生成训练用数据集

目 录

[背景 2](#_Toc10571576)

[前言 2](#_Toc10571577)

[一、采集真实数据集 3](#_Toc10571578)

[二、利用GAN生成数据集 3](#_Toc10571579)

[三、利用图形处理方法生成数据集 3](#_Toc10571580)

[**3.1、生成alpha图** 3](#_Toc10571581)

[**3.2、提取前景图** 5](#_Toc10571582)

[**3.3、切割透明目标图** 6](#_Toc10571583)

[**3.4、融合图片** 6](#_Toc10571584)

[**3.4.1、修正大小** 6](#_Toc10571585)

[**3.4.2、修正亮度** 7](#_Toc10571586)

[**3.4.3、融合图片** 7](#_Toc10571587)

[**3.4.4、生成xml文件** 7](#_Toc10571588)

# 背景

如何提升[深度学习](http://lib.csdn.net/base/deeplearning)模型的效果，换一种问法如何提高模型的准确率，或者反过来问如果网络模型效果不好，该怎么办？

# 前言

参照《提高泛化能力的思路》，更多的有效数据集是提升模型性能的有效手段首选，因此寻求方法获取更多的有效训练用数据集。

据此，有以下几种思路：

* 采集真实数据集
* 利用生成对抗网络GAN生成数据集
* 利用图形处理方法生成数据集

# 一、采集真实数据集

通过深度神经网络训练出来的模型，其性能的验证指标即为在实际应用场景中的识别准确率。

为了提高准确率，在准备训练集时需要尽可能筛选出与实际场景相同或者至少相近的数据以便更好的提取特征，训练出适用的高性能模型。

据此，最好的数据集，应当是采集于应用场景的真实数据，目前局限于无法获取该类数据，暂定的方案是在系统上线后采集数据再进行加工后不端补充完善训练集。

因此该部分的内容暂时不做详述，保留待后续补充。

# 二、利用GAN生成数据集

通过GAN生成数据集是个比较新兴的思路，在之前的一段时间尝试了一种风格迁移的方案：《Deep Photo Style Transfer》，因为原方案对MATLAB依赖较大，因此实际尝试的是另外一种方案，即为Deep Photo Styletransfer的一种纯Tensorflow实现。

参考链接：<https://www.atyun.com/4396.html>

在Deep Photo Style Transfer这篇论文之前，基于深度学习的风格迁移的效果都是绘画风，即使输入是写实的照片，输出也会有一些失真的畸变。这篇论文引入了一种把风格迁移限制为局部区域色彩空间上仿射变换的约束条件，并把这种约束条件表示为一个完全可微的参数项，进而消除了输出图像的畸变。该论文的实现基于 Neural Style transfer by Gatys 。

这种方案的核心贡献主要体现在两个方面：

* 内容图的结构保留：风格迁移中有一个矛盾是我们希望能够有强烈的局部效果，同时又保持内容图的结构不带来几何形变。如我们希望图1中的窗户打开灯光，却不希望边缘扭曲。之前的一些算法可以解决特定图像的扭曲问题，该论文提出的把风格迁移限制在色彩空间上的仿射变换上的方法则是一种通用的解决方案。
* 在语义层级上精准迁移风格，使迁移后的图像真实合理：以往的算法常会遇到一个问题，把风格迁移到一些不合理的对象上，如把草地的绿色迁移给蓝天。该论文通过使用语义分割图做指导，把风格迁移局限在相同语义的区域上避免了这个问题。

实际操作过程中，发现该论文所提供的网络对风格图像和内容图像的环境约束较多，譬如两者最好都是建筑或都是人脸，其效果方能较好的体现。在以创研路作为风格图像而以爱情隧道作为内容图像进行试验的过程中，发现其迁移效果很不尽如人意。示例图片待补充。

# 三、利用图形处理方法生成数据集

上述GAN的方案在尝试过程中发现无法控制最终生成的数据向期望的方向靠拢，在一段时间内无法获取真实数据的前提下，考虑通过类似PS技术生成数据集的方案，该方案共分为以下几个步骤：

* 利用MATLAB生成alpha图，目标边界柔化；
* 利用python基于OpenCV根据alpha图生成无背景图；
* 利用python切割无背景图，提取目标；
* 将目标图融合进真实背景图。

**3.1、生成alpha图**

基于MATLAB 7进行批量处理图片生成alpha图。

注：同时尝试了另外一种基于深度学习的生成alpha图的方案，应用名称为：remove.bg，该方案无需标注，自动抠图，效果不尽如人意。

首先需要针对需要处理的图片进行简单的标注操作，如下图：

注：左图为原图，中间图为简单标注的图片，白色标注为需要保留的区域，黑色标注为需要作为背景处理的区域，右图为最终生成的alpha图：

**  **

**左图 中间图 右图**

为了提高效率，进行了批量处理，首先需要将图片在标注前进行重编号，以便执行循环操作，循环代码示例如下：

% 定义变量

% i---循环变量，也就是循环次数

clc;clear;

for i = 1:6 %循环6次

img\_name=['D:\MATLAB\matting\pic\','pic\_',num2str(i),'.bmp']; %读取指定路径图片

scribs\_img\_name=['D:\MATLAB\matting\picm\','pic\_',num2str(i),'\_m.bmp']; %读取指定路径图片

alpha\_name=['D:\MATLAB\matting\alpha\','alpha\_',num2str(i),'.bmp'];%定义alpha图片路径及名称

if exist(img\_name,'file') == 2

if exist(scribs\_img\_name,'file') == 2

if exist(alpha\_name,'file') == 2 %重新标注抠图，启用判断，进行删除更新操作

delete(alpha\_name);

fprintf('删除第%d张alpha图片以便重新生成\n', i);

end

runMatting

fprintf('生成第%d张alpha图片\n', i);

