数字系统设计实验二

1. **网络模型结构详解**

选取的应用场景是目标检测，采用YOLOv3网络，该网络是CNN结构。其主要优势有：

1. 检测速度快，YOLO不使用滑窗和分类器，而是同时检测目标的位置和类别；
2. 背景错误少，不容易将不存在物体的某部分背景误认作某个物体；
3. 可以学到物体的泛化特征，若用自然图像训练、检测艺术图像，也可以得到较好的结果。

查看yolov3.cfg文件，可以详细获知网络模型结构。该结构包含6种层：卷积层（含池化层）、shortcut层、route层、upsample层和yolo层。shortcut层的作用是防止过拟合，route层的作用是形成分支。网络结构概括如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| layer | | feature | input | output |
| 0 | conv | C=32, K=3 | 416×416×3 | 416×416×32 |
| 1 | conv | C=64, K=3 | 416×416×32 | 416×416×64 |
| pooling | S=2 | 416×416×64 | 208×208×64 |
| 2 | conv | C=32, K=1 | 208×208×64 | 208×208×32 |
| 3 | conv | C=64, K=3 | 208×208×32 | 208×208×64 |
| 4 | shortcut | from=1 | 208×208×64 | 208×208×64 |
| 5 | conv | C=128, K=3 | 208×208×64 | 208×208×128 |
| pooling | S=2 | 208×208×128 | 104×104×128 |
| 6 | conv | C=64, K=1 | 104×104×128 | 104×104×64 |
| 7 | conv | C=128, K=3 | 104×104×64 | 104×104×128 |
| 8 | shortcut | from=5 | 104×104×128 | 104×104×128 |
| 9 | conv | C=64, K=1 | 104×104×128 | 104×104×64 |
| 10 | conv | C=128, K=3 | 104×104×64 | 104×104×128 |
| 11 | shortcut | from=8 | 104×104×128 | 104×104×128 |
| 12 | conv | C=256, K=3 | 104×104×128 | 104×104×256 |
| pooling | S=2 | 104×104×256 | 52×52×256 |
| 13 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 14 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 15 | shortcut | from=12 | 52×52×256 | 52×52×256 |
| 16 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 17 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 18 | shortcut | from=15 | 52×52×256 | 52×52×256 |
| 19 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 20 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 21 | shortcut | from=18 | 52×52×256 | 52×52×256 |
| 22 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 23 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 24 | shortcut | from=21 | 52×52×256 | 52×52×256 |
| 25 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 26 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 27 | shortcut | from=24 | 52×52×256 | 52×52×256 |
| 28 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 29 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 30 | shortcut | from=27 | 52×52×256 | 52×52×256 |
| 31 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 32 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 33 | shortcut | from=30 | 52×52×256 | 52×52×256 |
| 34 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 35 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 36 | shortcut | from=33 | 52×52×256 | 52×52×256 |
| 37 | conv | C=512, K=3 | 52×52×256 | 52×52×512 |
| pooling | S=2 | 52×52×512 | 26×26×512 |
| 38 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 39 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 40 | shortcut | from=37 | 26×26×512 | 26×26×512 |
| 41 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 42 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 43 | shortcut | from=40 | 26×26×512 | 26×26×512 |
| 44 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 45 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 46 | shortcut | from=43 | 26×26×512 | 26×26×512 |
| 47 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 48 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 49 | shortcut | from=46 | 26×26×512 | 26×26×512 |
| 50 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 51 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 52 | shortcut | from=49 | 26×26×512 | 26×26×512 |
| 53 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 54 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 55 | shortcut | from=52 | 26×26×512 | 26×26×512 |
| 56 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 57 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 58 | shortcut | from=55 | 26×26×512 | 26×26×512 |
| 59 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 60 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 61 | shortcut | from=58 | 26×26×512 | 26×26×512 |
| 62 | conv | C=1024, K=3 | 26×26×512 | 26×26×1024 |
| pooling | S=2 | 26×26×1024 | 13×13×1024 |
| 63 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 64 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 65 | shortcut | from=62 | 13×13×1024 | 13×13×1024 |
| 66 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 67 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 68 | shortcut | from=65 | 13×13×1024 | 13×13×1024 |
| 69 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 70 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 71 | shortcut | from=68 | 13×13×1024 | 13×13×1024 |
| 72 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 73 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 74 | shortcut | from=71 | 13×13×1024 | 13×13×1024 |
| 75 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 76 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 77 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 78 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 79 | conv | C=512, K=1 | 13×13×1024 | 13×13×512 |
| 80 | conv | C=1024, K=3 | 13×13×512 | 13×13×1024 |
| 81 | conv | C=255, K=1 | 13×13×1024 | 13×13×255 |
| 82 | yolo | ~ | | |
| 83 | route | from=79 | 13×13×512 | 13×13×512 |
| 84 | conv | C=256, K=1 | 13×13×512 | 13×13×256 |
| 85 | upsample | a=2 | 13×13×256 | 26×26×256 |
| 86 | route | from=85, 61 | 26×26×(256, 512) | 26×26×768 |
| 87 | conv | C=256, K=1 | 26×26×768 | 26×26×256 |
| 88 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 89 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 90 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 91 | conv | C=256, K=1 | 26×26×512 | 26×26×256 |
| 92 | conv | C=512, K=3 | 26×26×256 | 26×26×512 |
| 93 | conv | C=255, K=1 | 26×26×512 | 26×26×255 |
| 94 | yolo | ~ | | |
| 95 | route | from=91 | 26×26×256 | 26×26×256 |
| 96 | conv | C=128, K=1 | 26×26×256 | 26×26×128 |
| 97 | upsample | a=2 | 26×26×128 | 52×52×128 |
| 98 | route | from=97, 36 | 52×52×(128, 256) | 52×52×384 |
| 99 | conv | C=128, K=1 | 52×52×384 | 52×52×128 |
| 100 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 101 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 102 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 103 | conv | C=128, K=1 | 52×52×256 | 52×52×128 |
| 104 | conv | C=256, K=3 | 52×52×128 | 52×52×256 |
| 105 | conv | C=255, K=1 | 52×52×256 | 52×52×255 |
| 106 | yolo | ~ | | |

