tensorflow 的android demo中，detector代码修改，解决检测框不显示问题

原创 2017年09月12日 14:10:22

标签：android (http://so.csdn.net/so/search/s.do?q=android&t=blog) / tensorflow (http://so.csdn.net/so/search/s.do?q=tensorflow&t=blog) / deeplearning (http://so.csdn.net/so/search/s.do?q=deeplearning&t=blog)

451

编译安装tensorflow_demo，断断续续搞了2周多。
主要问题是workspace文件修改的不正确，sdk和ndk的api_level没有搞匹配，
64位的机器编译，ndk的api_level应该要比21高，不是14
还有，sdk和ndk的路径一定要写正确
其他就没问题了。
总的来说，安装bazel有点麻烦，
编译tensorflow源码其实挺简单的，就是配置要细心。

修改代码属于DetectorActivity.java
黑色是原来代码，红色为添加代码。
修改后，运行如下代码编译：
sudo bazel build -c opt //tensorflow/examples/android:tensorflow_demo

然后，运行命令
sudo adb install -r bazel-bin/tensorflow/examples/android/tensorflow_demo.apk

将demo安装到手机上，就可以看到效果了

```
/*
 * Copyright 2016 The TensorFlow Authors. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *       http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.tensorflow.demo;

import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
```

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.RectF;
import android.graphics.Typeface;
import android.media.Image;
import android.media.Image.Plane;
import android.media.ImageReader;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.SystemClock;
import android.os.Trace;
import android.util.Size;
import android.util.TypedValue;
import android.view.Display;
import android.widget.Toast;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import java.util.Vector;
import org.tensorflow.demo.OverlayView.DrawCallback;
import org.tensorflow.demo.env.BorderedText;
import org.tensorflow.demo.env.ImageUtils;
import org.tensorflow.demo.env.Logger;
import org.tensorflow.demo.tracking.MultiBoxTracker;
import org.tensorflow.demo.R;

/**
 * An activity that uses a TensorFlowMultiBoxDetector and ObjectTracker to detect and then track
 * objects.
```

```
 */
public class DetectorActivity extends CameraActivity implements OnImageAvailableListener {
  private static final Logger LOGGER = new Logger();

  // Configuration values for the prepackaged multibox model.
  private static final int MB_INPUT_SIZE = 224;
  private static final int MB_IMAGE_MEAN = 128;
  private static final float MB_IMAGE_STD = 128;
  private static final String MB_INPUT_NAME = "ResizeBilinear";
  private static final String MB_OUTPUT_LOCATIONS_NAME = "output_locations/Reshape";
  private static final String MB_OUTPUT_SCORES_NAME = "output_scores/Reshape";
  private static final String MB_MODEL_FILE = "file:///android_asset/multibox_model.pb";
  private static final String MB_LOCATION_FILE =
      "file:///android_asset/multibox_location_priors.txt";

  private static final int TF_OD_API_INPUT_SIZE = 300;
  private static final String TF_OD_API_MODEL_FILE =
      "file:///android_asset/ssd_mobilenet_v1_android_export.pb";
  private static final String TF_OD_API_LABELS_FILE = "file:///android_asset/coco_labels_list.txt";

  // Configuration values for tiny-yolo-voc. Note that the graph is not included with TensorFlow and
  // must be manually placed in the assets/ directory by the user.
  // Graphs and models downloaded from http://pjreddie.com/darknet/yolo/ may be converted e.g. via
  // DarkFlow (https://github.com/thtrieu/darkflow). Sample command:
  // ./flow --model cfg/tiny-yolo-voc.cfg --load bin/tiny-yolo-voc.weights --savepb --verbalise
  private static final String YOLO_MODEL_FILE = "file:///android_asset/graph-tiny-yolo-voc.pb";
  private static final int YOLO_INPUT_SIZE = 416;
  private static final String YOLO_INPUT_NAME = "input";
  private static final String YOLO_OUTPUT_NAMES = "output";
  private static final int YOLO_BLOCK_SIZE = 32;
```

android studio 集成TensorFlow Demo (http://blog.csdn.net/u010302327/article/details/78032852)

