



机器之心

2395
文章

990万
总阅读

[查看TA的文章>](#)

专栏 | 手机端运行卷积神经网络实践：基于TensorFlow和OpenCV实现文档检测功能

2017-05-30 13:15

手机 / line / 程序设计

机器之心投稿

作者：腾讯 iOS 客户端高级工程师冯华

本文作者通过一个真实的产品案例，展示了在手机客户端上运行一个神经网络的关键技术点。

文章目录：

1. 前言
2. 需求是什么
3. 传统的技术方案
4. 传统技术方案的难度和局限性
5. 思考如何改善
6. 无效的神经网络算法
7. 有效的神经网络算法
8. 训练网络
9. 训练数据集 (大量合成数据 + 少量真实数据)
10. 在手机设备上运行 TensorFlow
11. HED 网络在手机上遇到的奇怪 crash
12. 裁剪 TensorFlow
13. 裁剪 HED 网络
14. TensorFlow API 的选择
15. OpenCV 算法
16. 总结
17. 参考文献

前言

•

本文不是神经网络或机器学习的入门教学，而是通过一个真实的产品案例，展示了在手机客户端上运行一个神经网络的关键技术点。

•

大家都在搜：人民币最佳代言人



热门图集



全家出道！范冰冰妹妹5岁就拿了模特冠军



怀二胎了？董璇小腹微隆身材发福



39岁刘涛穿紧身衣健身 马甲线蜜桃臀惹人羡



刚刚，这个国家被中国外交部狠狠打脸



24小时热文

- 1

大数据杀熟涉嫌价格欺诈，住宿、出行和票务是重灾区

29万 阅读
- 2

中国最先进无人机大批外售，不怕技术泄露吗？

127万 阅读
- 3

继阿里之后，中老年人又被腾讯盯上了？

110万 阅读
- 📷 行将开启雨季

最前线 | 贾跃亭回国造车疑云不断，如何一边招人一边拿地？

134万 阅读
- 🚗 UBER

神反转？！Uber自动驾驶致死事故中，行人或存在过错

31万 阅读

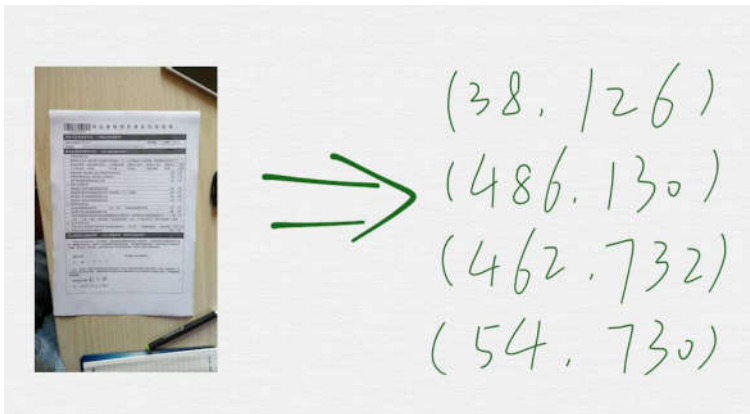
新闻 体育 汽车 房产 旅游 教育 时尚 科技 财经 娱乐 更多 登录狐友

VGG16/VGG19, Inception v1-v4 Net, ResNet 等, 这些分类网络通常义都可以作为其他算法中的基础网络结构, 尤其是 VGG 网络, 被很多其他的算法借鉴, 本文也会使用 VGG16 的基础网络结构, 但是不会对 VGG 网络做详细的入门教学

虽然本文不是神经网络技术的入门教程, 但是仍然会给出一系列的相关入门教程和技术文档的链接, 有助于进一步理解本文的内容

具体使用到的神经网络算法, 只是本文的一个组成部分, 除此之外, 本文还介绍了如何裁剪 TensorFlow 静态库以便于在手机端运行, 如何准备训练样本图片, 以及训练神经网络时的各种技巧等等

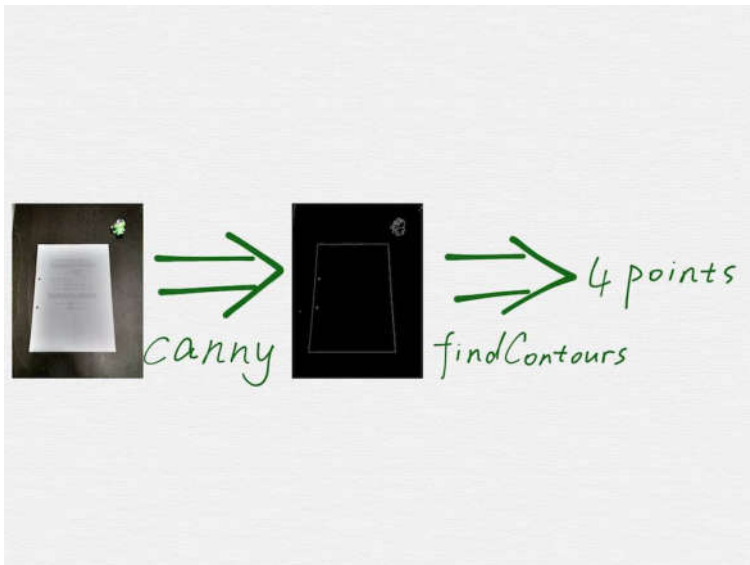
需求是什么



需求很容易描述清楚, 如上图, 就是在一张图里, 把矩形形状的文档的四个顶点的坐标找出来。

传统的技术方案

Google 搜索 opencv scan document, 是可以找到好几篇相关的教程的, 这些教程里面的技术手段, 也都大同小异, 关键步骤就是调用 OpenCV 里面的两个函数, cv2.Canny() 和 cv2.findContours()。



看上去很容易就能实现出来, 但是真实情况是, 这些教程, 仅仅是个 demo 演示而已, 用来演示的图片, 都是最理想的简单情况, 真实的场景图片会比这个复杂的多, 会有各种干扰因素, 调用 canny 函数得到的边缘检测结果, 也会比 demo 中的情况凌乱的多, 比如会检

广告

搜狐号推荐

-  搜狐科技视界
搜狐科技官方原创账号。聚焦TMT领域大事件、大趋势和新变化, 用我们的视角观...
-  驱动之家
驱动之家网站是在IT行业内居于领导地位的新闻资讯、产品评测、驱动程序下载...
-  搜狐先知道
搜狐集团官方账号
-  搜狐知道
搜狐知道是搜狐媒体推出的以视频直播、点播、在线问答方式为搜狐媒体用户构...
-  蓝鲸TMT
蓝鲸TMT网, 关注移动互联网创业报道; 关注互联网热点新闻事件。



