龙芯中科
LOONGSON TECHNOLOGY

人工智能应用解析

提纲

CONTENTS

一、OpenCV实现讲解

龙芯中科
LOONGSON TECHNOLOGY

OpenCV简介

OpenCV 是一个基于 BSD 许可(开源)发行的跨平台计算机视觉和机器学习软件库,可以运行在 Linux、Windows、Android 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成,同时提供了 Python、Ruby、MATLAB 等语言的接口,实现了图像处理和计算机视觉方面的很多通用算法。

OpenCV 用 C++语言编写,它具有 C ++,Python,Java 和 MATLAB 接口,并支持 Windows,Linux,Android 和 Mac OS,OpenCV 主要倾向于实时视觉应用,并在可用时利用 MMX和 SSE 指令, 如今也提供对于 C#、Ch、Ruby,GO 的支持。

OpenCV 提供的视觉处理算法非常丰富,并且它部分以 C 语言编写,加上其开源的特性,处理得当,不需要添加新的外部支持也可以完整的编译链接生成执行程序,所以很多人用它来做算法的移植。

一、OpenCV实现讲解

编程语言

OpenCV 用 C++语言编写,它的主要接口也是 C++语言,但是依然保留了大量的 C 语言接口。OpenCV库也有大量的 Python、Java and MATLAB/OCTAVE(版本 2.5)的接口。这些语言的API 接口函数可以通过在线文档获得。如今也提供对于 C#、Ch、Ruby 的支持。

所有新的开发和算法都是用 C++接口。一个使用 CUDA 的 GPU 接口也于 2010 年 9 月开始实现。

支持操作系统

OpenCV 可以在 Windows,Android,Maemo,FreeBSD,OpenBSD,iOS,Linux 和 MacOS 等平台上运行。使用者可以在 SourceForge 获得官方版本,或者从 SVN 获得开发版本。OpenCV 也是用 CMake。

在 Windows 上编译 OpenCV 中与摄像输入有关部分时,需要 DirectShow SDK 中的一些基类。该 SDK 可以从预先编译的 Microsoft Platform SDK(or DirectX SDK 8.0 to 9.0c / DirectX Media SDK prior to 6.0)的子目录 Samples\Multimedia\DirectShow\BaseClasses 获得。

一、OpenCV快速入门

龙芯中科 LOONGSON TECHNOLOGY

1.1、读取图像

使用 cv.imread()函数读取图像。cv.imread()函数需要两个参数，第一个参数是图像路径。第二个参数是一个标志,它指定了读取图像的方式。

cv.IMREAD_COLOR: 加载彩色图像。任何图像的透明度都会被忽视。它是默认标志。

cv.IMREAD_GRAYSCALE:以灰度模式加载图像

cv.IMREAD_UNCHANGED:加载图像,包括 alpha 通道

注意 除了这三个标志,你可以分别简单地传递整数 1、0 或-1。请参见下面的代码:

```
import numpy as np
import cv2 as cv
# 加载彩色灰度图像
img = cv.imread('img.jpg', 0)
```

即使图像路径错误,它也不会引发任何错误,但是 printf(img)会给出 None

一、OpenCV快速入门

龙芯中科 LOONGSON TECHNOLOGY

1.2、显示图像

使用函数 cv.imshow()在窗口中显示图像。窗口自动适合图像尺寸。

第一个参数是窗口名称,它是一个字符串。第二个参数是我们的对象。你可以根据需要创建任意多个窗口,但可以使用不同的窗口名称。

```
cv.imshow('image', img)
cv.waitKey(0)
cv.destroyAllWindows()
```

请参见下面的代码:

```
cv.namedWindow('image', cv.WINDOW_NORMAL)
cv.imshow('image', img)
cv.waitKey(0)
cv.destroyAllWindows()
```

cv.waitKey()是一个键盘绑定函数。其参数是以毫秒为单位的时间。该函数等待任何键盘事件指定的毫秒。如果您在这段时间内按下任何键,程序将继续运行。如果**0**被传递,它将无限期地等待一次敲击键。它也可以设置为检测特定的按键,例如,如果按下键 a 等；

cv.destroyAllWindows()会破坏我们创建的所有窗口。如果要销毁任何特定的窗口,请使用函数

cv.destroyWindow()在其中传递确切的窗口名称作为参数。

1.3、写入图像

使用函数 cv.imwrite()保存图像。第一个参数是要保存文件名,第二个参数是数据流。

程序:

```
import cv2 as cv
# 加载彩色灰度图像
img = cv.imread('img.jpg', 0)
cv.namedWindow('image', cv.WINDOW_NORMAL)
cv.imshow('image', img)
cv.waitKey(0)
cv.destroyAllWindows()
```

