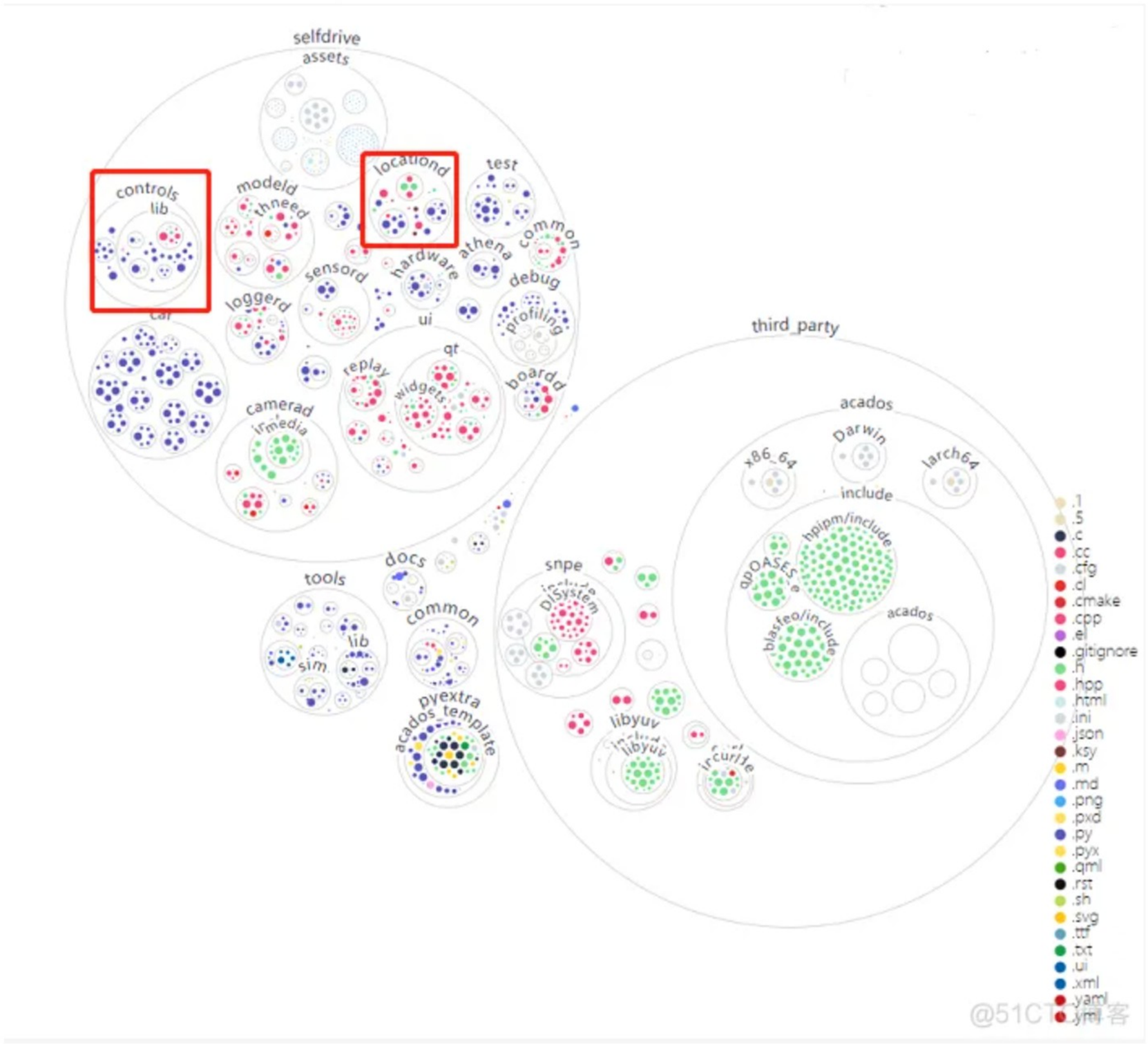
0. openpilot是什么

首先我们需要对openpilot要有个清楚的认知，openpilot主要基于python语言编写。openpilot进程之间通过ZMQ进行通信，使用订阅者和发布者模式，进程订阅其他进程的信息，进程一系列处理，将得到的结果发布出去，让其他进程获取其处理结果。整个openpilot项目可以分为以下几个模块：定位、决策、控制这几个部分。openpilot的实现原理类似于特斯拉靠的是纯视觉的解决方案，但因为camera只有两颗（一颗用于拍摄实现的路况，另一颗用于监控驾驶员），所以openpilot支持也比较有限， 主要支持车道保持 、ACC巡航、自动辅助变道这三个功能。当然如果是需要在汽车上使用起来需要使用EON、Harness这一系列官方的设备对接完成设备的安装与使用，但是我们也需要注意的是，openpilot同样是一个开源项目，这也就导致我们可以轻松地学习里面的一些操作以及算法。即使没有特殊的硬件或汽车，openpilot的所有服务也可以在PC上正常运行。这里官方推荐使用Carla Simulator在模拟中运行openpilot。这使openpilot可以在Ubuntu机器上的运行虚拟数据。