联系我们

🏠 新闻 体育 汽车 房产 旅游 教育 时尚 科技 财经 娱乐 更多 🔍 登录狐友

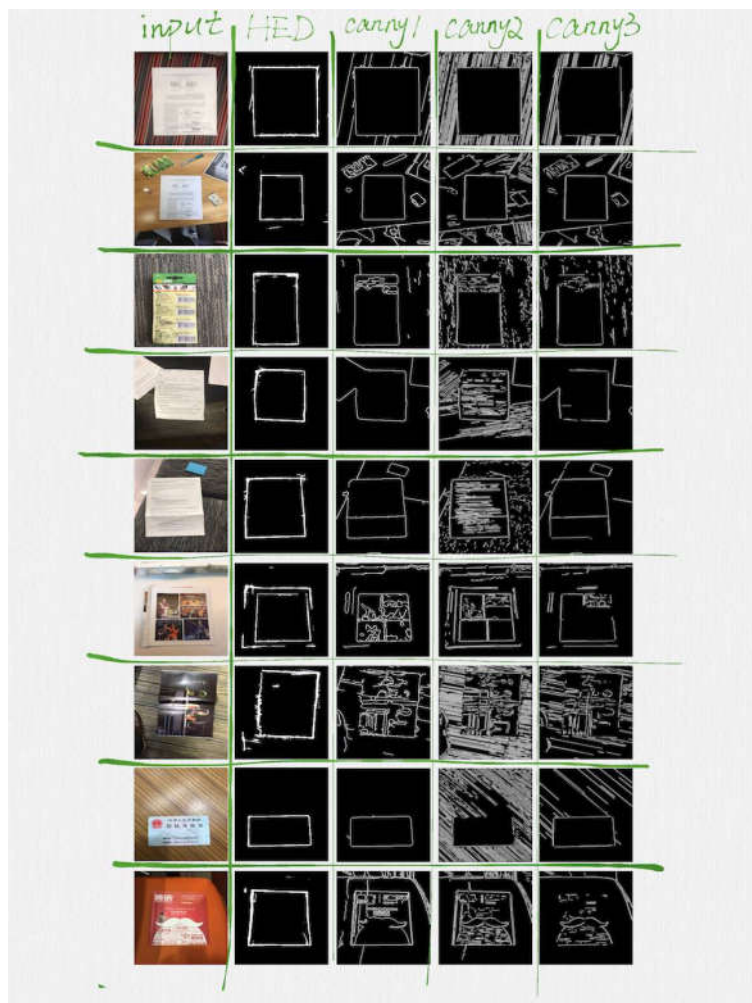
在距离不等的空隙。另外，findContours 函数也只能检测闭合的多边形的顶点，但是开不能确保这个多边形就是一个合理的矩形。因此在我们的第一版技术方案中，对这两个关键步骤，进行了大量的改进和调优，概括起来就是：

- 改进 canny 算法的效果，增加额外的步骤，得到效果更好的边缘检测图
- 针对 canny 步骤得到的边缘图，建立一套数学算法，从边缘图中寻找出一个合理的矩形区域

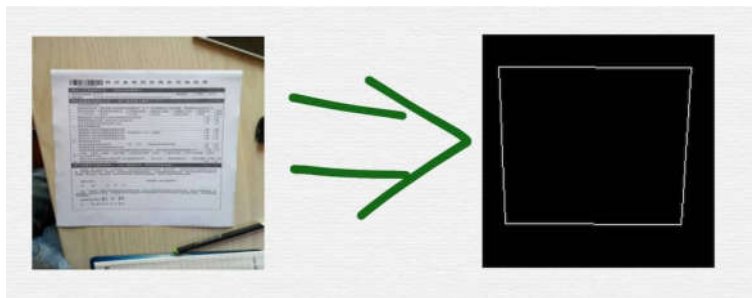
传统技术方案的难度和局限性

- canny 算法的检测效果，依赖于几个阈值参数，这些阈值参数的选择，通常都是人为设置的经验值，在改进的过程中，引入额外的步骤后，通常又会引入一些新的阈值参数，同样，也是依赖于调试结果设置的经验值。整体来看，这些阈值参数的个数，不能特别的多，因为一旦太多了，就很难依赖经验值进行设置，另外，虽然有这些阈值参数，但是最终的参数只是一组或少数几组固定的组合，所以算法的鲁棒性又会打折扣，很容易遇到边缘检测效果不理想的场景
- 在边缘图上建立的数学模型很复杂，代码实现难度大，而且也会遇到算法无能为力的场景

下面这张图表，能够很好的说明上面列出的这两个问题：



神经网络的输入和输出

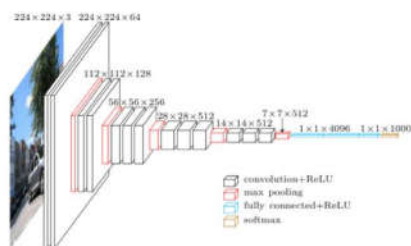


按照这种思路，对于神经网络部分，现在的需求变成了上图所示的样子。

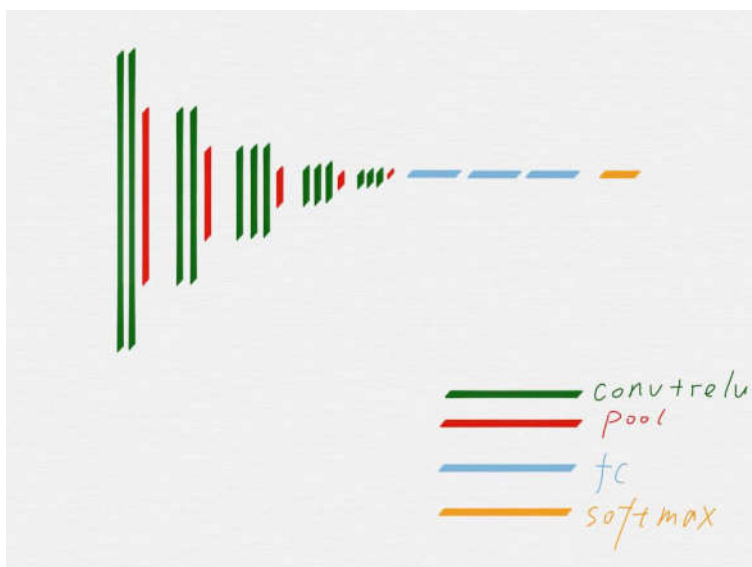
HED(Holistically-Nested Edge Detection) 网络

边缘检测这种需求，在图像处理领域里面，通常叫做 Edge Detection 或 Contour Detection，按照这个思路，找到了 Holistically-Nested Edge Detection 网络模型。

HED 网络模型是在 VGG16 网络结构的基础上设计出来的，所以有必要先看看 VGG16。



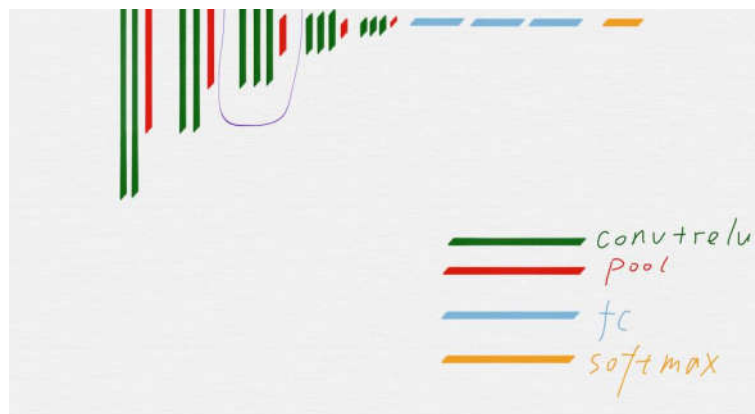
上图是 VGG16 的原理图，为了方便从 VGG16 过渡到 HED，我们先把 VGG16 变成下面这种示意图：