一、OpenCV快速入门

龙芯中科 LOONGSON TECHNOLOGY

1.4、读取摄像头

通常情况下,需用摄像机捕捉实时画面。Opencv 提供了一个非常简单的界面。从摄像头捕捉一段视频 ,将其转换成灰度视频并显示出来。

cap.read()返回布尔值(True/ False)。如果正确读取了帧,它将为 True。因此,可以通过检查此返回值来检查视频的结尾。

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0) if not cap.isOpened():
    print("Cannot open camera")
    exit() while True:
    # 逐帧捕获
    ret, frame = cap.read()
    # 如果正确读取帧, ret 为 True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    # 显示结果帧
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break# 完成所有操作后,释放
cap.release()
cv.destroyAllWindows()
```

1.5、读取视频

与从相机捕获类似,只需要将摄像机索引更改为视频文件路径。另外,在显示时,请使用适当的时间 cv.waitKey()。如果太小,则视频将非常快,而如果太大,则视频将变得很慢(可以用于显示慢动作)。正常情况下 25 毫秒就可以了。

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture('vtest.avi')while cap.isOpened():
    ret, frame = cap.read()
    # 如果正确读取帧,ret 为 True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

一、OpenCV快速入门

龙芯中科 LOONGSON TECHNOLOGY

1.6、视频保存

保存这个视频。对于图像,它非常简单,只需使用 cv.imwrite()。这里还需要做一些工作。这次创建一个 VideoWriter 对象。我们应该指定输出文件名(例如: output.avi)。然后我们应该指定 FourCC 代码(详见下一段)。然后传递帧率的数量和帧大小。最后一个是颜色标志。如果为 True,编码器期望颜色帧,否则它与灰度帧一起工作。

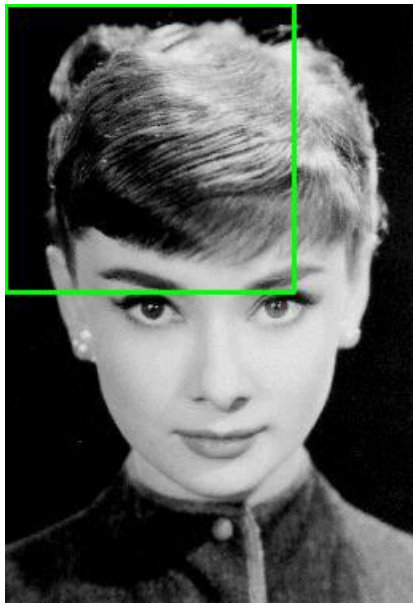
```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)# 定义编解码器并创建 VideoWriter 对象
fourcc = cv.VideoWriter_fourcc(*'XVID')
out = cv.VideoWriter('output.avi', fourcc, 20.0, (640,480))while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    frame = cv.flip(frame, 0)
    # 写翻转的框架
    out.write(frame)
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break# 完成工作后释放所有内容
cap.release()
out.release()
cv.destroyAllWindows()
```

二、Haar级联检测器

Haar级联检测器

haar级联算法是opencv最流行的目标检测算法，主要优点是速度太快了，尽管许多算法（如hog+线性svm、ssd、更快的RCNN、yolo等等）比haar级联算法更精确。但如果需要纯粹的速度，就是无法打败opencv的haar cascades。

haar级联检测的效果如下：



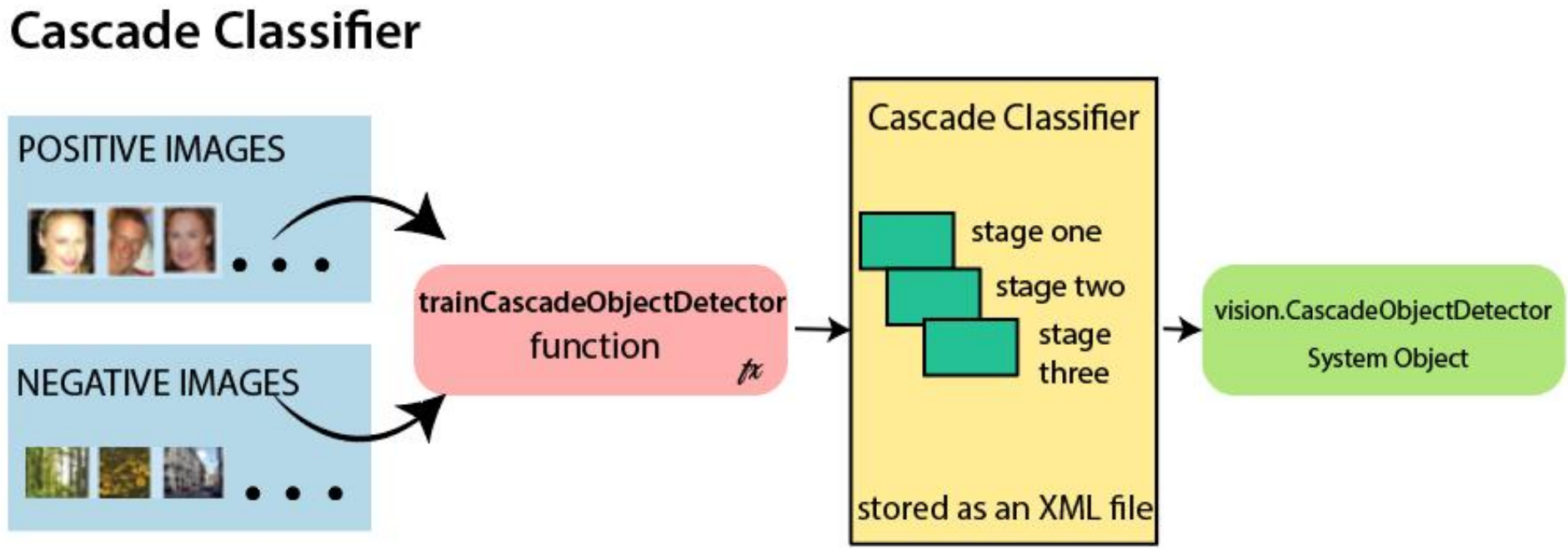
可以看到图中有固定大小的面在移动，就可以“训练”分类器来识别图像的给定区域是否包含人脸。

二、Haar级联检测器

龙芯中科 LOONGSON TECHNOLOGY

haar级联是什么？

HAAR 级联是一种机器学习方法，其中级联函数是从大量正面和负面图像中训练出来的。正图像是由人脸组成的图像，负图像是没有人脸的图像。在人脸检测中，图像特征被视为从图片中提取的数字信息，可以将一幅图像与另一幅图像区分开来。



二、Haar级联检测器

龙芯中科 LOONGSON TECHNOLOGY

haar级联预训练的模型

OpenCV 已经包含各种针对面部、眼睛、微笑等的预训练分类器。这些 XML 文件存储在 opencv/data/haarcascades/ 文件夹中

haarcascade_frontalface_default.xml：检测面部

haarcascade_eye.xml：检测左眼和右眼

haarcascade_smile.xml：检测面部是否微笑

haarcascade_eye_tree_eyeglasses.xml：检测是否带眼镜

haarcascade_frontalcatface.xml：检测猫脸

haarcascade_frontalcatface_extended.xml：检测猫脸延伸

haarcascade_frontalface_alt.xml：检测猫脸属性

haarcascade_frontalface_alt_tree.xml

haarcascade_frontalface_alt2.xml

haarcascade_fullbody.xml：检测全身

haarcascade_lefteye_2splits.xml：检测左眼

haarcascade_licence_plate_rus_16stages.xml：检测证件

haarcascade_lowerbody.xml：检测下半身

haarcascade_profileface.xml

haarcascade_righteye_2splits.xml：检测右眼

haarcascade_russian_plate_number.xml：检测俄罗斯字母车牌号

haarcascade_upperbody.xml：检测上半身

二、编写程序

龙芯中科
LOONGSON TECHNOLOGY

加载人脸识别haar级联文件

