پروژه پایانی درس بینایی کامپیوتر

مریم عظیم‌پور

۹۵۵۲۱۳۳۳

آموزش مدل

در فایل model_trainer طراحی شبکه یادگیری عمیق و مراحل آموزش آن پیاده‌سازی شده‌است.

استخراج پروپوزال، تابع get_proposals_for_train():

برای استخراج پروپوزال‌های مناسب برای پلاک، با توجه به ویژگی‌های ظاهری پلاک مانند داشتن خطوط عمودی نزدیک به هم، مستطیلی بودن پلاک و ...، تابع get_proposals_for_train پیاده سازی شده‌است. این تابع در فایل get_proposal.py قرار دارد.

```
def get_proposals_for_train(img,l,IMG_WIDTH = 200,IMG_HEIGHT = 200):

  sobel_filter = np.array([[-1,0,1],
                           [-2,0,2],
                           [-1,0,1]],dtype=np.float32)
  open1SE = np.ones((5,1),np.uint8)
  closeSE = np.ones((1,23),np.uint8)
  open2SE = np.ones((1,20),np.uint8)

  label = []
  proposals = []

  blur = cv2.bilateralFilter(img,5,75,75)

  gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)

  clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(10,10))
  gray = clahe.apply(gray)

  sobel = cv2.filter2D(np.float32(gray),-1,sobel_filter)
  abs = np.absolute(sobel)
  thresh = hysteresis_threshold(abs)
  open1 = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, open1SE)
  close = cv2.morphologyEx(open1, cv2.MORPH_CLOSE, closeSE)
  open2 = cv2.morphologyEx(close, cv2.MORPH_OPEN, open2SE)
  open2 = np.array(open2.tolist(),dtype= np.uint8)
```

در این تابع ابتدا نویزهای تصویر با استفاده از تابع آماده cv2.bilaterialFilter() تا حدودی رفع میشوند. این تابع برای رفع نویزهایی مانند دانه‌های برف و باران که از دسته نویزهای نمک و فلفل هستند مناسب است و در عین حال لبه‌ها را تضعیف نمی‌کند.

(منبع: https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

https://docs.opencv.org/master/dd/d6a/tutorial_js_filtering.html)

در مرحله بعدی با استفاده از روش کلاهه، کنتراست تصویر بهبود داده شده‌است تا لبه‌های عمودی بهتر تشخیص داده‌شوند. سپس برای تشخیص لبه‌ها و خطوط عمودی، تصویر با فیلتر sobel کانوالو شده و سپس با استفاده از آستانه گذاری دوسطحی، تصویر باینری شده‌است. تابع hysteresis_threshold() تابعی برای آستانه گذاری دو سطحی است و اعداد مناسب برای آستانه‌های آن با آزمون و خطا به دست آمده‌اند.

```
def hysteresis_threshold(image):
    lowThreshold=70
    highThreshold=90

    M, N = image.shape
    result = np.zeros((M,N), dtype=np.float32)

    for i in range(1,M-1):
      for j in range(1,N-1):
        if image[i][j] >= highThreshold:
          result[i][j] = 1
        elif image[i][j] < lowThreshold:
          result[i][j] = 0
        else:
          result[i][j] = 0
          for m in range(i-1, i+2):
            for n in range(j-1, j+2):
              if result[m][n] == 1 or image[m][n] >= highThreshold:
                result[i][j] = 1
                break

    return result
```

با باینری کردن تصویر، نواحی‌ای که مشتق کمی داشتند و خطوط ضعیفی بودند، حذف میشوند. پس از آستانه‌گذاری، برای حذف نواحی اضافه باقی‌مانده و لبه‌های اریب، از عملگر open با عنصر ساختاری open1SE استفاده شده‌است. سپس برای اتصال خطوط عمودی مربوط به اعداد پلاک، از عملگر close با عنصر ساختاری closeSE استفاده شده و در نهایت برای حذف خطوط عمودی اضافه مجددا از عملگر open با عنصر ساختاری open2SE استفاده شده‌است. اندازه مناسب برای عنصرهای ساختاری با آزمون و خطا به دست آمده‌است.