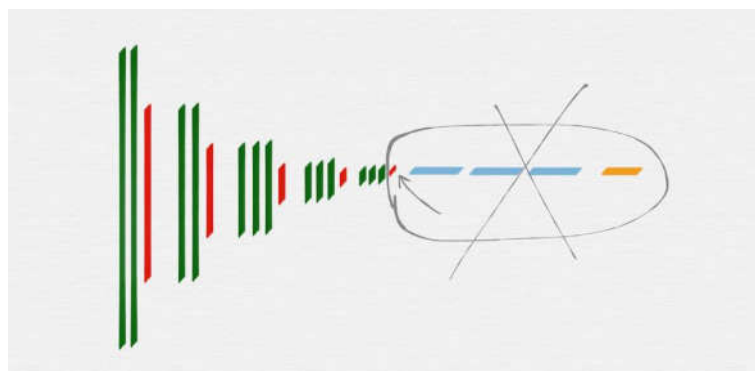
在上面这个示意图里，用不同的颜色区分了 VGG16 的不同组成部分。



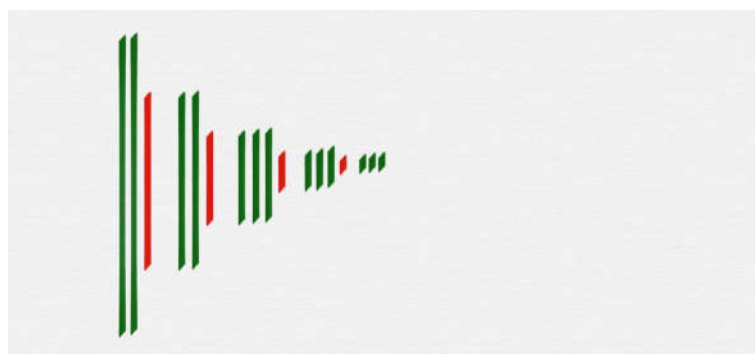
新闻 体育 汽车 房产 旅游 教育 时尚 科技 财经 娱乐 更多 登录狐友



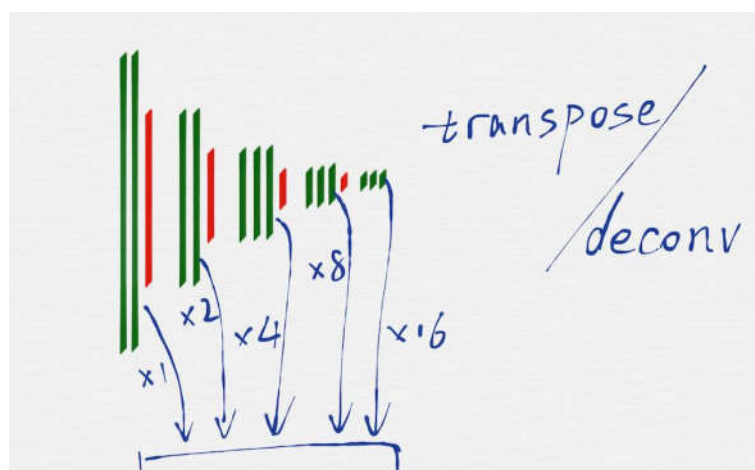
从示意图上可以看到，绿色代表的卷积层和红色代表的池化层，可以很明显的划分出五组，上图用紫色线条框出来的就是其中的第三组。



HED 网络要使用的就是 VGG16 网络里面的这五组，后面部分的 fully connected 层和 softmax 层，都是不需要的，另外，第五组的池化层 (红色) 也是不需要的。

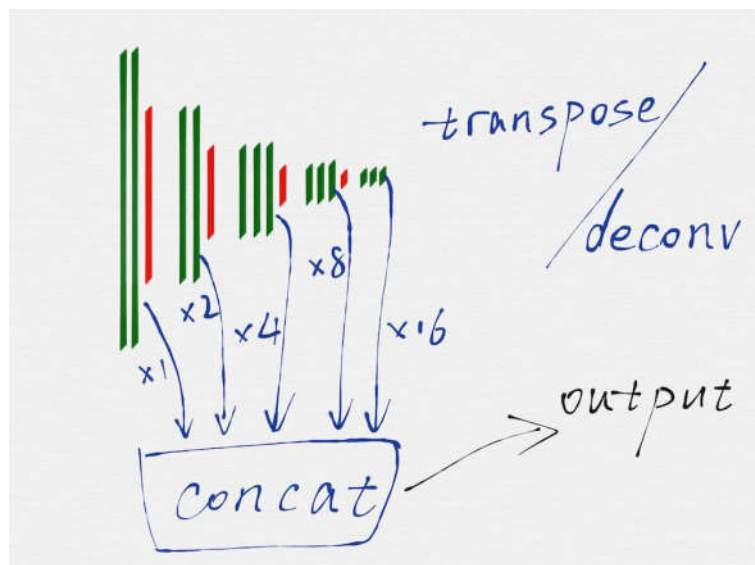


去掉不需要的部分后，就得到上图这样的网络结构，因为有池化层的作用，从第二组开始，每一组的输入 image 的长宽值，都是前一组的输入 image 的长宽值的一半。



新闻 体育 汽车 房产 旅游 教育 时尚 科技 财经 娱乐 更多 登录狐友

HED 网络是一种多尺度多融合 (multi-scale and multi-level feature learning) 的网络结构，所谓的多尺度，就是如上图所示，把 VGG16 的每一组的最后一个卷积层 (绿色部分) 的输出取出来，因为每一组得到的 image 的长宽尺寸是不一样的，所以这里还需要用转置卷积 (transposed convolution)/ 反卷积 (deconv) 对每一组得到的 image 再做一遍运算，从效果上看，相当于把第二至五组得到的 image 的长宽尺寸分别扩大 2 至 16 倍，这样在每个尺度 (VGG16 的每一组就是一个尺度) 上得到的 image，都是相同的大小了。



把每一个尺度上得到的相同大小的 image，再融合到一起，这样就得到了最终的输出 image，也就是具有边缘检测效果的 image。

基于 TensorFlow 编写的 HED 网络结构代码如下：

```
def hed_net(inputs, batch_size):

    # ref https://github.com/s9xie/hed/blob/master/examples/hed/train_val.prototxt

    with tf.variable_scope('hed', 'hed', [inputs]):

        with slim.arg_scope([slim.conv2d, slim.fully_connected],

            activation_fn=tf.nn.relu,

            weights_initializer=tf.truncated_normal_initializer(0.0, 0.01),

            weights_regularizer=slim.l2_regularizer(0.0005)):

            # vgg16 conv && max_pool layers

            net = slim.repeat(inputs, 2, slim.conv2d, 12, [3, 3], scope='conv1')

            dsn1 = net

            net = slim.max_pool2d(net, [2, 2], scope='pool1')

            net = slim.repeat(net, 2, slim.conv2d, 24, [3, 3], scope='conv2')

            dsn2 = net
```

