用户手册

**1 引言**

**1.1 编写目的**

欢迎使用纺织布匹瑕疵检测系统！为了帮助用户更好地了解和使用该系统平台，特编写本手册。在本手册中，我们对本系统的功能进行了详细讲解，包括各模块的详细操作步骤和配置方式。使用户能够快速了解本系统的所有功能及使用方法。

**1.2 背景**

纺织布匹瑕疵检测系统主要按照纺织企业对瑕疵识别的总体要求，紧密结合现有的瑕疵类型，着眼于精确识别最常见的瑕疵，实现瑕疵检测的全面化、可靠化。

**2 系统概述**

**2.1 目标**

现代化的工厂，在生产管理和产品检验方面应该更多地依赖计算机而不是人工。纺织企业是劳动密集型企业，在这方面远远达不到现代化标准的水平。所以开发这套布匹瑕疵检测系统，就是为了提高纺织企业的生产力，降低企业的成本，让企业进入互联网+时代。

**2.2 功能**

**系统主要包括如下主要功能：**

（1）图片的导入。工作人员可以通过该菜单将本地的图片导入到后端的服务器中。

（2）图片的识别。根据输入的图片得到识别后的图片，新的图片上标注了瑕疵的信息，是瑕疵识别系统最基本的目标。图片的识别可以是简单的分类，也可以是复杂的定位。

（3）图片的存储。图片进入系统之后，经过加工处理，形成新的识别之后的图片，这些图片会自动存放在服务器上的特定文件夹下，如果没有人工干预，存储时间是永久。

（4）图片的展示。为用户提供瑕疵的全面信息是瑕疵识别系统的职责，为了便于理解，输出的形式也是图片，瑕疵的信息全部都展示在图片上，包括位置和类别等信息。

**2.3 性能**

a.数据精确度。由于采用的训练算法是R-FCN，属于two stage网络，在识别的精度上要超过大部分的one stage网络。而我们在检测了几百张图片之后发现只有一两张图片出现了误检或漏检。

b.时间特性。由于two stage网络需要先生成候选框，所以在时间方面消耗要高于one stage网络。在比较理想的情况下，一秒大概可以识别4-5张图片。

c.灵活性。当需要检测的类型发生变化时，只需要将新的数据集放入到后端算法中进行训练，不需要修改超参数，通用性很强。

**3 运行环境**

­**3.1 硬件**

Intel Xeon CPU E5-2620

NVIDIA TITAN XP 12GB

**3.2 软件**

ubuntu 16.04

BVLC/caffe

cuda 9.0

cudnn 7.4

python 2.7

opencv 2.4.1

Django 1.8

各软件的版本可以根据配置灵活调整，但要注意软件的依赖一致以及版本对应，例如cuda和cudnn的一致性，cuda和GPU型号的一致性，在NVIDIA的官网，可以查询到cuda和GPU相关资料；利用B/S架构做web应用时，Django1.8对应的python版本为2.7，3.2，3.3，3.4，3.5，而不支持3.6，若用python3.6需要使用Django 1.11.x或更高版本。

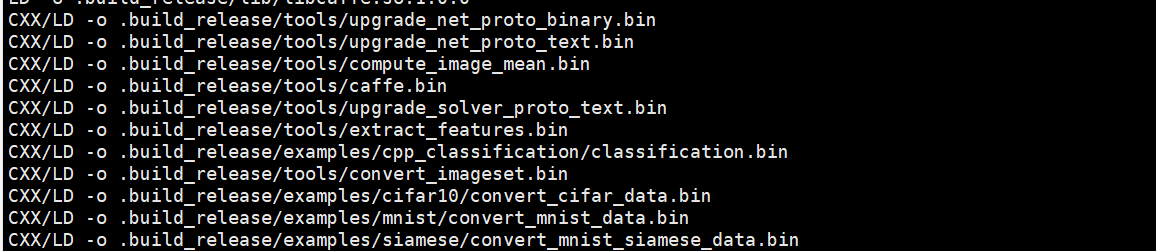
**4 使用说明**

**4.1 部署**

按照运行环境中的软件说明，进行项目的部署。ubuntu 16.04里自带python2.7，python环境不需要另外配置。cuda和cudnn到NVIDIA的官网上下载，需要注意的是安装这两个需要有管理员权限。cuda不是9.0的版本也无所谓，只需要cudnn的版本和cuda对应即可。opencv和Django都可以在ubuntu系统中使用pip指令安装。