نمونه‌هایی از تصاویر پس از این عملیات به صورت زیر هستند:

در مرحله‌ی بعد، با استفاده از cv2.findContours() نواحی متصل به دست آمده‌اند. سپس نواحی یافت شده از بزرگ به کوچک مرتب شده‌اند و برای این که از بین بزرگترین ناحیه‌ها، ناحیه‌ای که پلاک در آن قرار دارد مشخص شود، از توصیفگر شکل extent که برابر نسبت مساحت کانتور به مساحت box دربرگیرنده کانتور است، استفاده شده‌است. از توصیفگرهای دیگری مانند صلب بودن، کشیدگی و aspect ratio نیز استفاده شد اما نتیجه بهتری به دست نیامد. در این مرحله، ناحیه‌ای که area/rect_area بزرگتری داشته‌باشد، به عنوان پروپوزال انتخاب میشود و اگر هیچ ناحیه‌ای در تصویر پیدا نشد، کل تصویر به عنوان پروپوزال انتخاب میشود. همچنین لیبل پروپوزال برابر لیبل تصویر اصلی گرفته شده‌است.

(منابع: https://docs.opencv.org/master/dd/d49/tutorial_py_contour_features.html

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/)

```
cnts,_ = cv2.findContours(open2.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
cnts = sorted(cnts, key=len,reverse=True)

if len(cnts)>0:
  x1,y1,w1,h1 = cv2.boundingRect(cnts[0])

  if len(cnts)>1:

    area = cv2.contourArea(cnts[0])
    rect_area = w1*h1
    n1 = area/rect_area

    x2,y2,w2,h2 = cv2.boundingRect(cnts[1])
    area = cv2.contourArea(cnts[1])
    rect_area = w2*h2
    n2 = area/rect_area


    if n1 > n2 or w2<0.6*w1 or h2<0.4*h1:
      resized = cv2.resize(img[y1:y1+h1+1 , x1:x1+w1+1], (IMG_WIDTH,IMG_HEIGHT), interpolation = cv2.INTER_AREA)
      proposals.append(resized)
      label.append(1)

    else:
      resized = cv2.resize(img[y2:y2+h2+1 , x2:x2+w2+1], (IMG_WIDTH,IMG_HEIGHT), interpolation = cv2.INTER_AREA)
      proposals.append(resized)
      label.append(1)

  else:
    resized = cv2.resize(img[y1:y1+h1+1 , x1:x1+w1+1], (IMG_WIDTH,IMG_HEIGHT), interpolation = cv2.INTER_AREA)
    proposals.append(resized)
    label.append(1)

else:
  if l != 0:
    resized = cv2.resize(img, (IMG_WIDTH,IMG_HEIGHT), interpolation = cv2.INTER_AREA)
    proposals.append(resized)
    label.append(1)

return proposals,label
```

این روش برای تصاویری که قسمت‌های زیادی از خودرو و محیط اطراف آن نیز در تصویر وجود دارند و به ویژه تصاویری که مربوط به پلاک جلوی ماشین‌ها هستند، دارای خطا میباشد و در برخی از این تصاویر، ناحیه‌ای به جز پلاک به عنوان پروپوزال انتخاب میشود. دلیل این اتفاق این است که در اینگونه تصاویر، به جز پلاک ناحیه‌های دیگری نیز با خطوط عمودی کنار هم وجود دارند که ممکن است الگوریتم این نواحی را پلاک تشخیص دهد.

نمونه‌ای از پروپوزال‌های تشخیص داده‌شده توسط این الگوریتم به صورت زیر هستند:

دیتای آموزش:

برای تهیه‌ی داده‌های آموزش، ابتدا فایل Dataset.7z اکسترکت شده‌است.

```
#extract Dataset
!pip install py7zr
import py7zr
archive = py7zr.SevenZipFile('Dataset.7z', mode='r')
archive.extractall(path="/content/Dataset")
archive.close()
```