[新闻](#)[体育](#)[汽车](#)[房产](#)[旅游](#)[教育](#)[时尚](#)[科技](#)[财经](#)[娱乐](#)[更多](#)[登录狐友](#)

```
net = slim.repeat(net, 3, slim.conv2d, 48, [3, 3], scope='conv3')

dsn3 = net

net = slim.max_pool2d(net, [2, 2], scope='pool3')

net = slim.repeat(net, 3, slim.conv2d, 96, [3, 3], scope='conv4')

dsn4 = net

net = slim.max_pool2d(net, [2, 2], scope='pool4')

net = slim.repeat(net, 3, slim.conv2d, 192, [3, 3], scope='conv5')

dsn5 = net

# net = slim.max_pool2d(net, [2, 2], scope='pool5') # no need this pool layer

# dsn layers

dsn1 = slim.conv2d(dsn1, 1, [1, 1], scope='dsn1')

# no need deconv for dsn1

dsn2 = slim.conv2d(dsn2, 1, [1, 1], scope='dsn2')

deconv_shape = tf.pack([batch_size, const.image_height, const.image_width, 1])

dsn2 = deconv_mobile_version(dsn2, 2, deconv_shape) # deconv_mobile_version can
work on mobile

dsn3 = slim.conv2d(dsn3, 1, [1, 1], scope='dsn3')

deconv_shape = tf.pack([batch_size, const.image_height, const.image_width, 1])

dsn3 = deconv_mobile_version(dsn3, 4, deconv_shape)

dsn4 = slim.conv2d(dsn4, 1, [1, 1], scope='dsn4')

deconv_shape = tf.pack([batch_size, const.image_height, const.image_width, 1])

dsn4 = deconv_mobile_version(dsn4, 8, deconv_shape)

dsn5 = slim.conv2d(dsn5, 1, [1, 1], scope='dsn5')

deconv_shape = tf.pack([batch_size, const.image_height, const.image_width, 1])

dsn5 = deconv_mobile_version(dsn5, 16, deconv_shape)

# dsn fuse

dsn_fuse = tf.concat(3, [dsn1, dsn2, dsn3, dsn4, dsn5])

dsn_fuse = tf.reshape(dsn_fuse, [batch_size, const.image_height, const.image_width,
5]) #without this, will get error: ValueError: Number of in_channels must be known.

dsn_fuse = slim.conv2d(dsn_fuse, 1, [1, 1], scope='dsn_fuse')
```


🏠 新闻 体育 汽车 房产 旅游 教育 时尚 科技 财经 娱乐 更多 🔍 登录狐友

训练网络

cost 函数

论文给出的 HED 网络是一个通用的边缘检测网络，按照论文的描述，每一个尺度上得到的 image，都需要参与 cost 的计算，这部分的代码如下：

```
input_queue_for_train = tf.train.string_input_producer([FLAGS.csv_path])

image_tensor, annotation_tensor = input_image_pipeline(dataset_root_dir_string,
input_queue_for_train, FLAGS.batch_size)

dsn_fuse, dsn1, dsn2, dsn3, dsn4, dsn5 = hed_net(image_tensor, FLAGS.batch_size)

cost = class_balanced_sigmoid_cross_entropy(dsn_fuse, annotation_tensor) +

class_balanced_sigmoid_cross_entropy(dsn1, annotation_tensor) +

class_balanced_sigmoid_cross_entropy(dsn2, annotation_tensor) +

class_balanced_sigmoid_cross_entropy(dsn3, annotation_tensor) +

class_balanced_sigmoid_cross_entropy(dsn4, annotation_tensor) +

class_balanced_sigmoid_cross_entropy(dsn5, annotation_tensor)
```

按照这种方式训练出来的网络，检测到的边缘线是有一点粗的，为了得到更细的边缘线，通过多次试验找到了一种优化方案，代码如下：

```
input_queue_for_train = tf.train.string_input_producer([FLAGS.csv_path])

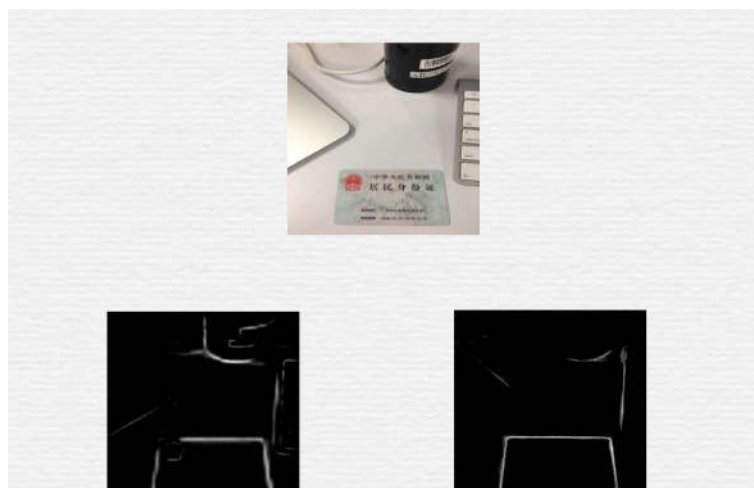
image_tensor, annotation_tensor = input_image_pipeline(dataset_root_dir_string,
input_queue_for_train, FLAGS.batch_size)

dsn_fuse, _, _, _, _ = hed_net(image_tensor, FLAGS.batch_size)

cost = class_balanced_sigmoid_cross_entropy(dsn_fuse, annotation_tensor)
```

也就是不再让每个尺度上得到的 image 都参与 cost 的计算，只使用融合后得到的最终 image 来进行计算。

两种 cost 函数的效果对比如下图所示，右侧是优化过后的效果：



[新闻](#)[体育](#)[汽车](#)[房产](#)[旅游](#)[教育](#)[时尚](#)[科技](#)[财经](#)[娱乐](#)[更多](#)[登录狐友](#)

另外还有一点，按照 HED 论文里的要求，计算 cost 的时候，不能使用常见的方差 cost，而应该使用 cost-sensitive loss function，代码如下：

```
def class_balanced_sigmoid_cross_entropy(logits, label, name='cross_entropy_loss'):

    """

    The class-balanced cross entropy loss,

    as in `Holistically-Nested Edge Detection

    <http://arxiv.org/abs/1504.06375>`_.

    This is more numerically stable than class_balanced_cross_entropy

    :param logits: size: the logits.

    :param label: size: the ground truth in {0,1}, of the same shape as logits.

    :returns: a scalar. class-balanced cross entropy loss

    """

    y = tf.cast(label, tf.float32)

    count_neg = tf.reduce_sum(1. - y) # the number of 0 in y

    count_pos = tf.reduce_sum(y) # the number of 1 in y (less than count_neg)

    beta = count_neg / (count_neg + count_pos)

    pos_weight = beta / (1 - beta)

    cost = tf.nn.weighted_cross_entropy_with_logits(logits, y, pos_weight)

    cost = tf.reduce_mean(cost * (1 - beta), name=name)

    return cost
```