```
// Which detection model to use: by default uses Tensorflow Object Detection API frozen
// checkpoints.  Optionally use legacy Multibox (trained using an older version of the API)
// or YOLO.
private enum DetectorMode {
  TF_OD_API, MULTIBOX, YOLO;
}
private static final DetectorMode MODE = DetectorMode.TF_OD_API;
//private static final DetectorMode MODE = DetectorMode.MULTIBOX;
```

0

```
// Minimum detection confidence to track a detection.
private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.1f;
private static final float MINIMUM_CONFIDENCE_MULTIBOX = 0.1f;
private static final float MINIMUM_CONFIDENCE_YOLO = 0.25f;
```

```
private static final boolean MAINTAIN_ASPECT = MODE == DetectorMode.YOLO;

private static final Size DESIRED_PREVIEW_SIZE = new Size(640, 480);

private static final boolean SAVE_PREVIEW_BITMAP = false;
private static final float TEXT_SIZE_DIP = 10;

private Integer sensorOrientation;

private Classifier detector;

private int previewWidth = 0;
private int previewHeight = 0;
private byte[][] yuvBytes;
private int[] rgbBytes = null;
private Bitmap rgbFrameBitmap = null;
```

```
private Bitmap croppedBitmap = null;

private boolean computing = false;

private long timestamp = 0;

private Matrix frameToCropTransform;
private Matrix cropToFrameTransform;
0
private Bitmap cropCopyBitmap;

private MultiBoxTracker tracker;

private byte[] luminance;

private BorderedText borderedText;

private long lastProcessingTimeMs;

@Override
public void onPreviewSizeChosen(final Size size, final int rotation) {
  final float textSizePx =
      TypedValue.applyDimension(
          TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP, getResources().getDisplayMetrics());
  borderedText = new BorderedText(textSizePx);
  borderedText.setTypeface(Typeface.MONOSPACE);

  tracker = new MultiBoxTracker(this);

  int cropSize = TF_OD_API_INPUT_SIZE;
  if (MODE == DetectorMode.YOLO) {
```

```
    detector =
        TensorFlowYoloDetector.create(
            getAssets(),
            YOLO_MODEL_FILE,
            YOLO_INPUT_SIZE,
            YOLO_INPUT_NAME,
            YOLO_OUTPUT_NAMES,
            YOLO_BLOCK_SIZE);
    cropSize = YOLO_INPUT_SIZE;
  } else if (MODE == DetectorMode.MULTIBOX) {
    detector =
        TensorFlowMultiBoxDetector.create(
            getAssets(),
            MB_MODEL_FILE,
            MB_LOCATION_FILE,
            MB_IMAGE_MEAN,
            MB_IMAGE_STD,
            MB_INPUT_NAME,
            MB_OUTPUT_LOCATIONS_NAME,
            MB_OUTPUT_SCORES_NAME);
    cropSize = MB_INPUT_SIZE;
  } else {
    try {
      detector = TensorFlowObjectDetectionAPIModel.create(
          getAssets(), TF_OD_API_MODEL_FILE, TF_OD_API_LABELS_FILE,
TF_OD_API_INPUT_SIZE);
      cropSize = TF_OD_API_INPUT_SIZE;
    } catch (final IOException e) {
      LOGGER.e("Exception initializing classifier!", e);
      Toast toast =
          Toast.makeText(
```

```
            getApplicationContext(), "Classifier could not be initialized", Toast.LENGTH_SHORT);
        toast.show();
        finish();
      }
    }

    previewWidth = size.getWidth();
    previewHeight = size.getHeight();

    final Display display = getWindowManager().getDefaultDisplay();
    final int screenOrientation = display.getRotation();

    LOGGER.i("Sensor orientation: %d, Screen orientation: %d", rotation, screenOrientation);

    sensorOrientation = rotation + screenOrientation;

    LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
    rgbBytes = new int[previewWidth * previewHeight];
    rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight, Config.ARGB_8888);
    croppedBitmap = Bitmap.createBitmap(cropSize, cropSize, Config.ARGB_8888);

    frameToCropTransform =
        ImageUtils.getTransformationMatrix(
            previewWidth, previewHeight,
            cropSize, cropSize,
            sensorOrientation, MAINTAIN_ASPECT);

    cropToFrameTransform = new Matrix();
    frameToCropTransform.invert(cropToFrameTransform);
    yuvBytes = new byte[3][];
```

```
trackingOverlay = (OverlayView) findViewById(R.id.tracking_overlay);
trackingOverlay.addCallback(
    new DrawCallback() {
      @Override
      public void drawCallback(final Canvas canvas) {
        tracker.draw(canvas);
        if (isDebug()) {
          tracker.drawDebug(canvas);
        }
      }
    });
```

```
addCallback(
    new DrawCallback() {
      @Override
      public void drawCallback(final Canvas canvas) {
        if (!isDebug()) {
          return;
        }
        final Bitmap copy = cropCopyBitmap;
        if (copy == null) {
          return;
        }

        final int backgroundColor = Color.argb(100, 255, 125, 2);
        canvas.drawColor(backgroundColor);

        final Matrix matrix = new Matrix();
        final float scaleFactor = 2;
        matrix.postScale(scaleFactor, scaleFactor);
        matrix.postTranslate(
```

```
            canvas.getWidth() - copy.getWidth() * scaleFactor,
            canvas.getHeight() - copy.getHeight() * scaleFactor);
        canvas.drawBitmap(copy, matrix, new Paint());

        final Vector<String> lines = new Vector<String>();
        if (detector != null) {
          final String statString = detector.getStatString();
          final String[] statLines = statString.split("\n");
          for (final String line : statLines) {
            lines.add(line);
          }
        }
        lines.add("************");

        lines.add("Frame: " + previewWidth + "x" + previewHeight);
        lines.add("Crop: " + copy.getWidth() + "x" + copy.getHeight());
        lines.add("View: " + canvas.getWidth() + "x" + canvas.getHeight());
        lines.add("Rotation: " + sensorOrientation);
        lines.add("Inference time: " + lastProcessingTimeMs + "ms");

        borderedText.drawLines(canvas, 10, canvas.getHeight() - 10, lines);
      }
    });
}

OverlayView trackingOverlay;

@Override
public void onImageAvailable(final ImageReader reader) {
  Image image = null;
```

```
++timestamp;
final long currTimestamp = timestamp;

try {
  image = reader.acquireLatestImage();

  if (image == null) {
    return;
  }

  Trace.beginSection("imageAvailable");

  final Plane[] planes = image.getPlanes();
  fillBytes(planes, yuvBytes);

  tracker.onFrame(
      previewWidth,
      previewHeight,
      planes[0].getRowStride(),
      sensorOrientation,
      yuvBytes[0],
      timestamp);
  trackingOverlay.postInvalidate();

  // No mutex needed as this method is not reentrant.
  if (computing) {
    image.close();
    return;
  }
  computing = true;
```

```
    final int yRowStride = planes[0].getRowStride();
    final int uvRowStride = planes[1].getRowStride();
    final int uvPixelStride = planes[1].getPixelStride();
    ImageUtils.convertYUV420ToARGB8888(
        yuvBytes[0],
        yuvBytes[1],
        yuvBytes[2],
        previewWidth,
        previewHeight,
        yRowStride,
        uvRowStride,
        uvPixelStride,
        rgbBytes);

    image.close();
  } catch (final Exception e) {
    if (image != null) {
      image.close();
    }
    LOGGER.e(e, "Exception!");
    Trace.endSection();
    return;
  }

  rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);
  final Canvas canvas = new Canvas(croppedBitmap);
  canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);

  // For examining the actual TF input.
  if (SAVE_PREVIEW_BITMAP) {
    ImageUtils.saveBitmap(croppedBitmap);
```

```
  }

  if (luminance == null) {
    luminance = new byte[yuvBytes[0].length];
  }
  System.arraycopy(yuvBytes[0], 0, luminance, 0, luminance.length);

  runInBackground(
      new Runnable() {
        @Override
        public void run() {

