|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Rev.** | **Author** | **Content of revision** |
| 2018-01-21 | 0.1 | lingming | The draft version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Vimicro kongming 平台

# 概述

Vimicro kongming平台提供一套操作系统无关（Windows/Linux）的人工智能（深度学习）部署迅捷接口。能极大的加速Windows以及Linux 操作系统上人工智能（深度学习）应用的开发。

# 软件环境

## Windows

### 2.1.1 软件安装

顺序安装以下软件：

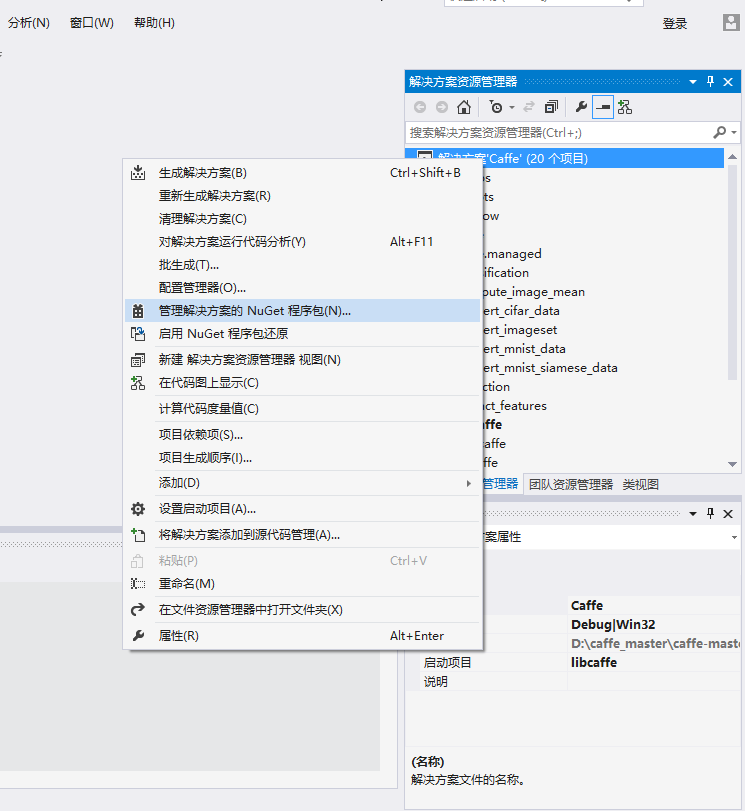
1. 如果需要使用Visual Studio 进行2次开发，先安装Visual Studio(建议2013)，
2. CUDA ，windows 10下默认安装为以下路径 C:\Program Files\NVIDIA Corporation 和 C:\Program Files\NVIDIA GPU Computing Toolkit:
   1. cuda\_8.0.61\_win10.exe
   2. 补丁cuda\_8.0.61.2\_windows.exe
3. CUDNN：
   1. 将压缩包cudnn-8.0-windows10-x64-v5.0-ga.zip 解压；将加压后目录cudnn-8.0-windows10-x64-v5.0-ga\cuda 中的3个文件夹：bin,include,lib 拷贝到目录C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0（如果提示冲突，直接覆盖即可）。
4. 如果需要使用Opencv等第3方软件，需要将压缩包NugetPackages，解压缩到合适路径。
5. Vimicro Only：如果需要深度开发，则将ok\_NuGetCache目录中的所有文件拷贝到类似目录C:\Users\xxxxxx\AppData\Local\NuGet\Cache.（可以跳过上述步骤4）