BVLC/caffe是caffe的官方版本，在安装的时候需要先到github中下载下来，使用源代码进行安装。在安装之前还需要使用pip指令安装caffe的相关依赖包，比如yaml,protobuf等等。相关依赖的名字都可以在百度上找到，这里就不一一说明，安装指令都是一样的。

在下载完BVLC/caffe之后，解压，进入路径，复制Makefile.config.example并重命名为Makefile.config，然后打开Makefile.config，修改指令。将USE\_CUDNN、WITH\_PYTHON\_LAYER取消注释，在INCLUDE\_DIRS := $(PYTHON\_INCLUDE) /usr/local/include后面添加/usr/include/hdf5/serial，在LIBRARY\_DIRS := $(PYTHON\_LIB) /usr/local/lib /usr/lib后添加/usr/lib/x86\_64-linux-gnu /usr/lib/x86\_64-linux-gnu/hdf5/serial。添加这两个是因为ubuntu 16.04版的相关文件的位置发生了变化，如果不添加编译会出现问题。文件修改完成之后就开始编译。在Makefile.config的同目录下输入指令make -j8，j8的含义是CPU的内核为8。如果出现如下的图片，证明编译完成：



然后输入指令make pycaffe，目的是将caffe添加到python中。最后修改环境变量，输入指令vim ~/.bashrc，在最后添加export PYTHONPATH="/your/caffe/python:$PYTHONPATH"，退出来之后输入指令source ~/.bashrc，使得环境变量的修改生效。然后输入python2，在python2中import caffe，如果不报错，说明caffe环境安装完成。

环境安装完成之后，进入到mysite2路径下，运行python2 manage.py runserver，这就开启了服务。此时在浏览器中输入指定的IP地址与端口号，就可以看到前端界面。比如我们的服务器IP地址是192.10.84.31，在mysite2路径下运行python2 manage.py runserver 192.10.84.31:5000，在浏览器中输入192.10.84.31:5000，就可以看到检测界面。