```
# 加载Haar级联数据文件，用于检测人面
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# 查找人脸
faces = face_cascade.detectMultiScale(img, 1.3, 2,minSize=(150,150),maxSize=(300,300))

# faces包含了所有的人脸，可以得到其x, y, w, h坐标
for (x, y, w, h) in faces:
    ......
```

二、编写程序

龙芯中科 LOONGSON TECHNOLOGY

完整代码

```
import cv2

class Face:
    def __init__(self):
        # 加载Haar级联数据文件，用于检测人面
        self.face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

    def find_face(self, img):
        # 灰度转换
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # 查找人脸
        faces = self.face_cascade.detectMultiScale(gray, 1.3, 2,minSize=(150,150),maxSize=(300,300))
        for (x, y, w, h) in faces:
            # 画出人面所在位置并灰度处理
            img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
        return img
```

三、NCNN自主框架

龙芯中科 LOONGSON TECHNOLOGY

ncnn图像识别

ncnn 是一个为手机端极致优化的高性能神经网络前向计算框架。ncnn 从设计之初深刻考虑手机端的部署和使用。无第三方依赖,跨平台,手机端 cpu 的速度快于目前所有已知的开源框架。基于 ncnn,开发者能够将深度学习算法轻松移植到手机端高效执行,开发出人工智能 APP,将 AI 带到你的指尖。 ncnn 目前已在腾讯多款应用中使用,如 QQ, Qzone, 微信,天天 P 图等。

https://github.com/Tencent/ncnn

ncnn 的模型支持 caffe、tensorflow、pytorch 等主流神经网络框架部署,有相应的模型转换工具。支持大部分常用的 CNN 网络:

Classical CNN: VGG/AlexNet/GoogleNet/Inception ...

Practical CNN: ResNet/DenseNet/SENet/FPN ...

Light-weight CNN: SqueezeNet/MobileNetV1/V2/V3/ShuffleNetV1/V2/MNasNet ...

Face Detection: MTCNN RetinaFace scrfd ...

Detection:VGG-SSD/MobileNet-SSD/SqueezeNet-SSD/MobileNetV2-SSDLite/MobileNetV3-SSDLite ...

Detection: Faster-RCNN R-FCN ...

Detection: YOLOV2 YOLOV3 MobileNet-YOLOV3 YOLOV4 YOLOV5 YOLOX ...Detection: NanoDet

Segmentation: FCN PSPNet UNet YOLACT ...

Pose Estimation: SimplePose ...

三、NCNN自主框架安装

龙芯中科 LOONGSON TECHNOLOGY

3.1 前期准备:修改 msa.h

gcc-8.5 修复了 mips msa fmadd intrinsic 参数顺序错误的 bug。龙芯 2K 开发板上的 gcc 是 8.3 版本,要手工修改 msa.h 绕过 bug。
用编辑器打开 /usr/lib/gcc/mips64el-linux-gnuabi64/8/include/msa.h,找到 __msa_fmadd _w,修改为正确的参数顺序。

```
// #define __msa_fmadd_w __builtin_msa_fmadd_w
#define __msa_fmadd_w(a, b, c) __builtin_msa_fmadd_w(c, b, a)
```

3.2 编译安装 ncnn

全局开启了 msa 和 loongson-mmi 扩展指令支持。ncnn 已支持直接用 simpleocv 替代 opencv 编译出 examples,如果要使用 simpleocv 则将-DNCNN_SIMPLEOCV 设为 ON,这里设
为 OFF 则采用龙芯 2K 开发板系统上的 opencv4.5.3。

```
mkdir -p build
cd build
cmake -DNCNN_DISABLE_RTTI=ON -DNCNN_DISABLE_EXCEPTION=ON -
DNCNN_RUNTIME_CPU=OFF
-DNCNN_MSA=ON -DNCNN_MMI=ON -DNCNN_SIMPLEOCV=OFF ..
cmake --build . -j 2
cmake --build . --target install
```

三、NCNN自主框架测试

龙芯中科 LOONGSON TECHNOLOGY

3.3 测试 benchncnn

benchncnn 将会测试各个模型的推理运行时间。将 ncnn/benchmark/*.param 拷贝到 ncnn/build/benchmark/