### 2.1.2 开发环境（Vimicro Only）：

1. 下载Wisdom 源码：

svn checkout <http://10.0.2.88/svn/ArSdk/trunk/caffe-master>

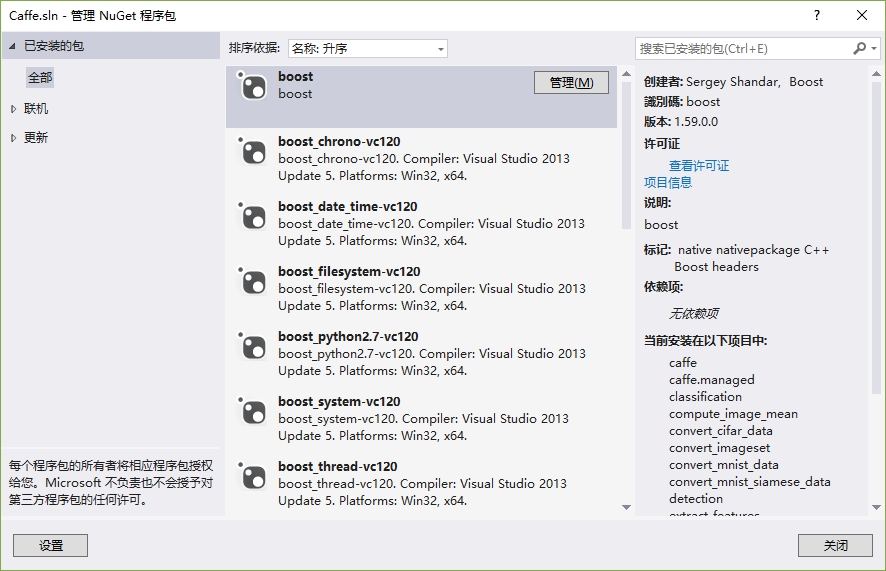
1. 使用Visual Studio 打开工程：caffe-master\windows\ Caffe.sln
2. 右键点击“解决方案’caffe’”, 选择”管理解决方案的NUGet程序包”



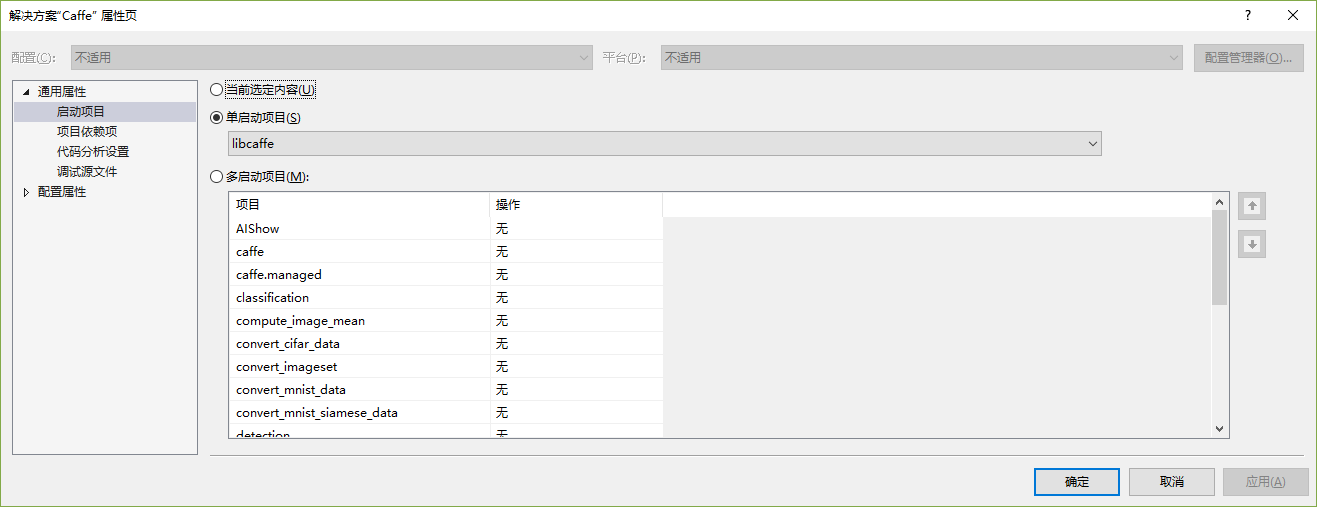
1. 点击右上角“还原”按键，等待安装程序完成（如果之前已经将ok\_NuGetCache目录拷贝到正确的位置，则很快完成；否则联网下载时间较长，且可能失败）。