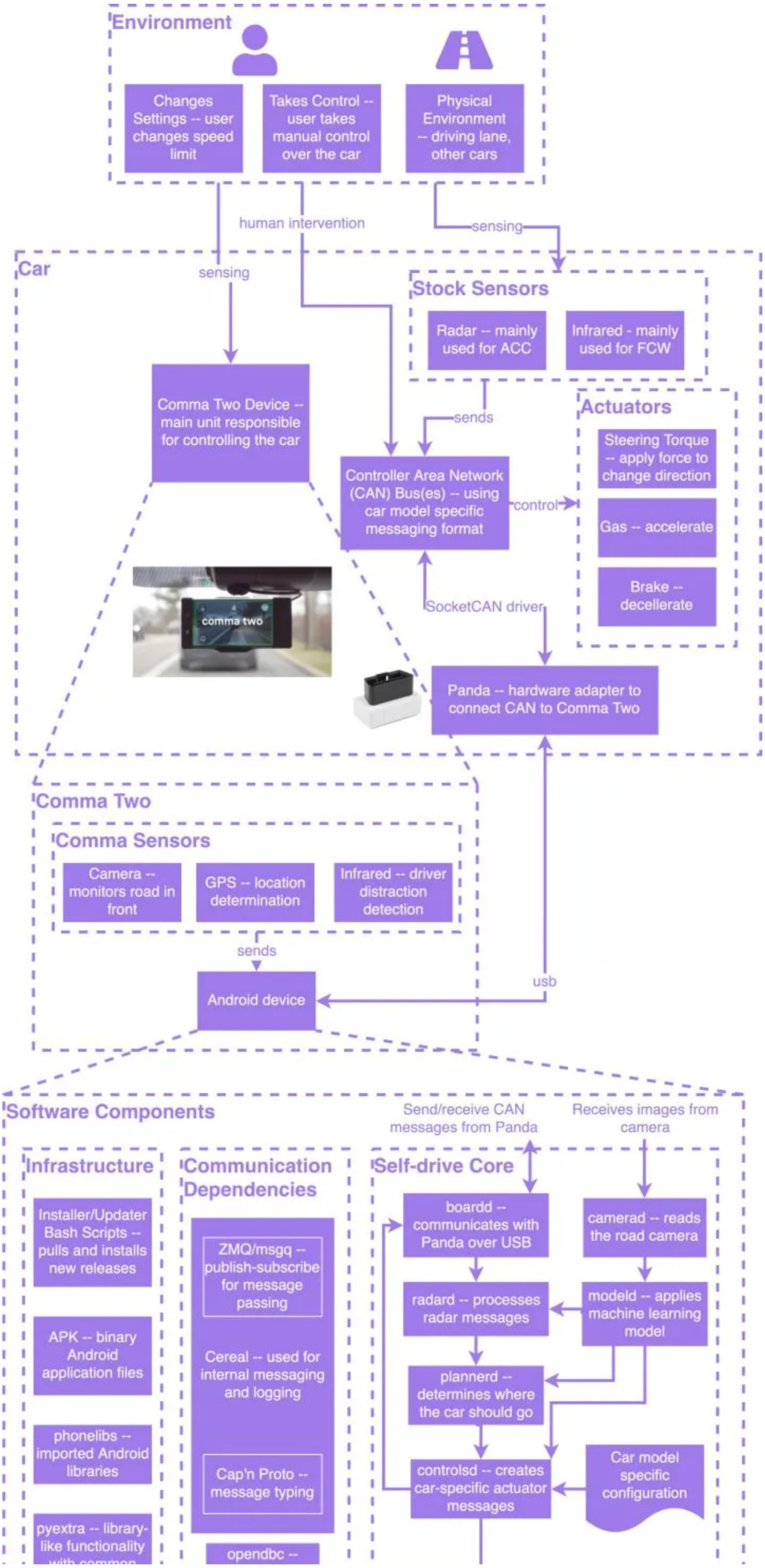
1. 数据结构

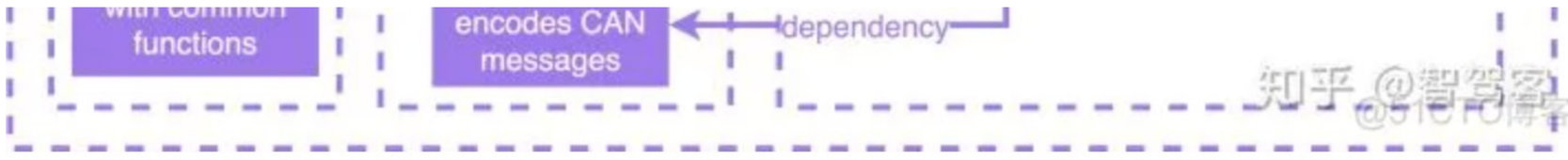
官方的代码还是比较清楚的，我们这里先来研究一下openpilot的整体目录




```
.
├── cereal                # 用于所有日志的消息规范和LIB
├── common                # openpilot中开发的类库功能
├── docs                  # 文档
├── opendbc               # 显示如何解释汽车数据的文件，并以人类可读的方式对理解车辆CAN总线流量所需的信息进行编码。
├── panda                 # CAN通信的代码
├── third_party           # 扩展第三方文件包
├── pyextra               # 扩展第三方python文件包
└── selfdrive             # 驾驶汽车所需的代码
    ├── assets            # 用于用户界面的字体、图像和声音
    ├── athena            # 允许与应用程序app实现通信
    ├── boardd            # 守护进程与面板的通信