转置卷积层的双线性初始化

在尝试 FCN 网络的时候，就被这个问题卡住过很长一段时间，按照 FCN 的要求，在使用转置卷积 (transposed convolution)/ 反卷积 (deconv) 的时候，要把卷积核的值初始化成双线性放大矩阵 (bilinear upsampling kernel)，而不是常用的正态分布随机初始化，同时还要使用很小的学习率，这样才更容易让模型收敛。

HED 的论文中，并没有明确的要求也要采用这种方式初始化转置卷积层，但是，在训练过程中发现，采用这种方式进行初始化，模型才更容易收敛。

这部分的代码如下：

```
def get_kernel_size(factor):

    """

    Find the kernel size given the desired factor of upsampling.
```

[新闻](#)[体育](#)[汽车](#)[房产](#)[旅游](#)[教育](#)[时尚](#)[科技](#)[财经](#)[娱乐](#)[更多](#)[登录狐友](#)

```
return 2 * factor - factor % 2

def upsample_filt(size):
    """
    Make a 2D bilinear kernel suitable for upsampling of the given (h, w) size.
    """
    factor = (size + 1) // 2
    if size % 2 == 1:
        center = factor - 1
    else:
        center = factor - 0.5
    og = np.ogrid[:size, :size]
    return (1 - abs(og[0] - center) / factor) * (1 - abs(og[1] - center) / factor)

def bilinear_upsample_weights(factor, number_of_classes):
    """
    Create weights matrix for transposed convolution with bilinear filter
    initialization.
    """
    filter_size = get_kernel_size(factor)
    weights = np.zeros((filter_size,
                        filter_size,
                        number_of_classes,
                        number_of_classes), dtype=np.float32)
    upsample_kernel = upsample_filt(filter_size)
    for i in xrange(number_of_classes):
        weights[:, :, i, i] = upsample_kernel
    return weights
```

训练过程冷启动

HED 网络不像 VGG 网络那样很容易就进入收敛状态，也不大容易进入期望的理想状态，主要是两方面的原因：

前面提到的转置卷积层的双线性初始化，就是一个重要因素，因为在 4 个尺度上，都需要反卷积，如果反卷积层不能收敛，那整个 HED 都不会进入期望的理想状态

另外一个原因，是由 HED 的多尺度引起的，既然是多尺度了，那每个尺度上得到的 image 都应该对模型的最终输出 image 产生贡献，在训练的过程中发现，如果输入 image 的尺寸是 224*224，还是很容易就训练成功的，但是当把输入 image 的尺寸调整为 256*256 后，很容易出现一种状况，就是 5 个尺度上得到的 image，会有 1~2 个 image 是无效的 (全部是黑色)

为了解决这里遇到的问题，采用的办法就是先使用少量样本图片 (比如 2000 张) 训练网络，在很短的训练时间 (比如迭代 1000 次) 内，如果 HED 网络不能表现出收敛的趋势，或者不能达到 5 个尺度的 image 全部有效的状态，那就直接放弃这轮的训练结果，重新开启下一轮训练，直到满意为止，然后才使用完整的训练样本集合继续训练网络。

训练数据集 (大量合成数据 + 少量真实数据)

HED 论文里使用的训练数据集，是针对通用的边缘检测目的的，什么形状的边缘都有，比如下面这种：



用这份数据训练出来的模型，在做文档扫描的时候，检测出来的边缘效果并不理想，而且这份训练数据集的样本数量也很小，只有一百多张图片 (因为这种图片的人工标注成本太高了)，这也会影响模型的质量。

现在的需求里，要检测的是具有一定透视和旋转变换效果的矩形区域，所以可以大胆的猜测，如果准备一批针对性更强的训练样本，应该是可以得到更好的边缘检测效果的。

借助第一版技术方案收集回来的真实场景图片，我们开发了一套简单的标注工具，人工标注了 1200 张图片 (标注这 1200 张图片的时间成本也很高)，但是这 1200 多张图片仍然有很多问题，比如对于神经网络来说，1200 个训练样本其实还是不够的，另外，这些图片覆盖的场景其实也比较少，有些图片的相似度比较高，这样的数据放到神经网络里训练，泛化的效果并不好。

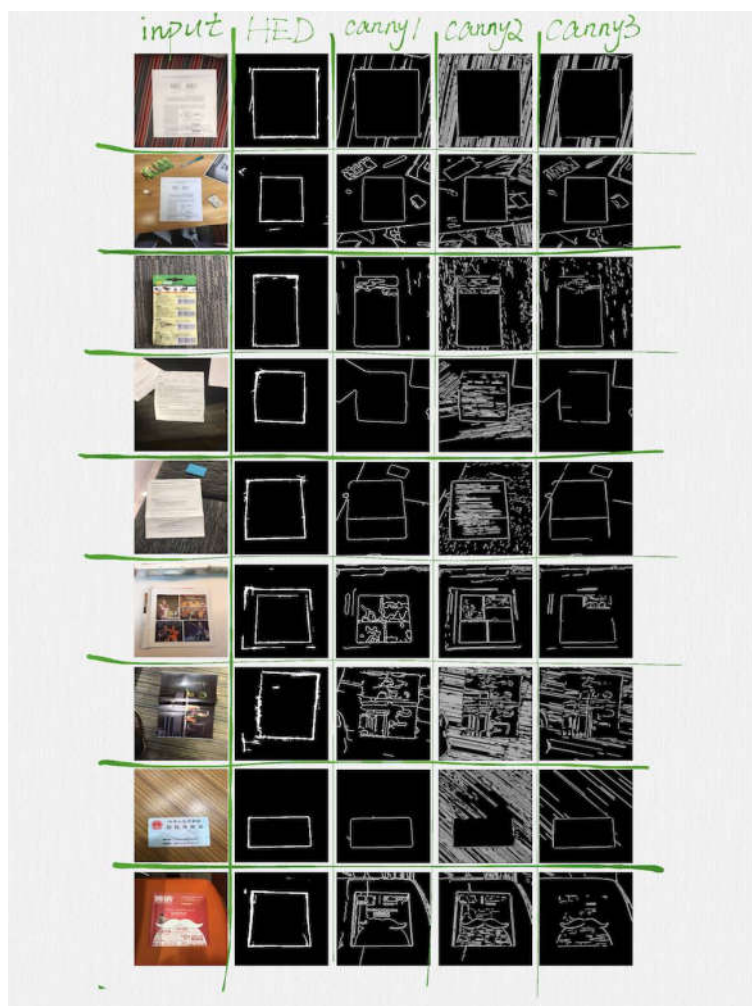
所以，还采用技术手段，合成了 80000 多张训练样本图片。



如上图所示，一张背景图和一张前景图，可以合成出一对训练样本数据。在合成图片的过程

- 在前景图上添加旋转、平移、透视变换
- 对背景图进行了随机的裁剪
- 通过试验对比，生成合适宽度的边缘线
- OpenCV 不支持透明图层之间的旋转和透视变换操作，只能使用最低精度的插值算法，为了改善这一点，后续改成了使用 iOS 模拟器，通过 CALayer 上的操作来合成图片
- 在不断改进训练样本的过程中，还根据真实样本图片的统计情况和各种途径的反馈信息，刻意模拟了一些更复杂的样本场景，比如凌乱的背景环境、直线边缘干扰等等