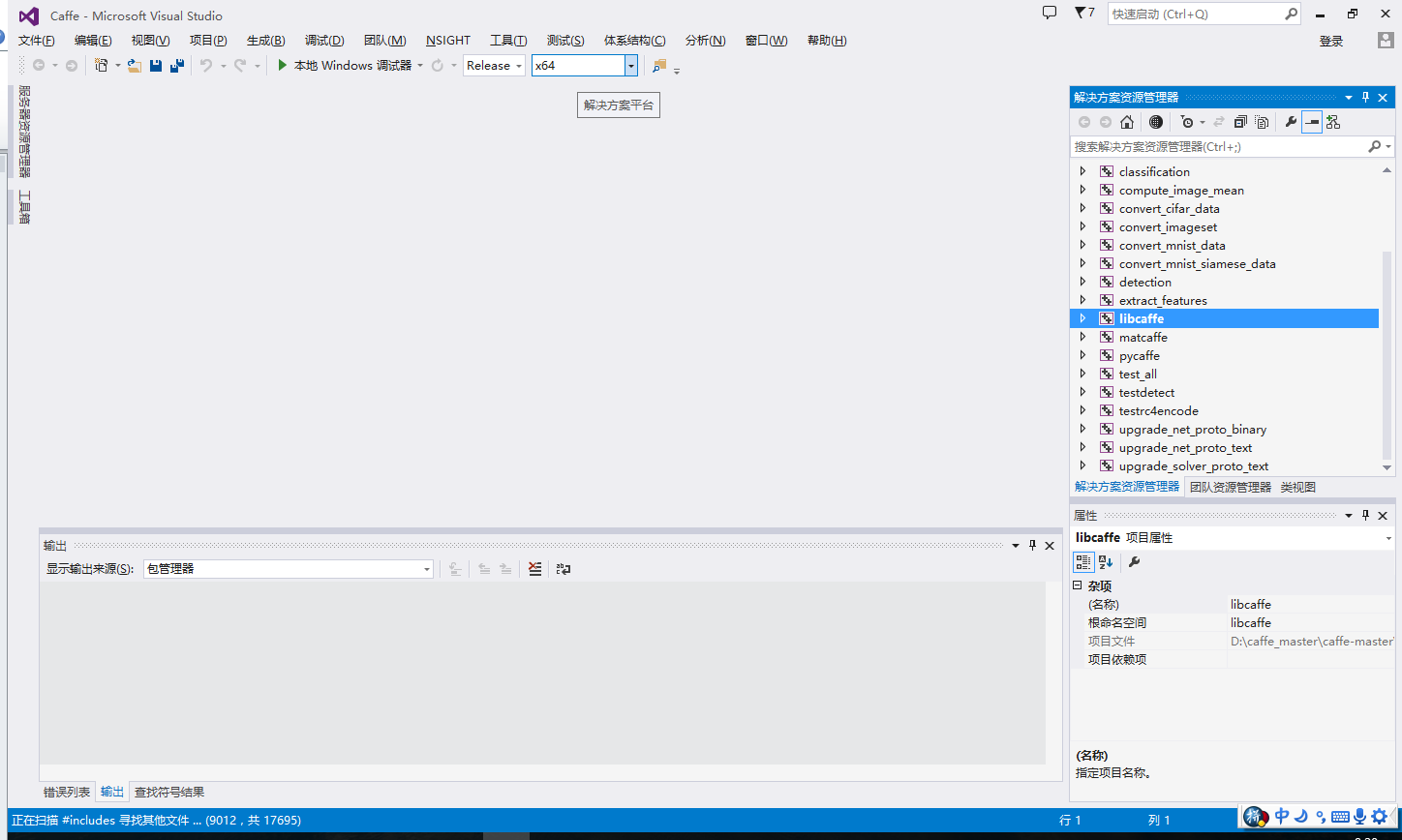
1. 点击左上角”已安装的包“，可以查看已经安装的NuGet包。



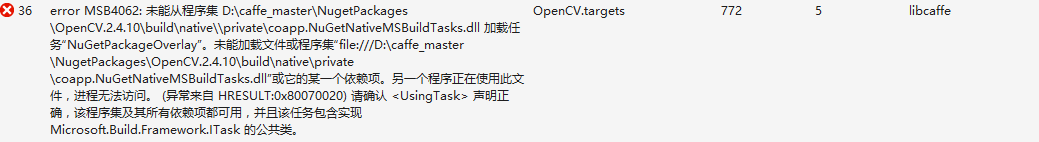
1. 设定解决方案的启动项目为libcaffe. 右键点击“解决方案’caffe’”, 选择“属性”。



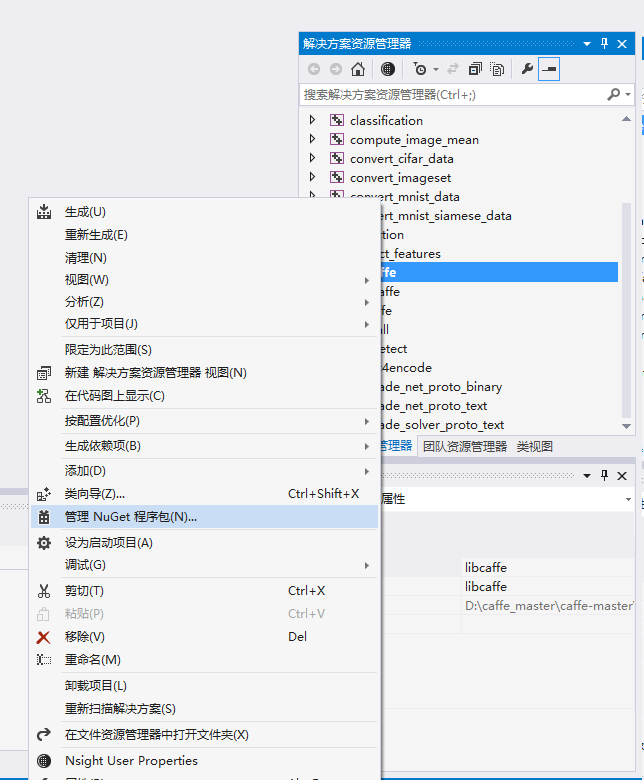
1. 设定解决方案平台为对应的，比如为 Release/X64



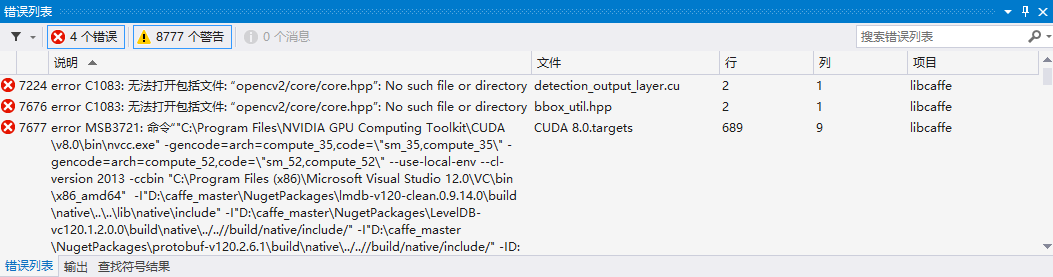
1. 右键点击工程libcaffe, 选择“生成”，等待编译结束。可能会出现以下错误，先跳过：



1. 右键点击工程caffe 和 detection 选择“生成”，等待编译结束。可能会出现同样错误。
2. 右键点击工程libcaffe，选择“管理NuGet程序包”



1. 在已安装中将提示出错的 安装包卸载(可能是opencv, gflags 之一)，后关闭。
2. 继续编译工程libcaffe，工程caffe 和 detection，选择“生成”。此时之前的错误会消失，但会提示新的错误，先跳过：



1. 右键点击工程libcaffe，选择“管理NuGet程序包”，选择左侧为“联机”，在右上角输入之前卸载的包，比如OpenCV；网络不好可能要多重复几次。然后点击安装。