**4.2 输入**

输入的必须要是图片，不管是.jpg还是.png，不能输入诸如文本、表格等，否则后台将会出错，无法识别。

**4.3 输出**

输出的也是图片，并且输入名与输出名完全相同，只是存储的位置不同而已。而且在所有图片输出到一个文件夹后，所有的输入文件将自动删除。

**5 运行说明**

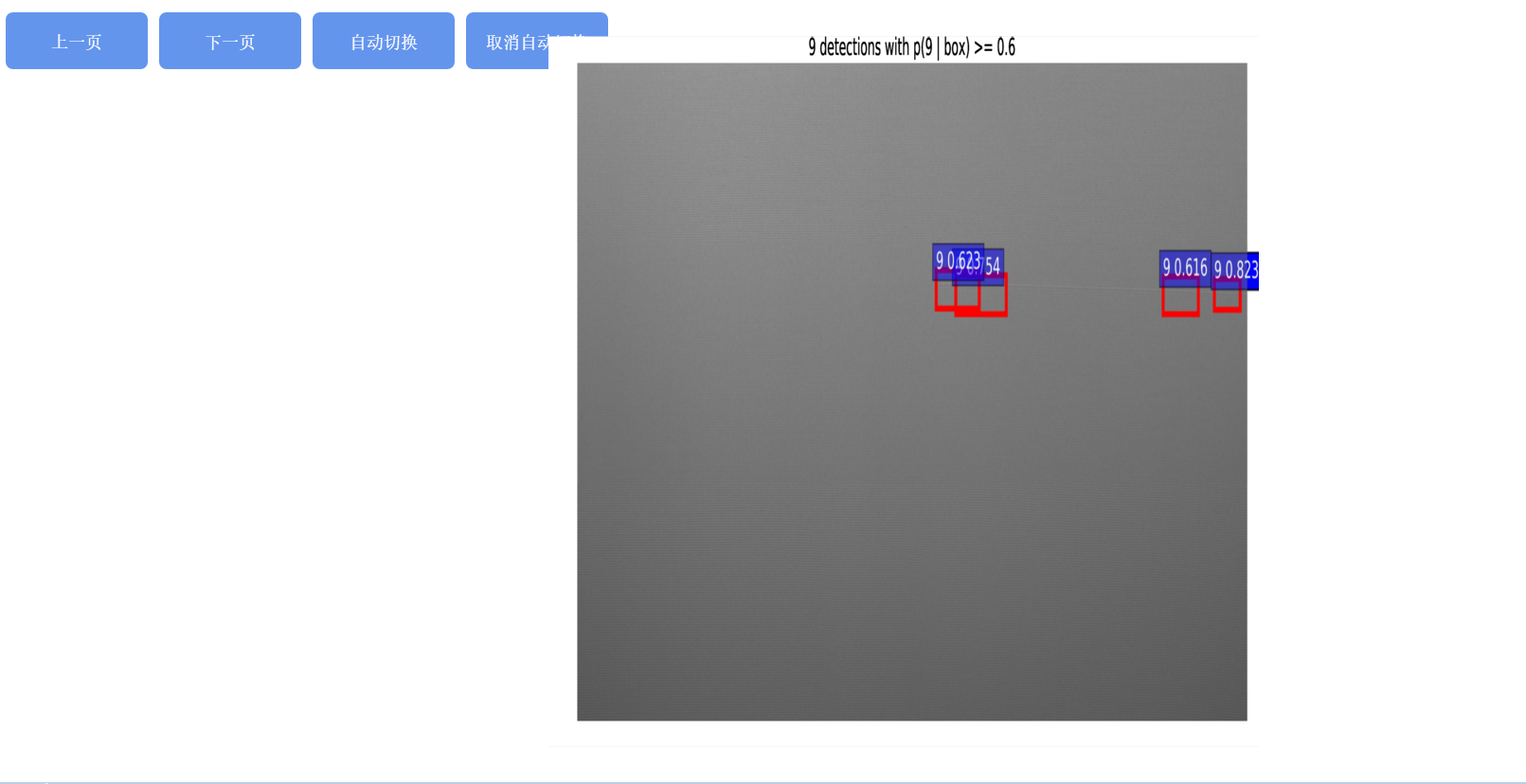
在后台的mysite2路径下输入指令python2 manage.py runserver 192.10.84.31:5000，如果IP地址不同则修改为相应的IP地址。然后在浏览器中输入192.10.84.31:5000，可以看到如下界面：



随意输入用户名，然后点击浏览，上传本地图片。然后点击开始识别，后台就开始识别上传的图片。当识别完成之后，界面会这样显示：



此时可以返回上一界面，点击检测结果按钮，就会出现如下的展示界面：



如果展示多张图片，可以点击上一页或下一页按钮。

**6 相关文件夹**

**下述文件夹中存放重要的数据文件，千万不能删除！！**

mysite2:网站的部署文件，前端的相关内容都在该文件夹下。

static：用来存储检测图片和JS文件的文件夹。

upload:用来存储上传图片的文件夹，当检测结束其中的内容自动删除。

datasets、fast\_rcnn、nms、roi\_data\_layer、rpn、transform、utils：里面存储着用于处理图片的函数，和R-FCN中的同名文件一致。

rfcn：里面存储着训练模型和测试用的部署文件，在检测的时候需要调用。