dang
    tung vung
        for i, (gX, gY, gW, gH) in enumerate(locs):
            k = k +1
            groupOutput = []

            imae = cv2.imread("images\Visa.png",0)
            binary = cv2.adaptiveThreshold(imae+9,255,cv2.
    ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,121,9)
            blurred = cv2.medianBlur(binary,1)
            blurred = imutils.resize(blurred, width=300)


            if(k==4):
                group =  blurred[gY - 3:gY + gH + 3, gX - 5:gX + gW
     + 3]
            else:
                group =  blurred[gY - 3:gY + gH + 3, gX - 3:gX + gW
     + 3]

            group2 = cv2.resize(group,(120,60))

            #Thuc hien cat 4 anh trong tung vung
            u = crop(group2, 60, 30, u)

            #Thuc hien nhan dang tung anh
            for digitROI in range(u-4,u,1):
                im_digit = cropped[str(digitROI)]

                im_digit = (255-im_digit)
                im_digit = cv2.resize(im_digit,(57,88))
```

```
                hog_img_data = pixels_to_hog_20([im_digit])

                kq = model.predict(hog_img_data)
                if (digitROI==14):
                    print(str(int(kq[0])))
                string = str(int(kq[0]))

                groupOutput.append(string)

            #Ghi ket qua len cac anh do
            cv2.rectangle(im,(gX - 5, gY - 5),(gX + gW + 5, gY + gH
 + 5), (0, 0, 255), 2)
            cv2.putText(im, "".join(groupOutput), (gX, gY - 15),cv2
.FONT_HERSHEY_SIMPLEX, 0.65, (255, 0, 0), 2)
            output.extend(groupOutput)

        return im

 #Khai bao ham lay so tu anh tham chieu (reference)
 def get_digits(contours, hierarchy):
        hierarchy = hierarchy[0]
        bounding_rectangles = [cv2.boundingRect(ctr) for ctr in
contours]
        final_bounding_rectangles = []
        u, indices = np.unique(hierarchy[:,-1], return_inverse=True
)
        most_common_heirarchy = u[np.argmax(np.bincount(indices))]
        for r,hr in zip(bounding_rectangles, hierarchy):
            x,y,w,h = r
            if ((w*h)>250) and (10 <= w <= 200) and (10 <= h <=
200) and hr[3] == most_common_heirarchy:
                final_bounding_rectangles.append(r)
        return final_bounding_rectangles

 #Khai bao ham sap xep ccacso tham chieu trong anh va danh so
thu tu
 def get_contour_precedence(contour, cols):
        return contour[1] * cols + contour[0]

 #Khai bao ham load anh tham chieu va xu li
 def load_digits_custom(img_file):
        train_data = []
        train_target = []
        start_class = 1
        im = cv2.imread(img_file)
        imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
        thresh = cv2.threshold(imgray,11,255,cv2.THRESH_BINARY_INV)
[1]

        _,contours,hierarchy = cv2.findContours(thresh.copy(), cv2.
RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        digits_rectangles = get_digits(contours,hierarchy)
```

```
        digits_rectangles.sort(key=lambda x:get_contour_precedence(
    x, im.shape[1]))

        for index,rect in enumerate(digits_rectangles):
            x,y,w,h = rect
            cv2.rectangle(im,(x-8,y-8),(x+w+8,y+h+8),(0,255,0),2)
            im_digit = thresh[y:y+h,x:x+w]
            im_digit = (255-im_digit)
            im_digit = cv2.resize(im_digit,(57,88))
            train_data.append(im_digit)
            train_target.append(start_class%10)

            if index>0 and (index+1) % 10 == 0:
                start_class += 1

        return np.array(train_data), np.array(train_target)

    #------------------chuan bi du lieu
    ----------------------------------------

    #Anh tham chieu va anh nhan dang
    TRAIN_IMG = 'images\ocr-reference-2.png'
    DETECT_IMG = "images\Visa.png"

    digits, labels = load_digits_custom(TRAIN_IMG)