1. 继续编译工程libcaffe，工程caffe 和 detection，选择“生成”。此时所有工程应该都能编译通过。如果哪个工程仍然提示类似问题，则按照上述步骤解决即可。
2. 在编译Debug版本时，可能还会出现上述问题，类似解决即可。

## Linux

### 2.1.1 软件安装

1. 使用Docker环境，或者按照以下方式安装：
2. 参考网页：<http://blog.csdn.net/autocyz/article/details/52299889/>
   1. 安装nvidia驱动（如果显卡不是最新的，CUDA 自带的驱动应该能搞定，可以跳过此步骤）
   2. 安装CUDA 8.0（安装过程中，如果提示此版本的显卡驱动 比已经安装的显卡驱动老，则选择不安装显卡驱动；如果没有上述步骤a）则默认安装）
   3. 安装CUDNN 5.1
   4. 安装opencv 2.4.13(不要安装3.0)：
      1. pip uninstall opencv-python,卸载opencv for python 3.3.0.10。
      2. 先安装一堆依赖库：sudo apt-get install --assume-yes libopencv-dev build-essential cmake git libgtk2.0-dev pkg-config python-dev python-numpy libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev libtiff5-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libxine2-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev libtbb-dev libqt4-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils unzip
      3. sudo apt-get install build-essential cmake git
      4. sudo apt-get install ffmpeg libopencv-dev libgtk-3-dev python-numpy python3-numpy libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev libtiff5-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libxine2-dev libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libv4l-dev libtbb-dev qtbase5-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils unzip
      5. 按照该步骤[http://blog.csdn.net/solomon1558/article/details/519672 安装opencv 2.4.13](http://blog.csdn.net/solomon1558/article/details/519672%20安装opencv%202.4.13)。
   5. 安装caffe：只需要安装各种依赖包即可。无需下载和编译caffe, 将使用Vimicro发布的库。

# 接口介绍：

## API

目前Wisdom平台提供如下API：

1. 初始化（目前下属两个名字没有实际意义, 可以为任意值）：

//objectName表示检测类型，例如：apple或者apple; gun

//DeviceName表示设备型号：例如 : GD\_AT10080B

int DetectionInit(char \*objectName, char \*DeviceName);

1. 检测文件（目前只支持image 文件，不支持video文件）：

//imagesPath表示图片路径, 例如:D : \1.jpg或者D:\1.jpg; D:\2.jpg

//missErrorRatio表示疑似度（目前没有实际意义）

char\* Detection(char \*imagesPath, int missErrorRatio);

返回jason或者其它格式字符串（根据config配置）。

Json文件说明：

{

"result": [{

"imgname": "000021.jpg",　//文件名

"width": 336,　　　　　　　//图片宽度

"height": 500,　　　　　//图片高度

"coords": [{　　　　　　//矩形框

"objecttype": "lian",//类别

"score": 80,　　　　　//得分，最高１００分

"x1": 85,　　　　　//左上顶点x坐标

"y1": 200,　　　　 //左上顶点y坐标

"x2": 107, 　　　　//右上顶点x坐标

"y2": 200, 　　　　//右上顶点x坐标

"x3": 107, 　　　　//右下顶点x坐标

"y3": 235, 　　　　//右下顶点x坐标

"x4": 85, 　　　　 //左下顶点x坐标

"y4": 235　　　　 //左下顶点x坐标

}]

}]

}

1. 检测OPENCV帧:

//org\_img表示输入图片

//missErrorRatio表示疑似度（目前没有实际意义）

char\* DetectionMat(cv::Mat& org\_img, int missErrorRatio);

返回jason或者其它格式字符串（根据config配置）。

1. 把结果画到图片上：

//org\_img表示输入图片

// pBox为之前返回jason或者其它格式字符串（根据config配置）

bool DetectionDraw(cv::Mat& org\_img, char\* pBox);