```
./benchncnn 8 2 0 -1 0
```

3.4 测试 example

build/example 文件夹下会生成各个模型的 demo,需要先下载对应的模型参数文件(后面会讲如何生成),地址:https://github.com/nihui/ncnn-assets/tree/master/models
如测试 scrfd(人脸识别)就需要: scrfd_500m-opt2.bin 、 scrfd_500m-opt2.param。将这两个参数文件和测试图片拷贝到 build/example。

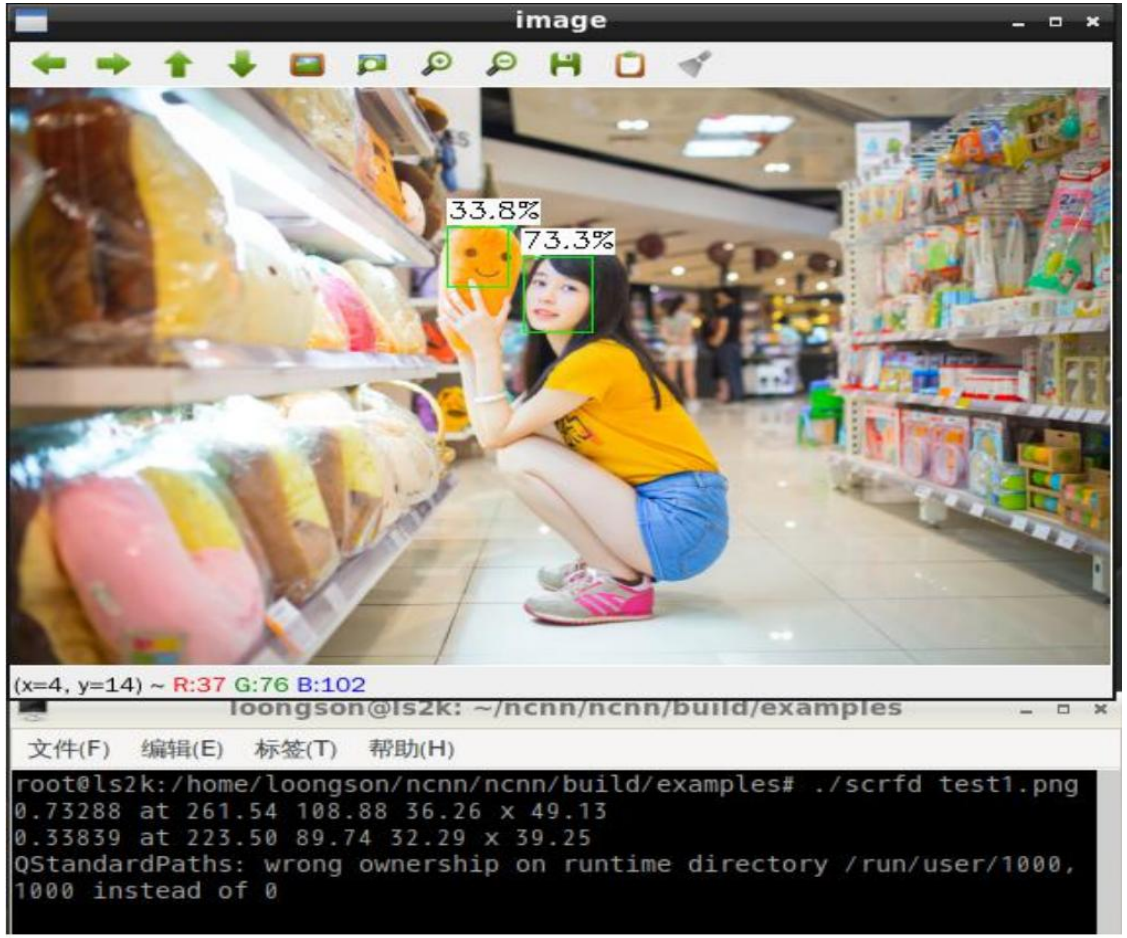
```
./scrfd test.jpg
```

三、NCNN自主框架测试

龙芯中科 LOONGSON TECHNOLOGY

3.5 测试 yolov5

yolov5(多分类目标检测),同样将 yolov5s_6.0.bin,yolov5s_6.0.param下载拷贝到目录下。
./yolov5 test.jpg



四、安装Caffe

龙芯中科 LOONGSON TECHNOLOGY

利用 anaconda 来安装 Caffe 十分简单快捷,仅需几条代码即可安装好所需的环境,并且anaconda 有助于服务器下不同用户之间的 package 版本、系统设置各自独立,互不干扰。

1)、创建虚拟环境

conda create -n your-envs-name
例如:
conda create -n mycaffe

2)、激活环境

conda activate mycaffe

3)、安装 Caffe

conda install -c engility caffe-gpu #GPU 版本,会自动安装好驱动对应的 CUDA、cudnn
或者
conda install -c engility caffe #CPU 版本

四、安装Caffe

龙芯中科 LOONGSON TECHNOLOGY

4)、创建虚拟环境

conda activate mycaffe
caffe #会有指令提示
python #进去 python
import caffe #加载 caffe 成功

5)、从 github 上下载 Caffe

sudo git clone https://github.com/BVLC/caffe.git

6)、测试

cd python python
在 python 中输入:
import caffe
注:创建共享文件夹的方法 在 Linux 系统中的 shfile 文件夹
mount -t vboxsf sh /home/v3edu/shfile

四、Caffe使用

龙芯中科 LOONGSON TECHNOLOGY

1、简介

caffe 是一个清晰而高效的深度学习框架,纯粹的 C++/CUDA 架构,支持命令行、Python 和 MATLAB 接口,可以在 CPU 和 GPU 直接无缝切换;
caffe 的主要优势:
(1)CPU 与 GPU 的无缝切换;
(2)模型与优化都是通过配置文件来设置,无需代码;
caffe 的使用接口有命令行,python 跟 matlab,在训练时只需命令行执行配置好的文件即可,至于训练好的模型可以使用 python 与 matlab 的接口进行调用。

2、LeNet 模型

LeNet 是手写数字库 MNIST 上应用比较经典的模型,具有 7 层网络结构,分别是卷积-下采样-卷积-下采样-全连-全连-分类层。
训练模型之前需要先准备好训练数据 MNIST,执行以下命令可以下载 MNIST 数据库:
./data/mnist/get_mnist.sh
由于 caffe 支持的数据类型不包括图像类型,所以常规做法需要将图像类型转为 lmdb 类型:
./examples/mnist/create_mnist.sh
准备好数据之后,我们需要定义我们的网络模型,在 caffe 中是通过.prototxt 配置文件来定义的,mnist 的模型文件保存在./examples/mnist/lenet_train_test.prototxt 中

四、Caffe使用

龙芯中科 LOONGSON TECHNOLOGY

3、训练模型

直接执行脚本命令就可以:

```
#!/usr/bin/env sh
set -e
./build/tools/caffe train --solver=examples/mnist/lenet_solver.prototxt $@
```

脚本中指定了 caffe 的路径并设为 train 模式,并以 lenet_solver.prototxt 中的参数开启训练,上面说过 lenet_solver.prototxt 中设定了网络模型配置文件的路径。

执行后可以看到:首先会读取配置文件初始化网络跟优化器,紧接着开始优化,可以看到训练过程中每 100 次迭代就会显示一个 loss,每 500 次迭代就会计算一次 test 准确率,总共 10000 次迭代,这些都可以在配置文件中设置。

```
I1112 11:45:36.121951  7854 solver.cpp:239] Iteration 9900 (18.1324 iter/s, 5.515s/100 iters), loss = 0.00620736
I1112 11:45:36.121996  7854 solver.cpp:258]     Train net output #0: loss = 0.00620749 (* 1 = 0.00620749 loss)
I1112 11:45:36.122004  7854 sgd_solver.cpp:112] Iteration 9900, lr = 0.00596843
I1112 11:45:41.599592  7854 solver.cpp:464] Snapshotting to binary proto file examples/mnist/lenet_iter_10000.caffemodel
I1112 11:45:41.618216  7854 sgd_solver.cpp:284] Snapshotting solver state to binary proto file examples/mnist/lenet_iter_10000.solverstate
I1112 11:45:41.644001  7854 solver.cpp:327] Iteration 10000, loss = 0.00333873
I1112 11:45:41.644047  7854 solver.cpp:347] Iteration 10000, Testing net (#0)
I1112 11:45:45.006284  7857 data_layer.cpp:73] Restarting data prefetching from start.
I1112 11:45:45.152503  7854 solver.cpp:414]     Test net output #0: accuracy = 0.9906
I1112 11:45:45.152560  7854 solver.cpp:414]     Test net output #1: loss = 0.0297759 (* 1 = 0.0297759 loss)
I1112 11:45:45.152568  7854 solver.cpp:332] Optimization Done.
I1112 11:45:45.152575  7854 caffe.cpp:250] Optimization Done.
```

四、Caffe使用-模型训练流程

龙芯中科 LOONGSON TECHNOLOGY

1、准备数据

(1)数据来源:任意的 jpg 或其他格式的图像数据;
(2)划分数据为 train 跟 val 数据集,并且使用文本记录好对应的标签;
(3)将数据转化为 lmdb,在 caffe 的根目录下,/build/tools/下有各种可以使用的命令行工作,为了将图像数据转为 lmdb,使用的是 convert_imageset 指令,具体如下:

```
$CAFFE_TOOLS/convert_imageset \
--resize_height=$RESIZE_HEIGHT \
--resize_width=$RESIZE_WIDTH \
--shuffle \
$IMAGE_DATA_ROOT \
$LABEL_DATA_PATH/label.txt \
$OUTPUT_PATH/data_lmdb
```

其中 CAFFE_TOOLS 是命令行路径;resize_height 根据自己需要决定是否要将原始图片 resize

四、Caffe使用-模型训练流程

龙芯中科 LOONGSON TECHNOLOGY

2、训练模型

前面已经讲述了如何配置模型文件跟优化文件,现在需要注意的是如何调用命令行来训练,很简单,只需指定优化配置文件的路径即可;至此模型训练结束,得到了想要的模型,存放在. caffemodel 中。

3、使用模型

(1)test.prototxt 描述的是测试的网络结构;
(2)ip2 表示需要提取特征的层的名字;

4、注意事项

(1)在将 jpg 数据转为 lmdb 时,都需要有 label 的文本信息,一般来说, train 跟 validation数据是带有 label 的,test 数据是没有的,一般是给 test 数据提供 index 文本信息;
(2)在模型训练过程中将导入 train 数据跟 val 数据,其中 train 数据是为了训练模型,val 数据给出识别率为调参提供参考;其中 val 数据不是必须的,根据需要在训练模型配置文件中定义;
(3)训练过程的模型文件既定义了训练模型也定义了验证模型,而提取特征过程的模型文件中只定义测试模型;
(4)在提取的特征中,每张图象的特征的次序跟输入的次序是不一样的,为了得到 caffe中模型训练过程的输入图像的次序,可以通过提取 test 数据的 index 文本信息,也就是 label信息;
(5)提取出来的特征以 lmdb 形式给出,需要根据需要自行转换;

五、实战案例：识别应用手势系统

龙芯中科 LOONGSON TECHNOLOGY

手势识别

训练神经网络模型，识别特定的手势。


0IMG_4370.JPG


1IMG_5919.JPG


2IMG_5529.JPG


3IMG_5953.JPG


4IMG_1162.JPG


5IMG_4770.JPG


6IMG_5822.JPG


7IMG_5315.JPG


8IMG_4924.JPG


9IMG_4357.JPG

五、实战案例：识别应用手势系统

龙芯中科 LOONGSON TECHNOLOGY

5.1 环境准备

1. ubuntu 系统,安装: caffe (可以虚拟机安装,非虚拟机可以使用 GPU 版本加速训练)
2. 龙芯 2k 教育派开发板,安装:ncnn

5.2 数据集处理

1. 下载数据集
2.新建 hand_lenet 文件夹,并且将数据集解压到里面。dataset 目录下每一类都已经按文件夹分类好 0-9 对应 10 个手势数据。新建一个 train 文件夹,将所有分类目录放进里面,项目的整体结构如下。此时还需要创建一个脚本来划分测试集 val 并且创建 txt 索引文件。
3. 创建 txt 索引文件
train.txt 文本内容是训练集图像的相对路径及其对应的 ground-truth label,其中相对路径的根目录是 train/文件夹,每一行记录一张图像的相对路径和对应的 label(所用的手势图像类别依次命名为 0-9),label 和相对路径之间有一个空格符隔开。