    #In thong tin anh tham chieu
    print('train data shape',digits.shape)
    print('train label shape',labels.shape)

    #Thuc hien sap xep ngau nhien va chuan bi du lieu de training
    digits, labels = shuffle(digits, labels, random_state=256) #Xao
    tron du lieu
    train_digits_data = pixels_to_hog_20(digits)
    X_train, X_test, y_train, y_test = train_test_split(
    train_digits_data, labels, test_size=0.7)

    #------------------training va nhan dang anh
    ----------------------------------------

    #Thuc hien training va in ket qua % do chinh xac
    model = KNN_MODEL(k = 3)
    model.train(X_train, y_train)
    preds = model.predict(X_test)
    print('Accuracy: ',accuracy_score(y_test, preds))

    #Thuc nhan dang anh
    model = KNN_MODEL(k = 5)
    model.train(train_digits_data, labels)
    im = proc_user_img(DETECT_IMG, model)

    #------------------xuat ket qua
```

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

    #Xuat ket qua ra dang anh va imshow
    cv2.imwrite("results\Ket_qua_visa.png",im)
    cv2.namedWindow("Ket_qua_visa",cv2.WINDOW_AUTOSIZE)
    cv2.imshow("Ket_qua_visa", im)

    #Xuat tung so truoc khi nhan dang ra plot show
    titles = {}
    photo = {}
    for so in range(0,16):
        titles[str(so)] = str(so)
        photo[str(so)] = cropped[str(so)]

    for i in range(0,16):
        plot.subplot(4,4,i+1), plot.imshow(photo[str(i)],'gray')
        plot.title(titles[str(i)])
        plot.xticks([]), plot.yticks([])
    plot.show()

    cv2.waitKey()
```

Listing 2: Recognition Visa

2.4 Đánh giá kết quả và kết luận

- Đầu tiên ta load hai ảnh gồm ảnh tham chiếu và ảnh thẻ tín dụng

```
    #Anh tham chieu va anh nhan dang
    TRAIN_IMG = 'images\ocr-reference-2.png'
    DETECT_IMG = 'images\visa.png'
    digits, labels = load_digits_custom(TRAIN_IMG)
```

- Hàm load_digits_custom để xử lí cắt 100 số trong ảnh tham chiếu ra và thực hiện xử lí nó thành mảng các số
- Sau đó ta test nó bằng các xáo trộn dữ liệu, thực hiện xử lí và đưa ra phần trăm chính xác

```
    #Thuc hien sap xep ngau nhien va chuan bi du lieu de training
    digits, labels = shuffle(digits, labels, random_state=256)  #
Xao tron du lieu
    train_digits_data = pixels_to_hog_20(digits)
    X_train, X_test, y_train, y_test = train_test_split(
train_digits_data, labels, test_size=0.7)

    #------------------training va nhan dang anh----------------
    #Thuc hien training va in ket qua % do chinh xac
    model = KNN_MODEL(k = 3)
    model.train(X_train, y_train)
    preds = model.predict(X_test)
    print('Accuracy: ',accuracy_score(y_test, preds))
```