    ├── camerad           # 通过驱动程序从摄像头传感器捕获图像
    ├── car               # 用于读取状态和控制执行器的车辆特定代码
    ├── common            # 用于守护程序的共享C/C++代码
    ├── controls          # 规划和控制
    ├── debug             # 帮助您调试和执行汽车端口的工具
    ├── locationd         # 精确定位与车辆参数估计
    ├── logcatd           # Android logcat作为服务
    ├── loggerd           # 汽车数据记录器和上传器
    ├── modeld            # 驾驶和监控模型
    ├── proclogd          # 从proc记录信息
    ├── sensord           # IMU接口代码
    ├── test              # 单元测试、系统测试和汽车模拟器
    └── ui                # 用户界面
```

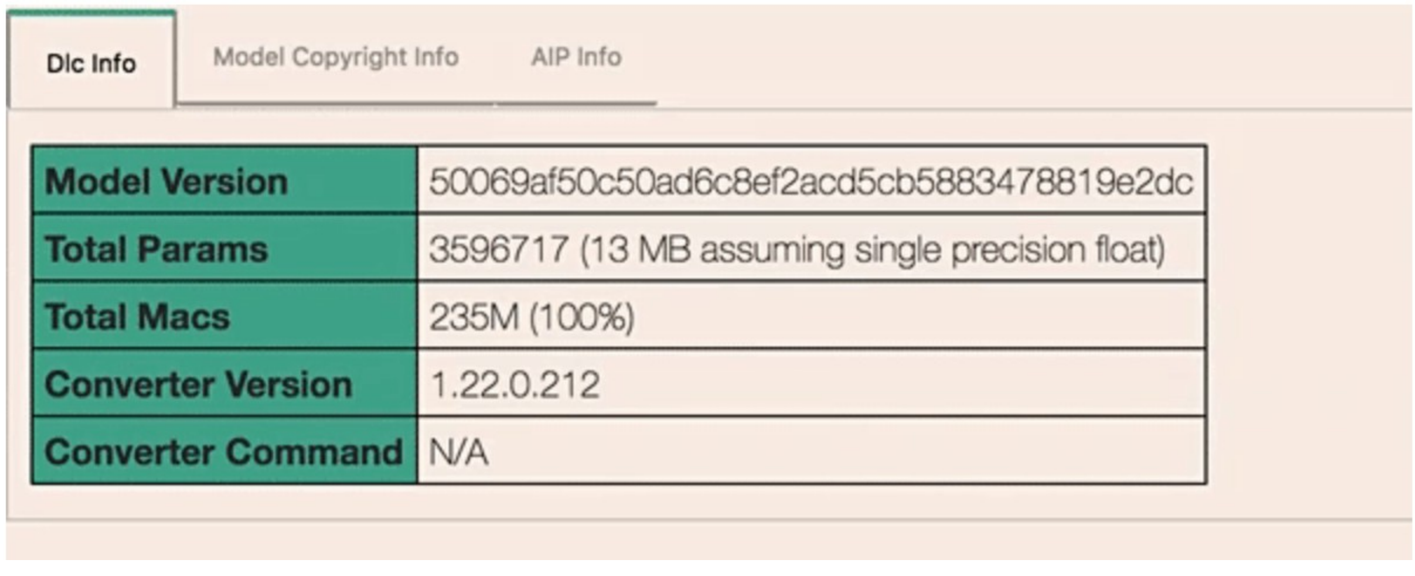
从这里我们可以发现基本上所有的核心操作都在selfdrive内部，当然也包括最核心的定位部分和规控部分。