val/目录的格式则和 train/目录不同,val/目录下并没有子文件夹,直接存放不同类别的图像,每一类取训练集数据的 10%验证集图像。val.txt 与 train.txt 内容相似,也是图像的相对路径及其对应的 ground-truth label。
通过 python 脚本来划分 val 数据集,同时创建 train.txt 和 val.txt
4. 转换数据格式为 LMDB 文件

五、实战案例：识别应用手势系统

龙芯中科 LOONGSON TECHNOLOGY

5.3 训练模型

1. 准备神经网络模型
神经网络模型采用经典的 Lenet 模型,LeNet 网络结构比较简单,但是刚好适合神经网络的入门学习。也可以采用更先进的模型如 Alexnet、GoogleNet、Resnet,更先进的模型往往意味着更高的准确率和运算资源。lenet.prototxt 是预测时使用的模型文件
训练时所需的文件由 lenet.prototxt 修改得到，直接将其复制一份，重命名 lenet_train_test.prototxt
2. 训练超参数设置文件
lenet_solver_adam.prototxt
3. 设置训练脚本
train_lenet.sh
运行脚本即可训练模型,每 5000 次保存一次模型在 save 文件夹中(要先手动创建),训练 10000 次的模型在 val 数据集下的正确率已经达到了 89%。可以调整参数比如学习率、训练次数、衰减率、优化方式等超参数继续训练。或者更换其他更深的模型。
4. 测试模型
test.py

五、实战案例：识别应用手势系统

龙芯中科 LOONGSON TECHNOLOGY

5.4 部署模型

1. 转换模型

在龙芯派上新建一个 model 文件夹,用于存放训练好的 caffe 模型,所需的文件有 lenet.prototxt 和 lenet_iter_10000.caffemodel。在编译好 ncnn/build/tools/caffe/下,caffe2ncnn

工具能将这两个文件转换成 ncnn 的模型文件,执行命令:

../ncnn/build/tools/caffe/caffe2ncnn lenet.prototxt lenet_iter_10000.caffemodel ncnn_hand.param ncnn_hand.bin

生成的 ncnn_hand.param、ncnn_hand.bin 就是用于部署的 ncnn 模型。

2. 添加 examples 源码

在/ncnn/ncnn/examples/下新建 cpp 文件 ncnn_hand.cpp:

ncnn_hand 就是我们的预测处理程序,首先要加载模型参数,对传入的图片进行处理(与训练时一致),最后从模型中获取预测分数,将最大值对应的标号进行输出,即可进行手势识别。

3. 修改 CMakeLists.txt

4. 编译

在 ncnn/build/examples/下打开命令行执行编译命令:make

5. 执行预测

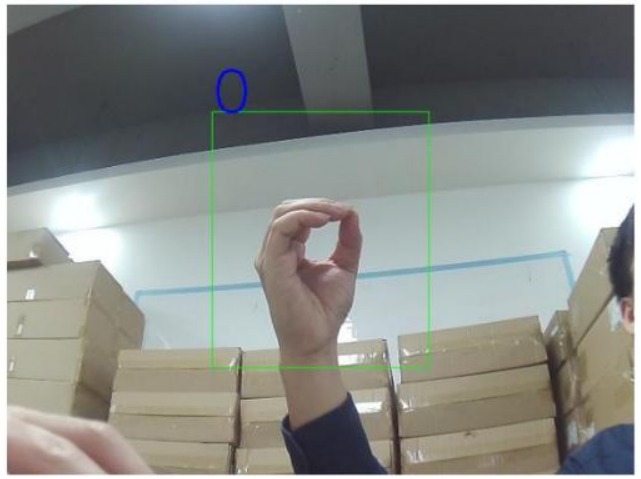
./ncnn_hand 0IMG_4370.JPG # 预测图片

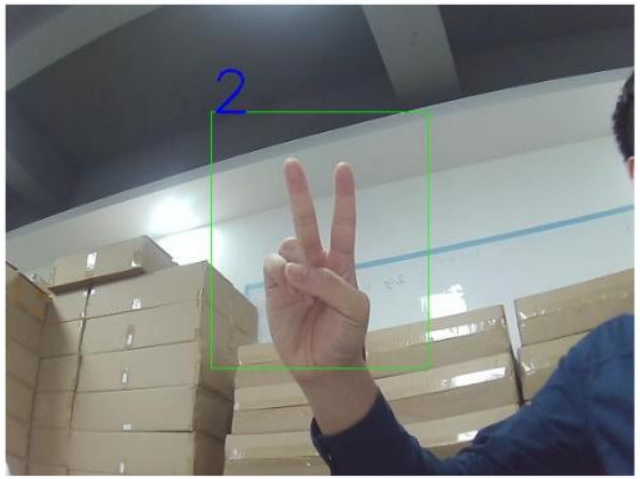
```
endif()