- Sau đó thực hiện nhận dạng ảnh

```
    #Thuc hien nhan dang anh
```

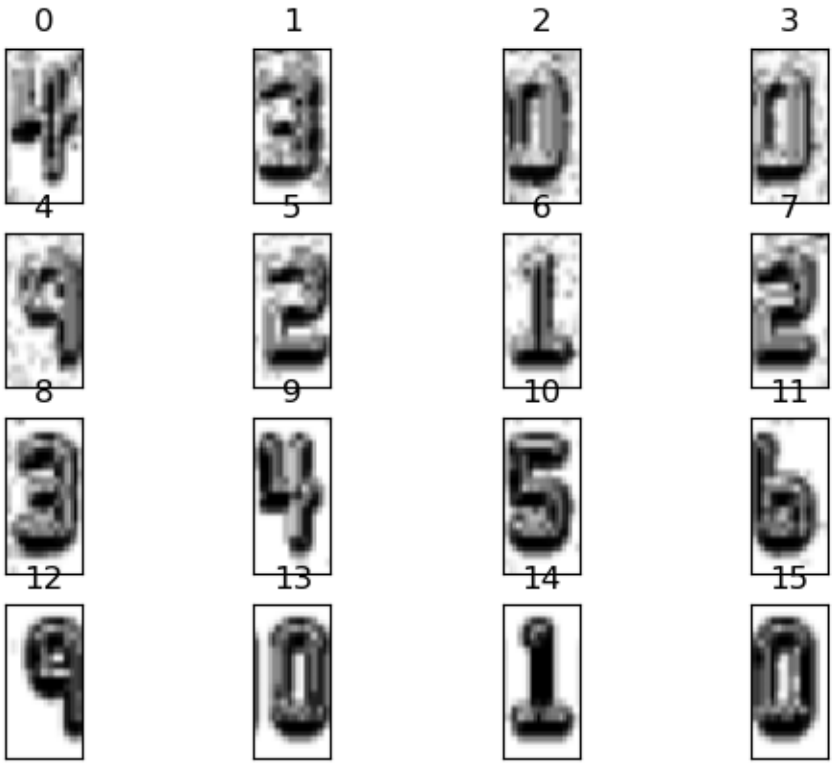
```
model = KNN_MODEL(k = 5)
model.train(train_digits_data, labels)
im = proc_user_img(DETECT_IMG, model)
```