经过不断的调整和优化，最终才训练出一个满意的模型，可以再次通过下面这张图表中的第二列看一下神经网络模型的边缘检测效果：



在手机上运行 TensorFlow

在手机上使用 TensorFlow 库

TensorFlow 官方是支持 iOS 和 Android 的，而且有清晰的文档，照着做就行。但是因为

种，就是我们在两个不同的 iOS APP 中遇到的问题和解决办法，可以作为参考：

•

A 产品使用的是 protobuf 2，同时由于各种历史原因，使用并且停留在了很旧的某个版本的 Base 库上，而 protobuf 3 的内部也使用了 Base 库，当 A 产品升级到 protobuf 3 后，protobuf 3 的 Base 库和 A 源码中的 Base 库产生了一些奇怪的冲突，最后的解决办法是手动修改了 A 源码中的 Base 库，避免编译时的冲突

•

B 产品也是使用的 protobuf 2，而且 B 产品使用到的多个第三方模块 (没有源码，只有二进制文件) 也是依赖于 protobuf 2，直接升级 B 产品使用的 protobuf 库就行不通了，最后采用的方法是修改 TensorFlow 和 TensorFlow 中使用的 protobuf 3 的源代码，把 protobuf 3 换了一个命名空间，这样两个不同版本的 protobuf 库就可以共存了

Android 上因为本身是可以使用动态库的，所以即便 app 必须使用 protobuf 2 也没有关系，不同的模块使用 dlopen 的方式加载各自需要的特定版本的库就可以了。

在手机上使用训练得到的模型文件

模型通常都是在 PC 端训练的，对于大部分使用者，都是用 Python 编写的代码，得到 ckpt 格式的模型文件。在使用模型文件的时候，一种做法就是用代码重新构建出完整的神经网络，然后加载这个 ckpt 格式的模型文件，如果是在 PC 上使用模型文件，用这个方法其实也是可以接受的，复制粘贴一下 Python 代码就可以重新构建整个神经网络。但是，在手机上只能使用 TensorFlow 提供的 C++ 接口，如果还是用同样的思路，就需要用 C++ API 重新构建一遍神经网络，这个工作量就有点大了，而且 C++ API 使用起来比 Python API 复杂的多，所以，在 PC 上训练完网络后，还需要把 ckpt 格式的模型文件转换成 pb 格式的模型文件，这个 pb 格式的模型文件，是用 protobuf 序列化得到的二进制文件，里面包含了神经网络的具体结构以及每个矩阵的数值，使用这个 pb 文件的时候，不需要再用代码构建完整的神经网络结构，只需要反序列化一下就可以了，这样的话，用 C++ API 编写的代码就会简单很多，其实这也是 TensorFlow 推荐的使用方法，在 PC 上使用模型的时候，也应该使用这种 pb 文件 (训练过程中使用 ckpt 文件)。

HED 网络在手机上遇到的奇怪 crash

在手机上加载 pb 模型文件并且运行的时候，遇到过一个诡异的错误，内容如下：

```
Invalid argument: No OpKernel was registered to support Op 'Mul' with these attrs.
Registered devices: [CPU], Registered kernels:

device='CPU'; T in [DT_FLOAT]

[[Node: hed/mul_1 = Mul[T=DT_INT32](hed/strided_slice_2, hed/mul_1/y)]]
```

之所以诡异，是因为从字面上看，这个错误的含义是缺少乘法操作 (Mul)，但是我用其他的神经网络模型做过对比，乘法操作模块是可以正常工作的。

Google 搜索后发现很多人遇到过类似的情况，但是错误信息又并不相同，后来在 TensorFlow 的 github issues 里终于找到了线索，综合起来解释，是因为 TensorFlow 是基于操作 (Operation) 来模块化设计和编码的，每一个数学计算模块就是一个 Operation，由于各种原因，比如内存占用大小、GPU 独占操作等等，mobile 版的 TensorFlow，并没有包含所有的 Operation，mobile 版的 TensorFlow 支持的 Operation 只是 PC 完整版 TensorFlow 的一个子集，我遇到的这个错误，就是因为使用到的某个 Operation 并不支持 mobile 版。

按照这个线索，在 Python 代码中逐个排查，后来定位到了出问题的代码，修改前后的代码如下：

[新闻](#)[体育](#)[汽车](#)[房产](#)[旅游](#)[教育](#)[时尚](#)[科技](#)[财经](#)[娱乐](#)[更多](#)[登录狐友](#)

```
input_shape = tf.shape(inputs)

# Calculate the ouput size of the upsampled tensor

upsampled_shape = tf.pack([input_shape[0],

input_shape[1] * upsample_factor,

input_shape[2] * upsample_factor,

1])

upsample_filter_np = bilinear_upsample_weights(upsample_factor, 1)

upsample_filter_tensor = tf.constant(upsample_filter_np)

# Perform the upsampling

upsampled_inputs = tf.nn.conv2d_transpose(inputs, upsample_filter_tensor,

output_shape=upsampled_shape,

strides=[1, upsample_factor, upsample_factor, 1])

return upsampled_inputs

def deconv_mobile_version(inputs, upsample_factor, upsampled_shape):

upsample_filter_np = bilinear_upsample_weights(upsample_factor, 1)

upsample_filter_tensor = tf.constant(upsample_filter_np)

# Perform the upsampling

upsampled_inputs = tf.nn.conv2d_transpose(inputs, upsample_filter_tensor,

output_shape=upsampled_shape,

strides=[1, upsample_factor, upsample_factor, 1])

return upsampled_inputs
```