سپس هرکدام از تصاویر دیتاست خوانده شده و در متغیر image ذخیره شده‌است و سپس با استفاده از تابع get_proposals_for_train() پروپوزال تصویر و لیبل مربوط به آن به دست آمده‌است. در نهایت پروپوزال و لیبل تمام تصاویر دیتاست به ترتیب در متغیرهای x_train و y_train ذخیره شده‌اند.

```
DATA_PATH = '/content/Dataset/'

classes = ['0','1','2']

x_train = []
y_train = []
for cl in classes:
  directory_name = os.path.join(DATA_PATH, cl)
  for filename in os.listdir(directory_name):
    filename_jpg = filename[:-4] + '.jpg'
    filename_str = os.path.join(directory_name, filename_jpg)
    image = mpimg.imread(os.path.join(filename_str))
    x_t,y_t = get_proposals_for_train(image,int(cl))

    x_t = np.squeeze(x_t)
    x_train.append(x_t)
    y_train.append(y_t)
```

در مرحله بعد، لیبل‌ها به فرمت مناسب برای شبکه تبدیل شده و از پروپوزال‌های به دست آمده، data generator ساخته شده‌است.

(منبع: https://keras.io/api/preprocessing/image/)

```
x_train = np.array(x_train)
y_train = keras.utils.to_categorical(y_train, n_classes)

train_datagen = ImageDataGenerator()
train_generator = train_datagen.flow(
    x_train,
    y_train,
    batch_size=BATCH_SIZE,
    shuffle=True,
)
```

معماری مدل:

برای این پروژه از شبکه resnet50 استفاده شده‌است که دارای یک لایه global average pooling، یک لایه فولی کانکتد با ۸۴ نورون و یک لایه دیگر با ۳ نورون به عنوان لایه نهایی است. بین این لایه‌ها از dropout استفاده شده تا از overfit شبکه تا حدی جلوگیری شود.

برای آموزش این شبکه از تابع خطای categorical_crossentropy و از اپتیمایزر adam استفاده شده‌است و شبکه با data generator ساخته شده از پروپوزال‌ها آموزش دیده‌است.

پس از آموزش شبکه، ساختار مدل به همراه وزن‌های به دست‌آمده در فایل trainedModel.h5 ذخیره شده‌است.

(منبع: https://medium.com/swlh/saving-and-loading-of-keras-sequential-and-functional-models-73ce704561f4

https://androidkt.com/how-to-save-and-load-model-weights-in-keras/)

```
def build_model(n):
  base_model = keras.applications.resnet50.ResNet50(
      include_top=False,
      weights=None
  )
  x = base_model.output
  x = GlobalAveragePooling2D()(x)
  x = Dropout(0.4)(x)
  x = Dense(84, activation= 'sigmoid')(x)
  x = Dropout(0.4)(x)
  predictions = Dense(n, activation= 'softmax')(x)
  model = Model(inputs = base_model.input, outputs = predictions)
  return model

model = build_model(n_classes)
loss = 'categorical_crossentropy'
optimizer = 'adam'
model.compile(loss= loss, optimizer= optimizer, metrics='accuracy')
model.fit(train_generator, epochs=EPOCHS, batch_size=BATCH_SIZE,verbose=True)

model.save('trainedModel',save_format='h5')
```

بخشی از نتایج حاصل از فرآیند آموزش شبکه به صورت زیر است.

```
Epoch 20/30
55/55 [==============================] - 27s 485ms/step - loss: 0.3596 - accuracy: 0.8674
Epoch 21/30
55/55 [==============================] - 27s 485ms/step - loss: 0.3724 - accuracy: 0.8580
Epoch 22/30
55/55 [==============================] - 27s 486ms/step - loss: 0.3564 - accuracy: 0.8633
Epoch 23/30
55/55 [==============================] - 27s 485ms/step - loss: 0.3651 - accuracy: 0.8611
Epoch 24/30
55/55 [==============================] - 27s 486ms/step - loss: 0.3216 - accuracy: 0.8884
Epoch 25/30
55/55 [==============================] - 27s 488ms/step - loss: 0.3301 - accuracy: 0.8871
Epoch 26/30
55/55 [==============================] - 27s 487ms/step - loss: 0.3181 - accuracy: 0.8834
Epoch 27/30
55/55 [==============================] - 27s 486ms/step - loss: 0.3867 - accuracy: 0.8541
Epoch 28/30
55/55 [==============================] - 27s 486ms/step - loss: 0.3042 - accuracy: 0.8894
Epoch 29/30
55/55 [==============================] - 27s 485ms/step - loss: 0.3287 - accuracy: 0.8776
Epoch 30/30
55/55 [==============================] - 27s 485ms/step - loss: 0.3053 - accuracy: 0.8831
```

ارزیابی و تست

خواندن داده‌های تست:

برای تهیه داده‌های تست، ابتدا نام و آدرس تصاویر از فایل dataset.txt خوانده شده و هرکدام از تصاویر پس از لود شدن، به تابع get_proposals_for_test() داده شده‌اند تا پروپوزال آنها به دست آید.