- Hàm proc_user_img() để tiền xử lí ảnh nhận dạng và nhận dạng nó
- Sau khi nhận dạng xong ta thực hiện imshow từng số trước nhận dạng và kết quả, đồng thời cũng xuất ảnh kết quả ra file .png

```
#Xuat ket qua ra dang anh va imshow
cv2.imwrite("results\Ket_qua_visa.png",im)
cv2.namedWindow("Ket_qua_visa",cv2.WINDOW_AUTOSIZE)
cv2.imshow("Ket_qua_visa", im)

#Xuat tung so truoc khi nhan dang ra plot show
titles = {}
photo = {}
for so in range(0,16):
    titles[str(so)] = str(so)
    photo[str(so)] = out_image[str(so)]

for i in range(0,16):
  plot.subplot(4,4,i+1), plot.imshow(photo[str(i)],'gray')
  plot.title(titles[str(i)])
  plot.xticks([]), plot.yticks([])
plot.show()

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

KẾT QUẢ NHẬN ĐƯỢC

- Các chử số trên thẻ sau khi được detect



Hình 8: Các chử số sau khi phát hiện trên thẻ Visa

- Kết quả cuối cùng sau khi nhận dạng thẻ Visa

Hình 9: Kết quả sau khi nhận dạng thẻ Visa

3 Nhận diện chữ số trên thẻ American Express

3.1 Đề bài

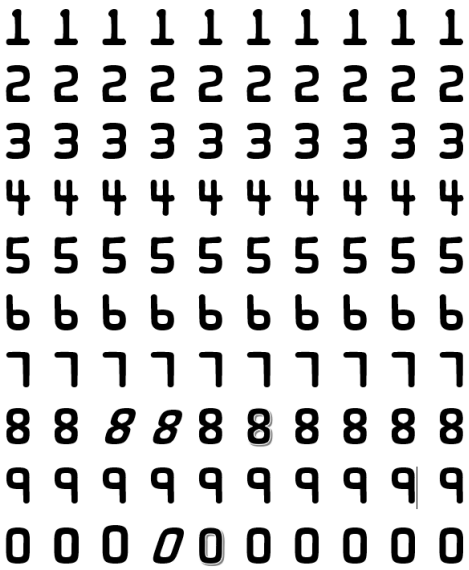
Nhận dạng chữ số trên thẻ tín dụng American Express

3.2 Mô hình nhận dạng

Vì thẻ American Express có ký tự chữ số khác với chữ số thường nên ta sẽ thực hiện training với tập dữ liệu riêng.

Ta lên mạng gõ tìm font chữ của thẻ American Express sau đó cài đặt, vào word gõ 100 ký tự số từ 0 đến 1 và chụp màng hình.

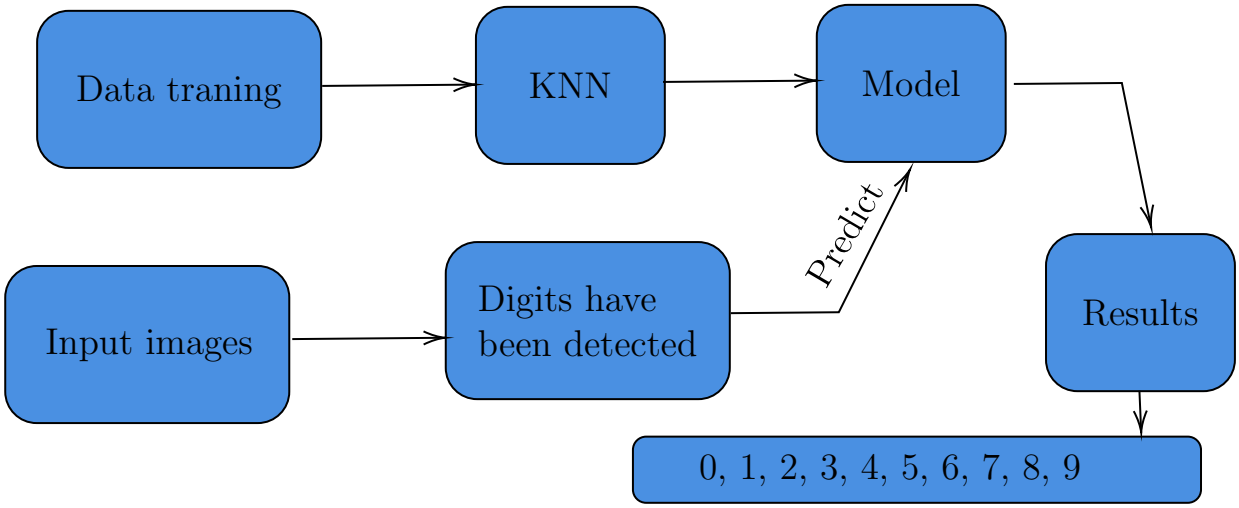
Dữ liệu training là ảnh dưới đây.



Hình 10: ảnh training American Express

- Vì lí do ảnh Visa có nền màu trắng khi lấy nhị phân ra ta qua nhiều giai đoạn xử lí nên các số không được nguyên bản, không được đẹp nên ta sẽ thực hiện biến

đổi ngẫu nhiên các số từ font chữ góc (in nghiêng, in đậm, bóng mờ, phóng to, thu nhỏ, thêm số từ font khác,...) và chọn kết quả cảm thấy tốt nhất



Hình 11: Lưu đồ nhận dạng chữ số trên thẻ Visa

3.3 Chương trình nhận dạng

```
import numpy as np
#from scipy.misc.pilutil import imresize
import cv2
from skimage.feature import hog
from matplotlib import pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
from sklearn import datasets
from skimage import exposure
import imutils
from imutils import contours
from sklearn import svm
from sklearn.metrics import accuracy_score


#Khai bao ham xu li pixel de nhan dang
def pixels_to_hog_20(img_array):
    hog_featuresData = []
    for img in img_array:
        fd = hog(img,
                 orientations=10,
                 pixels_per_cell=(5,5),
                 cells_per_block=(1,1),
                 visualize=False)
        hog_featuresData.append(fd)
    hog_features = np.array(hog_featuresData, 'float64')
```

```
    return np.float32(hog_features)

#Tao class KNN_MODEL
class KNN_MODEL():
    def __init__(self, k):
        self.k = k
        self.model = cv2.ml.KNearest_create()

    def train(self, samples, responses):
        self.model.train(samples, cv2.ml.ROW_SAMPLE, responses)

    def predict(self, samples):
        retval, results, neigh_resp, dists = self.model.findNearest
   (samples, self.k)
        return results.ravel()