if(OpenCV_FOUND OR NCNN_SIMPLEOCV)
    if(OpenCV_FOUND)
        message(STATUS "OpenCV library: ${OpenCV_INSTALL_PATH}")
        message(STATUS "    version: ${OpenCV_VERSION}")
        message(STATUS "    libraries: ${OpenCV_LIBS}")
        message(STATUS "    include path: ${OpenCV_INCLUDE_DIRS}")

        if(${OpenCV_VERSION_MAJOR} GREATER 3)
            set(CMAKE_CXX_STANDARD 11)
        endif()
    endif()

    include_directories(${CMAKE_CURRENT_SOURCE_DIR}/../src)
    include_directories(${CMAKE_CURRENT_BINARY_DIR}/../src)

    ncnn_add_example(ncnn_hand)
    ncnn_add_example(squeezenet)
    ncnn_add_example(squeezenet_c_api)
    ncnn_add_example(fasterrcnn)
    ncnn_add_example(rfcn)
    ncnn_add_example(yolov2)
    ncnn_add_example(yolov3)
```

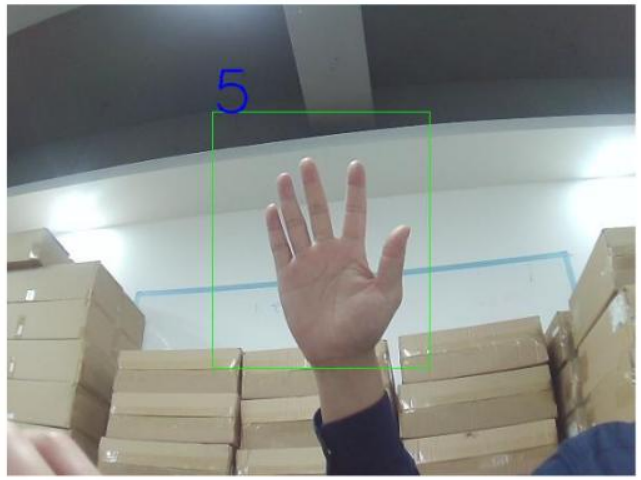
五、实战案例：识别应用手势系统

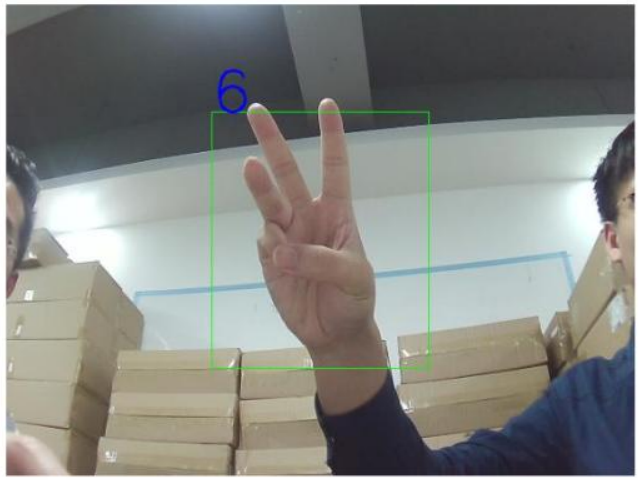
龙芯中科 LOONGSON TECHNOLOGY

5.5 识别结果









六、小任务

龙芯中科
LOONGSON TECHNOLOGY

- 1、ncnn计算框架的示例开发
- 2、手势识别系统

感谢观看！