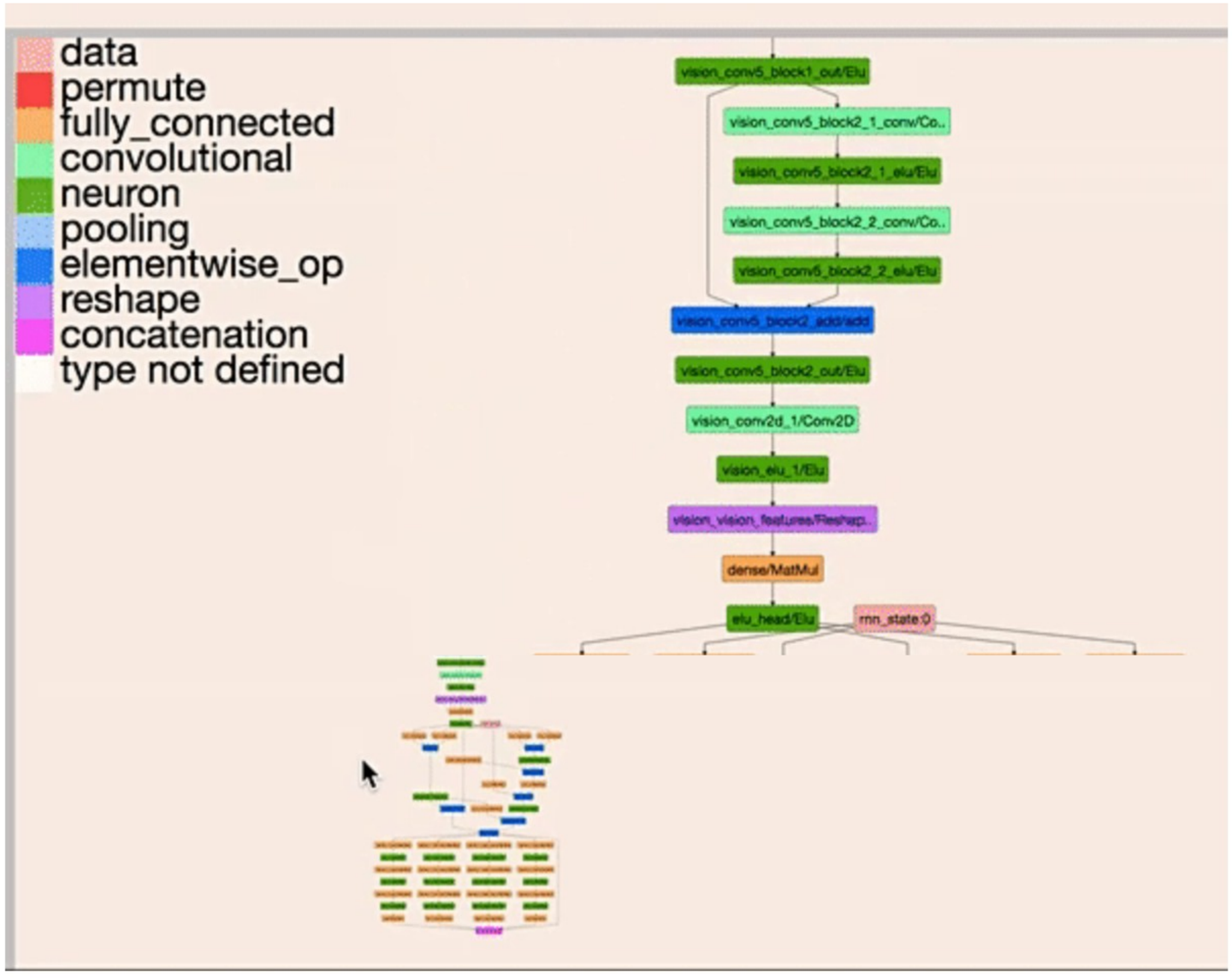

Environment
Changes Settings -- user changes speed limit
Takes Control -- user takes manual control over the car
Physical Environment -- driving lane, other cars
human intervention
sensing
sensing
Car
Stock Sensors
Radar -- mainly used for ACC
Infrared - mainly used for FCW
Comma Two Device -- main unit responsible for controlling the car
sends
Actuators
Steering Torque -- apply force to change direction
Controller Area Network (CAN) Bus(es) -- using car model specific messaging format
control
Gas -- accelerate
Brake -- decellerate
comma two
SocketCAN driver
Panda -- hardware adapter to connect CAN to Comma Two
Comma Two
Comma Sensors
Camera -- monitors road in front
GPS -- location determination
Infrared -- driver distraction detection
sends
usb
Android device
Software Components
Send/receive CAN messages from Panda
Receives images from camera
Infrastructure
Communication Dependencies
Self-drive Core
Installer/Updater Bash Scripts -- pulls and installs new releases
APK -- binary Android application files
phonelibs -- imported Android libraries
pyextra -- library-like functionality
ZMQ/msgq -- publish-subscribe for message passing
Cereal -- used for internal messaging and logging
Cap'n Proto -- message typing
opendbc --
boardd -- communicates with Panda over USB
camerad -- reads the road camera
radard -- processes radar messages
modeld -- applies machine learning model
plannerd -- determines where the car should go
controlsd -- creates car-specific actuator messages
Car model specific configuration