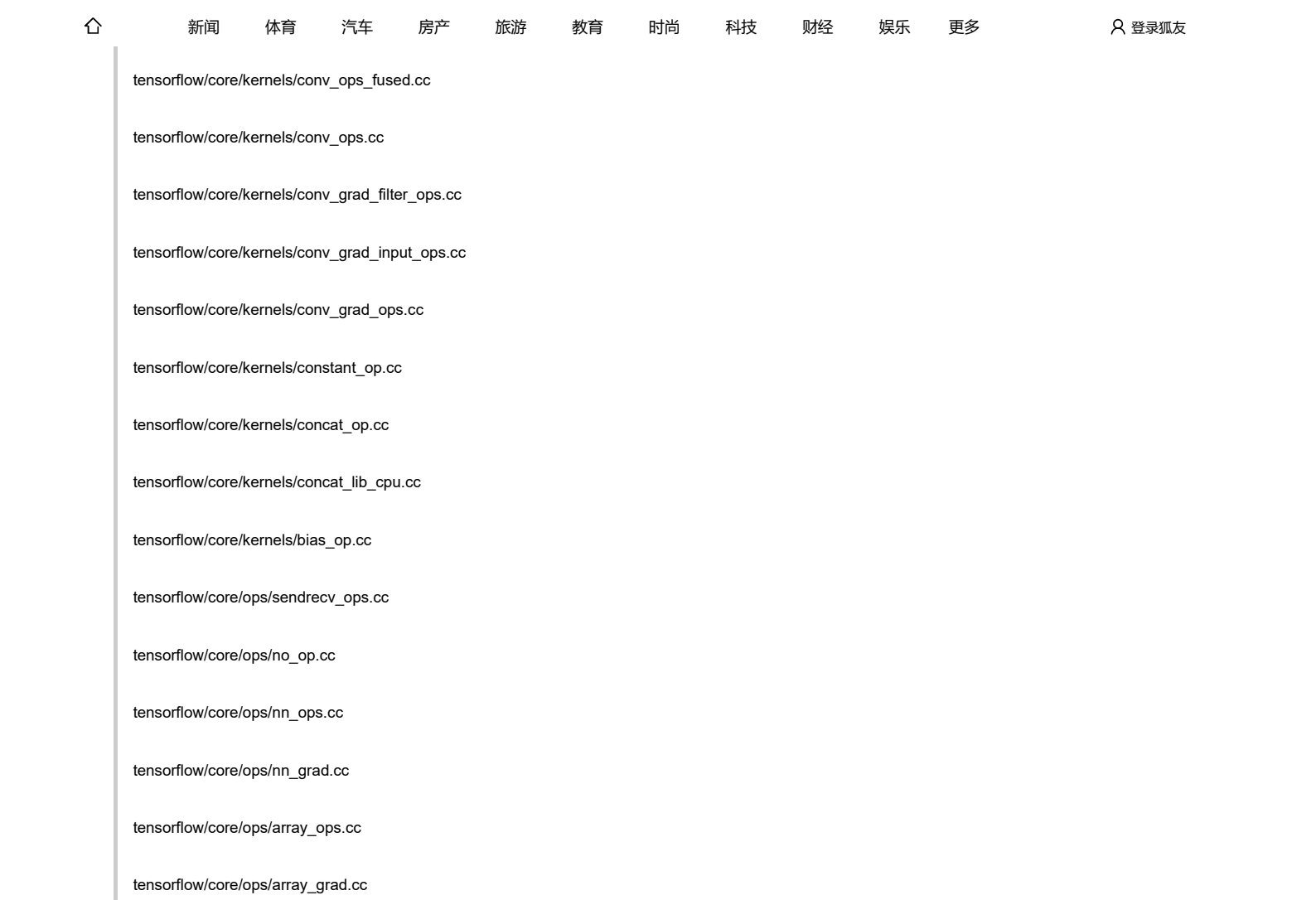
问题就是由 deconv 函数中的 tf.shape 和 tf.pack 这两个操作引起的，在 PC 版代码中，为了简洁，是基于这两个操作，自动计算出 upsampled_shape，修改过后，则是要求调用者用 hard coding 的方式设置对应的 upsampled_shape。

裁剪 TensorFlow

TensorFlow 是一个很庞大的框架，对于手机来说，它占用的体积是比较大的，所以需要尽量地缩减 TensorFlow 库占用的体积。

其实在解决前面遇到的那个 crash 问题的时候，已经指明了一种裁剪的思路，既然 mobile 版的 TensorFlow 本来就是 PC 版的一个子集，那就意味着可以根据具体的需求，让这个子集变得更小，这也就达到了裁剪的目的。具体来说，就是修改 TensorFlow 源码中的 tensorflow/tensorflow/contrib/makefile/tf_op_files.txt 文件，只保留使用到了的模块。针对 HED 网络，原有的 200 多个模块裁剪到只剩 46 个，裁剪过后的 tf_op_files.txt 文件如下：

	新闻	体育	汽车	房产	旅游	教育	时尚	科技	财经	娱乐	更多	登录狐友
tensorflow/core/kernels/where_op.cc												
tensorflow/core/kernels/unpack_op.cc												
tensorflow/core/kernels/transpose_op.cc												
tensorflow/core/kernels/transpose_functor_cpu.cc												
tensorflow/core/kernels/tensor_array_ops.cc												
tensorflow/core/kernels/tensor_array.cc												
tensorflow/core/kernels/split_op.cc												
tensorflow/core/kernels/split_v_op.cc												
tensorflow/core/kernels/split_lib_cpu.cc												
tensorflow/core/kernels/shape_ops.cc												
tensorflow/core/kernels/session_ops.cc												
tensorflow/core/kernels/sendrecv_ops.cc												
tensorflow/core/kernels/reverse_op.cc												
tensorflow/core/kernels/reshape_op.cc												
tensorflow/core/kernels/relu_op.cc												
tensorflow/core/kernels/pooling_ops_common.cc												
tensorflow/core/kernels/pack_op.cc												
tensorflow/core/kernels/ops_util.cc												
tensorflow/core/kernels/no_op.cc												
tensorflow/core/kernels/maxpooling_op.cc												
tensorflow/core/kernels/matmul_op.cc												
tensorflow/core/kernels/immutable_constant_op.cc												
tensorflow/core/kernels/identity_op.cc												
tensorflow/core/kernels/gather_op.cc												
tensorflow/core/kernels/gather_functor.cc												
tensorflow/core/kernels/fill_functor.cc												
tensorflow/core/kernels/dense_update_ops.cc												
tensorflow/core/kernels/deep_conv2d.cc												
tensorflow/core/kernels/xsmm_conv2d.cc												



需要强调的一点是，这种操作思路，是针对不同的神经网络结构有不同的裁剪方式，原则就是用到什么模块就保留什么模块。当然，因为有些模块之间还存在隐含的依赖关系，所以裁剪的时候也是要反复尝试多次才能成功的。

除此之外，还有下面这些通用手段也可以实现裁剪的目的：

- 编译器级别的 strip 操作，在链接的时候会自动的把没有调用到的函数去掉 (集成开发环境里通常已经自动将这些参数设置成了最佳组合)
- 借助一些高级技巧和工具，对二进制文件进行瘦身

借助所有这些裁剪手段，最终我们的 ipa 安装包的大小只增加了 3M。如果不做手动裁剪这一步，那 ipa 的增量，则是 30M 左右。

裁剪 HED 网络

按照 HED 论文给出的参考信息，得到的模型文件的大小是 56M，对于手机来说也是比较大的，而且模型越大也意味着计算量越大，所以需要考虑能否把 HED 网络也裁剪一下。