          Toast toast =
          Toast.makeText(
              getApplicationContext(), "run ....", Toast.LENGTH_SHORT);
          toast.show();

          final long startTime = SystemClock.uptimeMillis();
          final List<Classifier.Recognition> results = detector.recognizeImage(croppedBitmap);
          lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

          cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
          final Canvas canvas = new Canvas(cropCopyBitmap);
          final Paint paint = new Paint();
          paint.setColor(Color.RED);
          paint.setStyle(Style.STROKE);
          paint.setStrokeWidth(2.0f);

          float minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
          switch (MODE) {
              case TF_OD_API: minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API; break;
```

```
            case MULTIBOX: minimumConfidence = MINIMUM_CONFIDENCE_MULTIBOX; break;
            case YOLO: minimumConfidence = MINIMUM_CONFIDENCE_YOLO; break;
          }

          final List<Classifier.Recognition> mappedRecognitions =
              new LinkedList<Classifier.Recognition>();
          System.out.println(results.size());
          for (final Classifier.Recognition result : results) {
            RectF location = result.getLocation();
            //location= new RectF(location.top,location.left,location.bottom,location.right);
            location= new RectF(1.0f,1.0f,10.0f,40.0f);

            if (location != null && result.getConfidence() >= minimumConfidence) {
              canvas.drawRect(location, paint);

              cropToFrameTransform.mapRect(location);
              result.setLocation(location);
              mappedRecognitions.add(result);
            }
          }

          tracker.trackResults(mappedRecognitions, luminance, currTimestamp);
          trackingOverlay.postInvalidate();

          requestRender();
          computing = false;
        }
      });

    Trace.endSection();
  }
```

```
protected void processImageRGBbytes(int[] rgbBytes ) {
  //rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);
  //final Canvas canvas = new Canvas(croppedBitmap);
  //canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);
   ++timestamp;
  final long currTimestamp = timestamp;
```

```
Toast toast =
        Toast.makeText(
            getApplicationContext(), "zhaomingming 13718371754", Toast.LENGTH_SHORT);
    toast.show();
    //onImageAvailable();
```

```
rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);
  final Canvas canvas = new Canvas(croppedBitmap);
  canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);

  // For examining the actual TF input.
  if (SAVE_PREVIEW_BITMAP) {
    ImageUtils.saveBitmap(croppedBitmap);
  }

  //if (luminance == null) {
  //  luminance = new byte[yuvBytes[0].length];
  //}
  //System.arraycopy(yuvBytes[0], 0, luminance, 0, luminance.length);

  runInBackground(
      new Runnable() {
        @Override
```

```
public void run() {

  Toast toast =
  Toast.makeText(
     getApplicationContext(), "run ....", Toast.LENGTH_SHORT);
  toast.show();

  final long startTime = SystemClock.uptimeMillis();
  final List<Classifier.Recognition> results = detector.recognizeImage(croppedBitmap);
  lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

  cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
  //final Canvas canvas = new Canvas(cropCopyBitmap);
  final Paint paint = new Paint();
  paint.setColor(Color.RED);
  paint.setStyle(Style.STROKE);
  paint.setStrokeWidth(2.0f);

  float minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
  switch (MODE) {
     case TF_OD_API: minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API; break;
     case MULTIBOX: minimumConfidence = MINIMUM_CONFIDENCE_MULTIBOX; break;
     case YOLO: minimumConfidence = MINIMUM_CONFIDENCE_YOLO; break;
  }

  final List<Classifier.Recognition> mappedRecognitions =
     new LinkedList<Classifier.Recognition>();
  System.out.println(results.size());
  for (final Classifier.Recognition result : results) {
    Toast.makeText(
     getApplicationContext(), "result", Toast.LENGTH_SHORT);
```