#Khai bao ham xu li anh, nhan dang anh
out_image = {}
def proc_user_img(img_file, model):
    print('loading "%s for digit recognition" ...' % img_file)
    im = cv2.imread(img_file)
    im = imutils.resize(im,width=300)
    imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    rectKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9, 3))
    sqKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    tophat = cv2.morphologyEx(imgray, cv2.MORPH_TOPHAT, rectKernel)
    gradX = cv2.Sobel(tophat, ddepth=cv2.CV_32F, dx=1, dy=0,ksize
   =-1)
    gradX = np.absolute(gradX)
    (minVal, maxVal) = (np.min(gradX), np.max(gradX))
    gradX = (255 * ((gradX - minVal) / (maxVal - minVal)))
    gradX = gradX.astype("uint8")
    gradX = cv2.morphologyEx(gradX, cv2.MORPH_CLOSE, rectKernel)

    thresh = cv2.threshold(gradX, 0, 255,cv2.THRESH_BINARY | cv2.
   THRESH_OTSU)[1]
    thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, sqKernel)

    #Phat giac duong bien de dong khung
    _,contours,hierarchy = cv2.findContours(thresh.copy(), cv2.
   RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    locs = []

    #Loai bo cac vung de chon duoc 4 vung anh co 4 so tren ma the
    for i,c in enumerate(contours):
        (x,y,w,h) = cv2.boundingRect(c)
        ar = w/float(h)

        if ar > 2.5 and ar < 5:
            if (w > 40 and w < 80) and (h > 10 and h < 20):
                locs.append((x, y, w, h))
```

```
    locs = sorted(locs, key=lambda x:x[0])
    output = []
    dem = 0
    #Dua vao toa do duong bien cac vung ta thuc hien nhan dang tung
    vung
    j = 1 #Bien dem vong for
    for i, (gX, gY, gW, gH) in enumerate(locs):
        k = 0 #Bien dem vong for nho tu 0 - 3
        groupOutput = []
        group = imgray[gY - 3:gY + gH + 3, gX - 3:gX + gW + 3]
        group = cv2.threshold(group, 0, 255,cv2.THRESH_BINARY | cv2
.THRESH_OTSU)[1]
        _,contours,hierarchy = cv2.findContours(group.copy(), cv2.
RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
        count = i

        #Thuc hien nhan dang tung anh
        for i,c in enumerate(contours):
            (x,y,w,h) = cv2.boundingRect(c)
            if (w > 2):
                im_digit = group[y-1:y+h+1,x-1:x+w+1]
                im_digit = (255-im_digit)
                im_digit = cv2.resize(im_digit,(57,88))
                hog_img_data = pixels_to_hog_20([im_digit])
                kq = model.predict(hog_img_data)
                string = str(int(kq))
                groupOutput.append(string)
                if (count == 0):
                    out_image[str(3 - dem)] = im_digit
                    t = 3 - dem
                if (count == 1):
                    out_image[str(13 - dem)] = im_digit
                    t = 13 - dem
                if (count == 2):
                    out_image[str(24 - dem)] = im_digit
                    t = 24 - dem
                k = k +1
                dem = dem + 1

        j = j +1
        #Ghi ket qua len cac anh do
        cv2.rectangle(im,(gX - 5, gY - 5),(gX + gW + 5, gY + gH +
5), (255, 0, 255), 2)
        cv2.putText(im, "".join(groupOutput[::-1]), (gX, gY - 15),
cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 0, 255), 2)
        output.extend(groupOutput)

    return im

#Khai bao ham lay so tu anh tham chieu (reference)
def get_digits(contours, hierarchy):
    hierarchy = hierarchy[0]
```