HED 网络是用 VGG16 作为基础网络结构，而 VGG 又是一个得到广泛验证的基础网络结构，因此修改 HED 的整体结构肯定不是一个明智的选择，至少不是首选的方案。

考虑到现在的需求，只是检测矩形区域的边缘，而并不是检测通用场景下的广义的边缘，可以认为前者的复杂度比后者更低，所以一种可行的思路，就是保留 HED 的整体结构，修改



新闻

体育

汽车

房产

旅游

教育

时尚

科技

财经

娱乐

更多

登录狐友

按照这种思路，经过多次调整和尝试，最终得到了一组合适的卷积核的数量参数，对应的模型文件只有 4.2M，在 iPhone 7P 上，处理每帧图片的时间消耗是 0.1 秒左右，满足实时性的要求。

神经网络的裁剪，目前在学术界也是一个很热门的领域，有好几种不同的理论来实现不同目的的裁剪，但是，也并不是说每一种网络结构都有裁剪的空间，通常来说，应该结合实际情况，使用合适的技术手段，选择一个合适大小的模型文件。

TensorFlow API 的选择

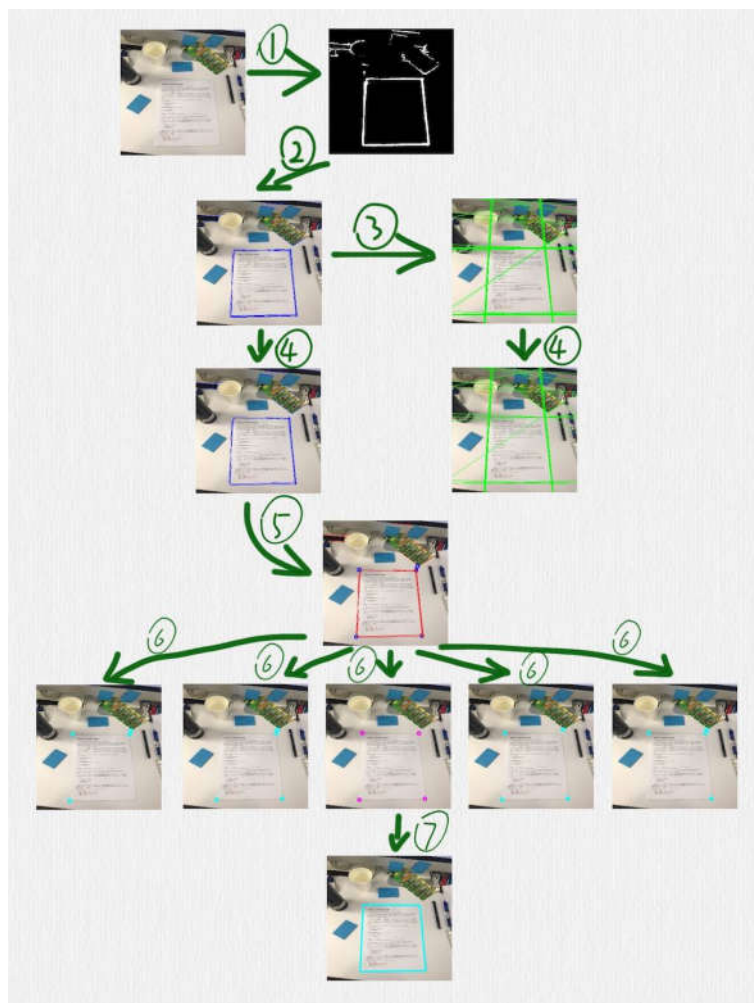
TensorFlow 的 API 是很灵活的，也比较底层，在学习过程中发现，每个人写出来的代码，风格差异很大，而且很多工程师又采用了各种各样的技巧来简化代码，但是这其实反而在无形中又增加了代码的阅读难度，也不利于代码的复用。

第三方社区和 TensorFlow 官方，都意识到了这个问题，所以更好的做法是，使用封装度更高但又保持灵活性的 API 来进行开发。本文中的代码，就是使用 TensorFlow-Slim 编写的。

OpenCV 算法

虽然用神经网络技术，已经得到了一个比 canny 算法更好的边缘检测效果，但是，神经网络也并不是万能的，干扰是仍然存在的，所以，第二个步骤中的数学模型算法，仍然是需要的，只不过因为第一个步骤中的边缘检测有了大幅度改善，所以第二个步骤中的算法，得到了适当的简化，而且算法整体的适应性也更强了。

这部分的算法如下图所示：



- 对于监督学习，数据的标注成本很高，这一步很容易出现瓶颈
- 论文、参考代码和自己的代码，这三者之间不完全一致也是正常现象
- 对于某些需求，可以在模型的准确度、大小和运行速度之间找一个平衡点

工程角度

- end-to-end 网络无效的时候，可以用 pipeline 的思路考虑问题、拆分业务，针对性的使用神经网络技术
- 至少要熟练掌握一种神经网络的开发框架，而且要追求代码的工程质量
- 要掌握神经网络技术中的一些基本套路，举一反三
- 要在学术界和工业界中间找平衡点，尽可能多的学习一些不同问题领域的神经网络模型，作为技术储备

参考文献

因不能添加外链，请点击以下原文发布地址查看。

原文地址：<http://fengjian0106.github.io/2017/05/08/Document-Scanning-With-TensorFlow-And-OpenCV/>

更多有关GMIS 2017大会的内容，请点击「阅读原文」查看机器之心官网 GMIS 专题↓↓↓



[返回搜狐，查看更多](#)


声明：本文由入驻搜狐的作者撰写，除搜狐官方账号外，观点仅代表作者本人，不代表搜狐立场。

阅读 (2964) 不感兴趣 投诉

本文相关推荐

澎湃 澎湃新闻 · 今天 08:42

用诺基亚手机边打电话边充电，印度女子被炸死



观察者网 · 昨天 21:38 26

联想发布首款区块链手机！高科技？价格其实很感人



每日经济新闻 · 今天 07:32 5

美团上海21日起开打价格战 曾被交委约谈不得低价竞争

观察者网 · 昨天 23:58 1

3个股市利好消息，这1板块重点关注，或可大涨

广告 · 今天 10:44

最前线 | 微博回话抖音：与腾讯、快手、美拍均有合作，“只是没有你”

36氪 · 昨天 22:38

独家：淘宝下架区块链白皮书代写

Bianews · 昨天 23:48

Facebook这次非死不可？巨量信息被窃引爆信任危机

澎湃 澎湃新闻 · 昨天 21:47 2

相同商品不同价格？身处网络不止裸奔，还待宰



互联网早读课 · 今天 08:00

别浪费大把时间研究K线，一招教你踩准买卖点

广告 · 今天 10:44

【早报】乐视网重启招聘；扎克伯格遭英国议员传唤



虎嗅APP · 今天 07:20 1

最前线 | 网易严选对手来了！苏宁极物开店在即，计划3年扩张至300家



36氪 · 昨天 12:33

当当网卖身：心疼背后更多是可怜之人必有可悲之处？

2018/3/21 星期三 10:45