(منبع: https://www.w3schools.com/python/python_file_open.asp)

```
import os
import random
import cv2
import keras
import numpy as np
import matplotlib.image as mpimg
import sklearn
from sklearn import metrics
import get_proposal

f = open("dataset.txt", "r")
data = []
for line in f:
  data.append(line.strip())

x_test = []
for i in data:
  image = mpimg.imread(os.path.join(i))
  x_te = get_proposal.get_proposals_for_test(image)
  x_test.append(x_te)
```

تابع get_proposals_for_test():

این تابع دقیقا مانند تابع get_proposals_for_train() است با این تفاوت که در این تابع لیبلی تولید نمی‌شود و خروجی تابع تنها پروپوزال مربوط به تصویر ورودی است. این تابع در فایل get_proposal.py قرار دارد.

دسته‌بندی داده‌های تست:

برای دسته‌بندی داده‌های تست، ابتدا مدل ذخیره شده در فاز آموزش، لود شده‌است و سپس خروجی شبکه برای هر تصویر در فایل predictions.txt ذخیره شده‌است.

(منبع: https://www.guru99.com/reading-and-writing-files-in-python.html#:~:text=Use%20the%20function%20open(%22filename,with%20read%20and%20write%20permissions.&text=Use%20the%20readlines%20function%20to,the%20file%20one%20by%20one.)

```
loaded_model = keras.models.load_model('trainedModel')
pred = loaded_model.predict(x_test)

p = open("predictions.txt","w+")
for i in range(len(data)):
  stri = data[i]+", " + str(np.argmax(pred[i]))
  p.write(stri+"\n")
```

ارزیابی با داده‌های دیتاست

برای ارزیابی شبکه، یک فایل dataset.txt با فرمت ذکر شده در داکیومنت پروژه ساخته شده‌است که نام ۱۰۰ تصویر رندم از هر کلاس در آن نوشته شده‌است.

```
#making test data

f = open("dataset.txt","w+")

data_path = '/content/Dataset/'
classes = ['0','1','2']
for cl in classes:
  dir_name = os.path.join(data_path, cl)
  filename = random.sample(os.listdir(dir_name),100)
  for i in filename:
    path = os.path.join(dir_name, i)
    f.write(path+"\n")
f.close()
```

پس از این کار، فایل dataset.txt خوانده شده و از تصاویر آن پس از لود شدن در متغیر image، پروپوزال استخراج شده و همچنین لیبل مربوط به تصاویر از آدرس آنها استخراج شده است (نام فولدری که تصویر در آن قرار دارد). سپس مدل ترین شده، لود شده است و با استفاده از آن خروجی شبکه برای هر تصویر به دست آمده‌است. در انتها با استفاده از sklearn.metrics.f1_score()، عملکرد الگوریتم ارزیابی شده‌است. دقت تقریبی این الگوریتم ۷۷درصد به دست آمده‌است.

```
#evaluate

f = open("dataset.txt", "r")
data = []
for line in f:
  data.append(line.strip())

x_test = []
y_true = []
for i in data:
  image = mpimg.imread(os.path.join(i))
  x_te = get_proposal.get_proposals_for_test(image)
  x_test.append(x_te)
  y_true.append(i[-10])

y_test = y_true
x_test = np.array(x_test)
x_test = np.squeeze(x_test)
y_true = keras.utils.to_categorical(y_true, n_classes)

loaded_model = keras.models.load_model('trainedModel')
pred = loaded_model.predict(x_test)
y_pred = []
p = open("predictions.txt","w+")
for i in range(len(data)):
  y_pred.append(str(np.argmax(pred[i])))
  stri = data[i]+", " + y_pred[i]
  p.write(stri+"\n")

import sklearn
from sklearn import metrics
sklearn.metrics.f1_score(y_test, y_pred, average='macro')
```

```
0.7774856506050819
```

این بخش در فایل model_trainer پیاده سازی شده‌است.