```
    bounding_rectangles = [cv2.boundingRect(ctr) for ctr in
   contours]
    final_bounding_rectangles = []
    u, indices = np.unique(hierarchy[:,-1], return_inverse=True)
    most_common_heirarchy = u[np.argmax(np.bincount(indices))]
    for r,hr in zip(bounding_rectangles, hierarchy):
        x,y,w,h = r
        if ((w*h)>250) and (10 <= w <= 200) and (10 <= h <= 200)
   and hr[3] == most_common_heirarchy:
            final_bounding_rectangles.append(r)
    return final_bounding_rectangles

#Khai bao ham sap xep ccacso tham chieu trong anh va danh so thu tu
def get_contour_precedence(contour, cols):
    return contour[1] * cols + contour[0]

#Khai bao ham load anh tham chieu va xu li
def load_digits_custom(img_file):
    train_data = []
    train_target = []
    start_class = 1
    im = cv2.imread(img_file)
    imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(imgray,11,255,cv2.THRESH_BINARY_INV)[1]

    _,contours,hierarchy = cv2.findContours(thresh.copy(), cv2.
   RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    digits_rectangles = get_digits(contours,hierarchy)
    digits_rectangles.sort(key=lambda x:get_contour_precedence(x,
   im.shape[1]))

    for index,rect in enumerate(digits_rectangles):
        x,y,w,h = rect
        cv2.rectangle(im,(x-8,y-8),(x+w+8,y+h+8),(0,255,0),2)
        im_digit = thresh[y:y+h,x:x+w]
        im_digit = (255-im_digit)

        im_digit = cv2.resize(im_digit,(57,88))
        train_data.append(im_digit)
        train_target.append(start_class%10)

        if index>0 and (index+1) % 10 == 0:
            start_class += 1

    return np.array(train_data), np.array(train_target)

#------------------chuan bi du lieu----------------------

#Anh tham chieu va anh nhan dang
TRAIN_IMG = 'images\ocr-reference-3.png'
DETECT_IMG = 'images\American_Express.png'
```

```
digits, labels = load_digits_custom(TRAIN_IMG)
#In thong tin anh tham chieu
print('train data shape',digits.shape)
print('test data shape',labels.shape)

#Thuc hien sap xep ngau nhien va chuan bi du lieu de training
digits, labels = shuffle(digits, labels, random_state=256)  #Xao
   tron du lieu
train_digits_data = pixels_to_hog_20(digits)
X_train, X_test, y_train, y_test = train_test_split(
   train_digits_data, labels, test_size=0.7)

#------------------training va nhan dang anh----------------
#Thuc hien training va in ket qua % do chinh xac
model = KNN_MODEL(k = 3)
model.train(X_train, y_train)
preds = model.predict(X_test)
print('Accuracy: ',accuracy_score(y_test, preds))

#Thuc hien nhan dang anh
model = KNN_MODEL(k = 5)
model.train(train_digits_data, labels)
im = proc_user_img(DETECT_IMG, model)

print(np.array(out_image).shape)

#------------------xuat ket qua----------------------

#Xuat ket qua ra dang anh va imshow
cv2.imwrite("results\Ket_qua_AmericanExpress.png",im)
cv2.namedWindow("Ket_qua_AmericanExpress",cv2.WINDOW_AUTOSIZE)
cv2.imshow("Ket_qua_AmericanExpress", im)

#Xuat tung so truoc khi nhan dang ra plot show
titles = {}
photo = {}
for so in range(0,15):
    titles[str(so)] = str(so)
    photo[str(so)] = out_image[str(so)]

for i in range(0,15):
  plot.subplot(4,4,i+1), plot.imshow(photo[str(i)],'gray')
  plot.title(titles[str(i)])
    plot.xticks([]), plot.yticks([])

plot.show()
cv2.waitKey()
cv2.destroyAllWindows()
```

Listing 3: Recognition American Express

3.4 Đánh giá kết quả và kết luận

- Đầu tiên ta load hai ảnh gồm ảnh tham chiếu và ảnh thẻ tín dụng