```
          toast.show();
            //final RectF location = result.getLocation();
            RectF location_tmp = result.getLocation();
            //RectF location_trans= new RectF(location.top,location.left,location.bottom,location.right);
            //location= new RectF(location.left,location.top,location.right,location.bottom);
            //final RectF location= new RectF(300.0f-location_tmp.top,location_tmp.left,300.0f-
location_tmp.bottom,location_tmp.right);
            //final RectF location_tmp= new
RectF(SizeUtils.dp2px(1.0f),SizeUtils.dp2px(15.0f),SizeUtils.dp2px(100.0f),SizeUtils.dp2px(30.0f));
            float scale=1280.0f/720.0f;
            final RectF location= new RectF(300.0f-location_tmp.top*scale,location_tmp.left/scale,300.0f-
location_tmp.bottom*scale,location_tmp.right/scale);
            //final RectF location= new
RectF(location_tmp.left,location_tmp.top,location_tmp.right,location.bottom);
            canvas.drawRect(location, paint);
            canvas.drawRect(new RectF(1.0f,1.0f,20.0f,20.0f), paint);
            canvas.drawRect(new RectF(21.0f,21.0f,40.0f,40.0f), paint);
            canvas.drawRect(new RectF(41.0f,41.0f,101.0f,101.0f), paint);
            canvas.drawRect(new RectF(101.0f,101.0f,201.0f,201.0f), paint);


            //location = new RectF(10, 10, 100, 100);
            if (location != null && result.getConfidence() >= minimumConfidence) {

              //canvas.drawRect(location, paint);
              canvas.drawRect(location, paint);

              //canvas.drawRect(location, paint);

              cropToFrameTransform.mapRect(location);
              result.setLocation(location);
```

```
            mappedRecognitions.add(result);
          }
        }

        tracker.trackResults(mappedRecognitions, luminance, currTimestamp);
        trackingOverlay.postInvalidate();

        requestRender();
        computing = false;

        if (postInferenceCallback != null) {
          postInferenceCallback.run();
        }
      }
    });


  }

  @Override
  protected int getLayoutId() {
    return R.layout.camera_connection_fragment_tracking;
  }

  @Override
  protected Size getDesiredPreviewFrameSize() {
    return DESIRED_PREVIEW_SIZE;
  }