1. 按照voc的格式，保存所有结果，包括JPEGImages/Annotations/AnnotationJpegs

// src 源目录

// dst 目标目录

// fn 文件名

// pBox, Detection返回的结果

//savenobox，没有框时，是否保存AnnotationJpegs信息

## 配置文件

每个应用场景需要指定配置文件，对AI系统进行部分配置，不同的应用下配置会不一样：

1. 配置检测模型config.cfg（模型1）：

gpus = 0 设定使用的GPU号 客户可修改

weight = config/detect/model\_300.bin 客户可以更新为release的新模型

outmode = json 检测输出结果类型，可以是json或者xml客户可修改

person\_scores = 0.3确认是人的可能性，大于改门限值的人会进行后续处理，值较小可能误判，值太大可能会丢失一些人 客户可修改

encry = none 或者为rc4 是否使用加密的权重和模型，vimicro可修改

weight = config/detect/VGG\_coco\_SSD\_500x500\_iter\_200000.caffemodel vimicro可修改

model = config/detect/deploy.prototxt vimicro可修改

labelmapfile = config/detect/labelmap\_coco.prototxt vimicro可修改

postmode = classify\_uniform vimicro可修改

runmode = normal vimicro可修改

background\_scores = 1.0 vimicro可修改

meannum = 3

mean\_0 = 0.0 #104.0 vimicro可修改

mean\_1 = 0.0 #117.0 vimicro可修改

mean\_2 = 0.0 #123.0 vimicro可修改

1. 配置分类模型subconfig.cfg(模型2), 仅介绍和1有区别部分：

……

none\_scores = 0.6 判断是none种类的得分门限客户可修改

none\_color = 0xFF0000 none种类的框颜色（RGB）客户可修改

clothe\_scores = 0.6 判断是none种类的得分门限客户可修改

clothe\_color = 0x0000FF none种类的框颜色（RGB）客户可修改

full\_scores = 0.6 判断是none种类的得分门限客户可修改

full\_color = 0x00FF00 none种类的框颜色（RGB）客户可修改

## 加解密(vimicro only)

提供给客户的模型必须是加密的模型，加密工具代码见testrc4encode.cpp，vc有对应的工程，linux下也有对应的编译脚本。

使用举例：

./testrc4encode.bin 0 test.prototxt　test.caffemodel model.bin 12345678901234567890

参数说明

0: 加密或者解密，０-加密，１-解密

test.prototxt：prototxt文件

test.caffemodel：caffemodel文件

model.bin：生成的加密文件

12345678901234567890：密钥

注意事项：

1. 参数输入顺序，特别是prototxt文件和caffemodel文件，不能错。
2. prototxt文件input层格式必须是

　　layer {

name: "VGG\_VOC0712\_SSD\_300x300\_deploy"

type: "Input"

top: "data"