```
#Anh tham chieu va anh nhan dang
TRAIN_IMG = 'images\ocr-reference-3.png'
DETECT_IMG = 'images\American_Express.png'
digits, labels = load_digits_custom(TRAIN_IMG)
```

- Hàm load_digits_custom để xử lí cắt 100 số trong ảnh tham chiếu ra và thực hiện xử lí nó thành mảng các số
- Sau đó ta test nó bằng các xáo trộn dữ liệu, thực hiện xử lí và đưa ra phần trăm chính xác

```
#Thuc hien sap xep ngau nhien va chuan bi du lieu de training
digits, labels = shuffle(digits, labels, random_state=256)  #
Xao tron du lieu
train_digits_data = pixels_to_hog_20(digits)
X_train, X_test, y_train, y_test = train_test_split(
train_digits_data, labels, test_size=0.7)

#------------------training va nhan dang anh----------------
#Thuc hien training va in ket qua % do chinh xac
model = KNN_MODEL(k = 3)
model.train(X_train, y_train)
preds = model.predict(X_test)
print('Accuracy: ',accuracy_score(y_test, preds))
```

- Sau đó thực hiện nhận dạng ảnh

```
#Thuc hien nhan dang anh
model = KNN_MODEL(k = 5)
model.train(train_digits_data, labels)
im = proc_user_img(DETECT_IMG, model)
```

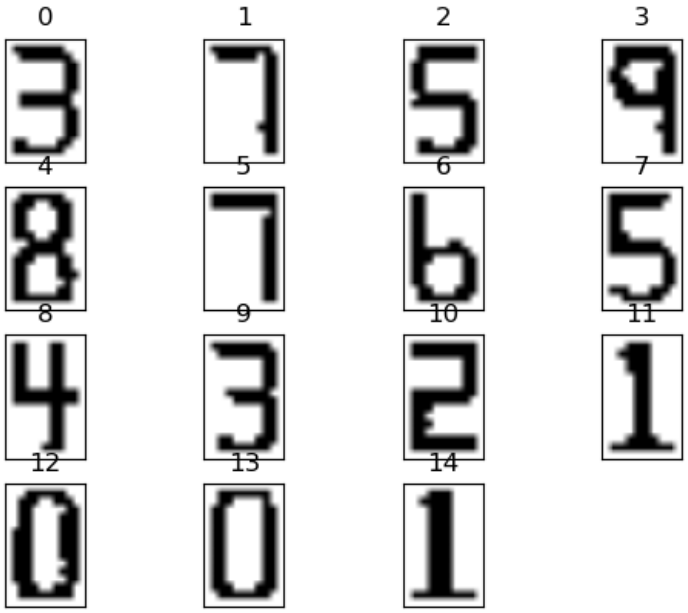
- Hàm proc_user_img() để tiền xử lí ảnh nhận dạng và nhận dạng nó
- Sau khi nhận dạng xong ta thực hiện imshow từng số trước nhận dạng và kết quả, đồng thời cũng xuất ảnh kết quả ra file .png

```
#Xuat ket qua ra dang anh va imshow
cv2.imwrite("results\Ket_qua_AmericanExpress.png",im)
cv2.namedWindow("Ket_qua_AmericanExpress",cv2.WINDOW_AUTOSIZE)
cv2.imshow("Ket_qua_AmericanExpress", im)

#Xuat tung so truoc khi nhan dang ra plot show
titles = {}
photo = {}
for so in range(0,15):
    titles[str(so)] = str(so)
    photo[str(so)] = out_image[str(so)]

for i in range(0,15):
    plot.subplot(4,4,i+1), plot.imshow(photo[str(i)],'gray')
    plot.title(titles[str(i)])
    plot.xticks([]), plot.yticks([])
```

```
plot.show()
```

KẾT QUẢ NHẬN ĐƯỢC

- Các chử số trên thẻ sau khi được detect



Hình 12: Các chử số sau khi phát hiện trên thẻ American Express

- Kết quả cuối cùng sau khi nhận dạng thẻ American Express

Hình 13: Kết quả sau khi nhận dạng thẻ American Express