  @Override
  public void onSetDebug(final boolean debug) {
```

```
    detector.enableStatLogging(debug);
  }
}
```


相关文章推荐

MobileNet 训练检测网实验总结 (http://blog.csdn.net/burning_keyboard/article/details/776…

MobileNet 训练检测网实验总结MobileNet做目标检测的包在tensorflow/models/object_detection里。先下载一个ssd_mobilenet_v1的预训练模型。...

burning_keyboard (http://blog.csdn.net/burning_keyboard) 2017年08月26日 22:24 2041

tensorflow开源Tensorflow Object Detection API安装运行测试 (http://blog.csdn.net/weixin…

TensorFlow对象检测API是一个建立在TensorFlow之上的开源框架，可以轻松构建，训练和部署对象检测模型。 安装： Tensorflow对象检测API依赖于以下项： Protobu...

weixin_35654926 (http://blog.csdn.net/weixin_35654926) 2017年07月19日 22:32 3277

【程序员之路】我是前端工程师，怎么了？

TensorFlow —— 30秒搞定物体检测 (http://blog.csdn.net/wangli0519/article/details/736115…

Google发布了新的TensorFlow物体检测API，包含了预训练模型，一个发布模型的jupyter notebook，一些可用于使用自己数据集对模型进行重新训练的有用脚本。 使用该API可以快速...

wangli0519 (http://blog.csdn.net/wangli0519) 2017年06月22日 22:04 5576

SSD: Single Shot MultiBox Detector in TensorFlow(翻译) (http://blog.csdn.net/jnulzl/artic…

本文基本算是：GitHub:SSD-Tensorflow上的中文版一、环境配置基本环境：Windows 10 + GTX950M1、安装Anaconda3()下载Anaconda3双击安装即可，注意：...

jnulzl (http://blog.csdn.net/jnulzl) 2017年04月01日 19:45 5902

darkflow测试和训练yolo (http://blog.csdn.net/u011961856/article/details/76582669)

参考自github:https://github.com/thtrieu/darkflow darkflow实现了将darknet翻译成tensorflow,可以用tensorflow加载darkn...

u011961856 (http://blog.csdn.net/u011961856) 2017年08月02日 16:19 1614

在windows 7下安装unbuntu16.04 虚拟机进行tensorflow 的编译安装。 (http://blog.csdn.ne…

看了很多相关文章，自己也需要动手试一试，结果发现还是有一些要注意的地方： 我的电脑环境 笔记本：ThinkPad T450 X86_64 系统环境： Windows7 64位系统 ，...

fu_shuwu (http://blog.csdn.net/fu_shuwu) 2017年07月25日 19:38 1023

谷歌开源TensorFlow Object Detection API物体识别系统 (http://blog.csdn.net/zchang81/ar…

近日，谷歌在其开源博客上发表了一篇名为《Supercharge your Computer Vision models with the TensorFlow Object Detection API...

zchang81 (http://blog.csdn.net/zchang81) 2017年06月16日 16:53 4830

真正从零开始，TensorFlow详细安装入门图文教程！ (http://blog.csdn.net/u014696921/artic…

AI这个概念好像突然就火起来了，年初大比分战胜李世石的AlphaGo成功的吸引了大量的关注，但其实看看你的手机上的语音助手，相机上的人脸识别，今日头条上帮你自动筛选出来的新闻，还有各大音乐软件...

u014696921 (http://blog.csdn.net/u014696921) 2016年09月07日 16:01 5793

SSD+caffe｜Single Shot MultiBox Detector 目标检测（一）(http://blog.csdn.net/sinat_26…

作者的思路就是Faster R-CNN+YOLO，利用YOLO的思路和Faster R-CNN的anchor box的思想。 .0 导读（本节来源于BOT大赛嘉宾问答环节 ）SSD 这里的设计就...

sinat_26917383 (http://blog.csdn.net/sinat_26917383) 2017年03月28日 17:13 6273

tensorflow 几个android demo源码环境搭建 (http://blog.csdn.net/cxq234843654/article/det…

tensorflow 几个android demo源码环境搭建说明tensorflow上有几个example，本文重点关注其在android上的应用，源码在git Hub上有，这个apk是什么样子，大家...


cxq234843654 (http://blog.csdn.net/cxq234843654) 2017年04月27日 17:31 4628

tensorflow ssd mobilenet模型训练 (http://blog.csdn.net/u010302327/article/details/78248…

经过了65个小时左右的训练，终于训练出android能跑的模型了 首先下载models 再下载数据集VOC2012 先配置环境，说实话哪些有用哪些没用都没搞清楚 $ pip install p...


0
u010302327 (http://blog.csdn.net/u010302327) 2017年10月16日 12:24 586

TensorFlow学习笔记之源码分析（3）---- retrain.py (http://blog.csdn.net/daydayup_668819…


https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/image_retraining/retrain.py...


daydayup_668819 (http://blog.csdn.net/daydayup_668819) 2017年03月29日 10:21 3195

Tensorflow手写数字识别在android中的实现 (http://blog.csdn.net/wcs__1349/article/details…

说明 下载TensorFlow Android Demo git clone --recurse-submodules https://github.com/tensorflow/tensor...


wcs__1349 (http://blog.csdn.net/wcs__1349) 2017年07月14日 16:25 651

Android TextView属性详解 (http://blog.csdn.net/h183288132/article/details/47206579)

android:autoLink ：设置是否当文本为URL链接/email/电话号码/map时，文本显示为可点击的链接。可选值(none/web /email/phone/map/all) andr...


h183288132 (http://blog.csdn.net/h183288132) 2015年08月02日 13:17 5308

androidstudio检测代码内的中文字符(解决安卓国际化修改字符问题) (http://blog.csdn.net/qq…

androidstudio检测代码内的中文字符在堆完代码后,国际化可谓是咋们的一个头痛问题,,需要一个一个查找,然后一个一个修改
下面介绍一个方法缓解这个问题: 打开全局搜索 在Text to fi...

qq_29967217 (http://blog.csdn.net/qq_29967217) 2017年02月24日 11:25 727