input\_param {

shape {

dim:1

dim:3

dim:300

dim:300

}

}

}

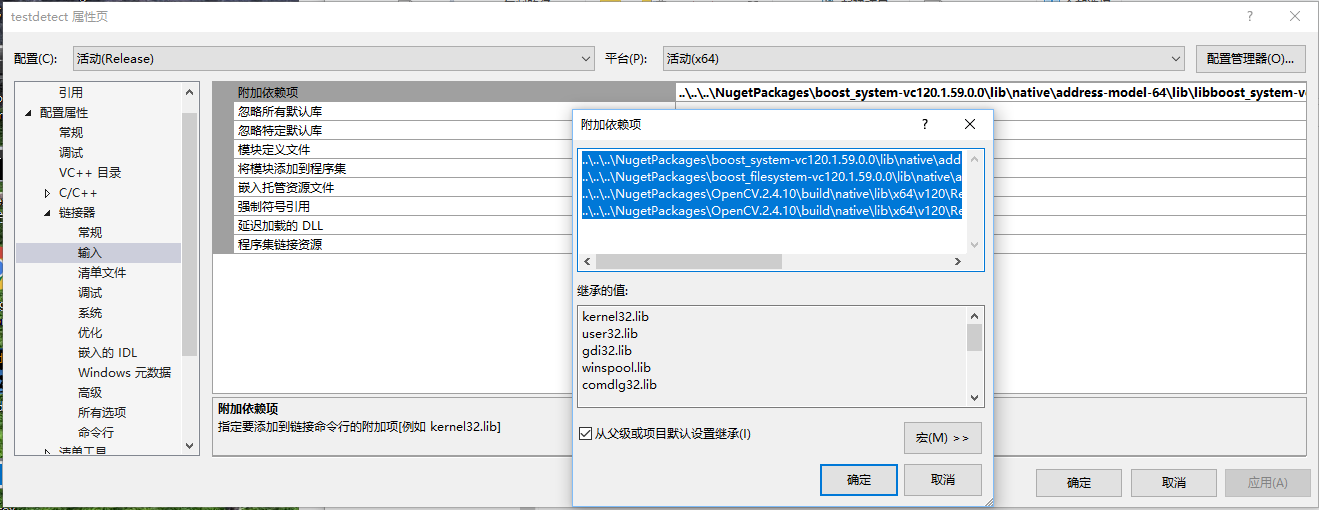
## 开发示例

### windows Visual Studio 2013

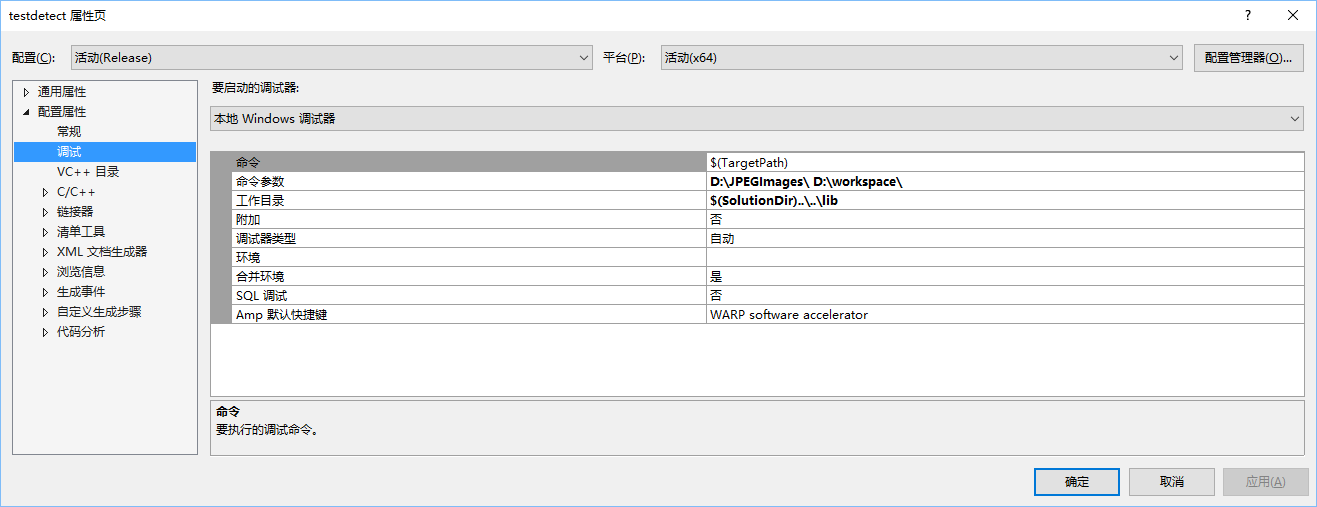
1. 将Visual Studio 示例project-release.zip 解压缩
2. 将依赖库NugetPackages.zip解压缩到同级目录，比如：



1. 打开工程project-release\sample\testdetect\ testdetect.sln
2. 默认不需要修改，如果依赖库的路径改变，则右键点击工程属性修改：连接器->输入->附加依赖项。



1. 修改调试中的命令参数为 待处理图像目录（以”\”结尾）， 和输出目录；确保工作目录如下图：



1. 将project-release/config文件夹拷贝到project-release\lib目录下；
2. 编译运行。

### 3.3.2 linux(ubuntu)