1. **网络参数量和计算量分析**

（1）对于卷积层，设卷积核大小为，通道数为，图像宽，高，则：

参数量

计算量

（2）对于全连接层，设输入维度为，输出维度为，则：

参数量

计算量

（3）对于池化层，设池化区域大小为，通道数为，图像宽，高，则：

参数量

计算量

（4）对于shortcut层，它将均相同的两层的对应像素相加，故：

参数量

计算量

（5）对于route层，它将两个相同的层在方向上连接在一起，故：

参数量

计算量

（6）对于upsample层，它通过插值生成加倍的图像，设倍数为，则：

参数量

计算量

（7）对于yolo层，将其视作全连接层，输入维度，输出维度。

根据上述分析，逐一计算出每层网络的参数量和计算量，再相加。详细的计算结果见附件yolov3-Params-FLPOs.xlsx。

计算结果：整个网络的参数量为713.5M，计算量为45.64GFLOPs。

1. **实时推断系统**

本次实验利用<https://github.com/qqwweee/keras-yolo3>网站上提供的YOLO实现代码，通过在Jetson Nano上构建keras环境来运行目标识别模型。

1. 数据采集：利用CSI Camera的接口代码可以实现摄像头图像的获取。
2. 数据处理：采用yolo-tiny轻量级模型，在nano上的处理速度大概可以达到3~4帧/秒。
3. 数据显示：利用Jupyter的ipywidgets小工具将图像显示在PC机上。

实时识别的效果请见压缩包内的视频。

1. **本次实验中遇到的困难**

**朱结奥：**

我在本次实验中遇到了各种各样的困难问题：首先是Jetson Nano上keras 的安装。keras需要非常多的lib支持，比如BLAS、CBLAS等，而Jetson Nano上的sudo apt-get命令并不能很好地下载我需要的包，因此只能按照网络上零零散散的教程下载这些包的源代码进行本地make，编译后再移动到lib里。然而即使困难重重，我仍然通过不懈的努力解决了一个又一个问题，终于将keras安装到了nano上。但是良好的开端只是成功的一半，另一半困难又来了：keras虽然import成功了，但是运行时发生了错误。我仔细考察了错误的原因，结合stackoverflow上的提示发现应该是pillow包的安装错误。于是我先卸载了pillow包，按照教程编译了某个库以后再重新安装pillow，终于解决了这个问题。