上面的这些是openpilot的目录结构，核心的部分就是controls和locationd两个部分，与上文对应的是下图openpilot总体的架构图。首先作为输入部分深度学习模型位于modeld/models/supercombo.onnx文件中，该模型拥有非常鲁邦的输入输出模型。网络backbone部分，采用了Google团队的 Efficientnet-B2结构。该结构采用复合缩放策略，主要特点是效果好，速度快。卷积部分下采样5次，为了减少参数量，有几个conv是采用group conv，激活函数函数采用Elu

- 输入数据：
  - image stream：以20 Hz频率记录的连续图像帧（256*512*3 RGB）
  - wide image stream：以20 Hz频率记录的连续图像帧（256*512*3 RGB）
  - desire：命令模型发送one-hot encoded向量以执行某些操作：8
  - traffic convention：使用one-hot encoded向量告诉模型是右车道还是左车道：2
  - recurrent state：反馈到GRU中用于时间上下文的循环状态向量：512
- 输出数据：
  - plan：5条预计规划线在33个时间步的预测平均值和标准偏差：4955 = 5 * 2 * 33 * 15(x,y,z位置；x,y,z速度；x,y,z加速度；r,p,y角度；r,p,y角速度)
  - lanelines：4条Laneline（左外、左外、右外和右外）在33个时间步长下的预测平均值和标准偏差: 528 = 4 * 2 * 33 * 2
  - laneline probabilties：4条直线中的每一条都存在的概率：8=4*2
  - road-edges：2条道路边缘（左侧和右侧）在33个时间步长下的预测平均值和标准偏差：264=2 * 2 * 33 * 2
  - leads：潜在2个引导车的假设在0,2,4,6,8,10s时的预测平均值和标准偏差：102=2*(2*64+3)
  - lead probabilities：从现在起，在0、2、4s时刻有一辆领先车的概率：3=1*3
  - desire state：模型认为自己正在执行8个潜在期望动作中的每一个的概率：8
  - meta：关于场景的各种元数据：80=1+35+12+3
  - pose：当前平移和旋转速率的预测平均值和标准偏差：12=2*6
  - recurrent state：反馈到GRU中用于时间上下文的循环状态向量：512




data
permute
fully_connected
convolutional
neuron
pooling
elementwise_op
reshape
concatenation
type not defined
vision_conv5_block1_out/Elu
vision_conv5_block2_1_conv/Co..
vision_conv5_block2_1_elu/Elu
vision_conv5_block2_2_conv/Co..
vision_conv5_block2_2_elu/Elu
vision_conv5_block2_add/add
vision_conv5_block2_out/Elu
vision_conv2d_1/Conv2D
vision_elu_1/Elu
vision_vision_features/Resha..
dense/MatMul
elu_head/Elu
rnn_state:0