%fprintf('当前图片名称：%s\n', img\_name);

%fprintf('当前标注图片名称：%s\n', scribs\_img\_name);

else

fprintf('第%d张标注图不存在，请标注\n', i);

if exist(alpha\_name,'file') == 2 %重新标注抠图，启用判断，进行删除更新操作

delete(alpha\_name);

fprintf('标注图不存在,删除第%d张多余alpha图片\n', i);

end

end

else

fprintf('第%d张原图不存在\n', i);

if exist(alpha\_name,'file') == 2 %进行删除更新操作

delete(alpha\_name);

fprintf('原图不存在,删除第%d张多余alpha图片\n', i);

end

end

end

fprintf('已执行完%d张图片，操作结束', i);

对应的脚本文件：

项目的完整备份：



**3.2、提取前景图**

在python中利用OpenCV进行操作，根据原图和对应的alpha图提取前景图，代码如下：

import cv2

import numpy as np

img = cv2.imread("1.jpg")

mask = cv2.imread("2.jpg", 0) #读取灰度图像

height,width,channel=img.shape

b, g, r = cv2.split(img)

#-----------------1.获取透明的前景图像-----------

dstt = np.zeros((4,height,width),dtype=img.dtype)

dstt[0][0:height,0:width] = b

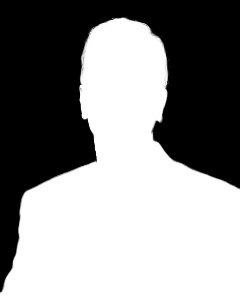
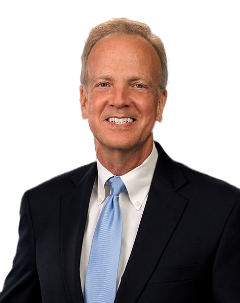
dstt[1][0:height,0:width] = g

dstt[2][0:height,0:width] = r

dstt[3][0:height,0:width] = mask

cv2.imwrite("fore.png", cv2.merge(dstt))

执行效果如下：左为原图，中间为alpha图，右为透明的前景图。

代码文件：

该段代码需要加入循环，以便批量生成无背景的透明前景图。

**3.3、切割透明目标图**

考虑到生成图的整体大小会大于目标实际大小，需要针对这类图片进行处理，切割并保存与目标大小保持一致的图片。

切割图片的代码如下：

import cv2

img = cv2.imread("test.jpg")

print(img.shape)

cropped = img[0:128, 0:512] # 裁剪坐标为[y0:y1, x0:x1]

cv2.imwrite("cv\_cut\_test.jpg", cropped)

上述代码中的四个参数，y0、y1、x0、x1通过遍历待裁剪图片的各个像素的rgb值，背景为255、255、255，非3个255的就是目标区域，取出目标实际大小的min y、max y、min x、max x四个值代入，实现代码如下：

暂缺。

**3.4、融合图片**

根据一定的规则，进行图片的融合，同时生成xml标注文件。

**3.4.1、修正大小**

根据目标放置于背景图中的位置，对上述切割的图片进行大小的调整。

需要明确知道目标的实际尺寸Ms，同时确认背景图片中作为参照的对象的实际尺寸（以铁路轨道为例，定义为Gs）。

背景图中铁轨在不同位置的大小可计算得到，如下图：



在固定角度、高度的前提下，计算出铁轨在图片中不同y坐标中的大小，定义为Gp。

在已知目标实际尺寸Ms、轨道实际尺寸Gs、在y坐标中轨道图片中尺寸Gp，可以计算出放置于该y坐标中的目标尺寸Mp，公式如下：

Mp = Ms \* （Gp / Gs）

修正图片大小的实现代码如下：

暂略。

**3.4.2、修正亮度**

暂时还未找到思路通顺的方案。

**3.4.3、融合图片**

利用python融合图片，示例如下：

import cv2

import numpy as np

img = cv2.imread("2.png") #图二代表嵌入的小图片，分辨率100\*100

co = cv2.imread("1.png", -1)#图一代表一张大的背景图，分辨率大于400\*400

scr\_channels = cv2.split(co)

dstt\_channels = cv2.split(img)

b, g, r, a = cv2.split(co)

for i in range(3):

dstt\_channels[i][100:200,300:400] = dstt\_channels[i][100:200,300:400]\*(255.0-a)/255 #图2融合进图1的位置为y坐标100到200，x坐标300到400

dstt\_channels[i][100:200,300:400]+=np.array(scr\_channels[i]\*(a/255), dtype=np.uint8)

cv2.imwrite("img\_target.png", cv2.merge(dstt\_channels))

上述代码文件：

**3.4.4、生成xml文件**

根据上述步骤融合图片后，生成对应图片的xml标注文件，xml文件的示例如下：



<annotation>

<folder>JPEGImages</folder>

<filename>02.jpg</filename>

<path>D:\labelImg\JPEGImages\02.jpg</path>

<source>

<database>Unknown</database>

</source>

<size>

<width>1920</width>

<height>1200</height>

<depth>3</depth>

</size>

<segmented>0</segmented>

<object>

<name>ship</name>

<pose>Unspecified</pose>

<truncated>0</truncated>

<difficult>0</difficult>

<bndbox>

<xmin>730</xmin>

<ymin>100</ymin>

<xmax>1261</xmax>

<ymax>886</ymax>

</bndbox>

</object>

</annotation>

如上，在执行上文中的融合等操作的过程中，会获取到该xml文件中的关键信息，按照对应关系填入即可。相关代码：暂略。