岗马克视觉科技

landmark

AI+ 边缘计算视觉赋能行业智能化升级

关于我们 About us

上海岚马克视觉科技有限公司成立于2018年9月，是一家以**边缘AI视觉感知技术**为核心，集软/硬件/系统平台研发能力为一体的人工智能初创型**高新技术企业**

让**绿色低碳的AI视觉感知**真正落地各行各业，从**端到端**赋能产业智能化、自动化升级改造，是公司肩负的使命

公司产品主要应用于技术安防、智慧城市、数字多媒体、材料化工等行业领域

公司总部位于上海黄浦，欧洲研发分部位于法国La Rochelle

www.landmarkvision.cn 上海市黄浦区雁荡路6号320室 (021) 64086212

我们的优势
Our strengths

Academy

公司依托**东南大学人工智能/大数据/智能嵌入式**团队，优化迭代最新技术、快速更新产品性能。目前已与中国科学院、山西大学、上海交通大学等高校、科研院所开展产学研合作。

Industry

国家电网公司 STATE GRID CORPORATION OF CHINA

CHINA TOWER 中国铁塔

上海电气 SHANGHAI ELECTRIC

公司得到**国家电网、中国铁塔、上海电气**等大型国家企事业单位的大力支持，分别有相关试点、示范性产业智慧升级改造项目签约落地。

Partners

HIKVISION

NVIDIA

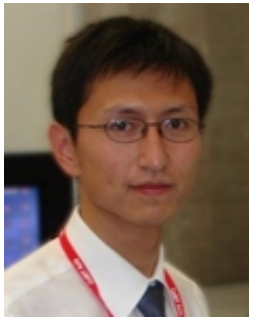
公司与**海康威视、英伟达订立战略合作协议。**双方双方将充分发挥各自优势，在边缘计算、端到端计算机视觉感知应用领域开展紧密合作。

我们的团队
Our team

龚　　昊 博士
Hao GONG, Dr.

Co-founder
法国Grenoble University
信号图像处理分析博士
2项欧盟科研项目
1项国家863计划项目
核心成员/负责人
ADAS/自动驾驶专家
计算机视觉专家

明　祖衡 博士
Zuheng MING, Dr..



CTO
法国La Rochelle University
信息图像交互博士后
多项法国/欧盟科研项目
核心成员/负责人

计算机视觉/语音识别算法专家

陆 凯捷 先生
Kaijie LU, M.B.A.

V.P. of Sales & Marketing
上海交通大学
工学学士/MBA
挪威船级社（中国）有限公司
高级项目经理（2017）
苏州嘉志股权投资管理有限公司
创始合伙人/投资总监 （2018）

从GPU到边缘计算设备

AI项目开发的一些思考

① 如何缩短项目开发周期，快速应对不同客户的多样需求，完成PoC演示，推进项目落地?

② 针对不同的实际场景，如何对算法硬件平台进行选型?

③ 如何综合现有技术手段，解决CV以及深度学习算法无法解决的盲区问题，为客户提供最优的解决方案?

④ 如何充分挖掘边缘计算设备的算力，降低单路视频分析的成本?

⑤ 如何形成数据的闭环，在项目部署后便捷地采集数据，不断迭代优化模型，提高客户的满意度?

AI项目的一般开发交付流程

1. 数据采集：现场数据采集、数据标注、数据集校验；
2. 模型训练：设计模型、训练模型；
3. 模型部署：模型转化、模型量化、模型裁剪、模型微调；
4. 业务开发：根据项目的需求，设计业务规则，完成相应业务逻辑处理代码的编写调试；
5. 项目部署：制作安装包或者docker镜像，安装部署在目标平台上；
6. 模型优化：根据现场的应用采集数据，优化模型

流程中的难点

数据

没有实际场景的数据/数据很少

已有模型达不到理想效果->
客户不给试用机会->
更不用说数据采集优化模型

需求

客户很配合，数据不是问题

客户需求很多，要求很高

现实

一堆客户不愿买单的PoC项目

买家秀/卖家秀

面对纷繁复杂的应用场景，**客户自己也可能不太清楚的不确定需求**

流程中的难点

暴露垃圾	提供基于视频流的暴露垃圾检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
道路不洁	提供基于视频流的道路不洁检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
非法小广告	提供基于视频流的非法小广告检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
违规户外广告	提供基于视频流的违规户外广告检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
积存垃圾渣土	提供基于视频流的积存垃圾渣土检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
沿街晾挂	提供基于视频流的沿街晾挂检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
私搭乱建	提供基于视频流的私搭乱建检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
渣土车未密闭运输	提供基于视频流的渣土车未密闭运输检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
公用设施损坏（井盖类）	提供基于视频流的公用设施损坏（井盖类）检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
垃圾满溢	提供基于视频流的垃圾满溢检测与预警，支持划定检测区域，支持以红色方框动态标注提醒，支持生成报警信息
绿地踩踏	提供基于视频流的绿地踩踏检测与预警，支持划定检测区域，绿地附近人员/机非警戒，进入并停留一段时间报警，支持以红色方框动态标注提醒，支持生成报警信息

徘徊检测	行人在指定区域徘徊检测与识别，支持设置徘徊次数
入侵检测	行人或车辆在指定区域入侵检测与识别，支持设置入侵区域
攀爬检测	行人攀爬行为检测与识别，支持设置攀爬识别区域
拨电话识别	提供基于视频流的拨电话识别与预警，支持以红色方框动态标注提醒，支持生成报警信息
吸烟检测	行人吸烟行为检测与识别
警戒拌线	自动检测人或车辆穿越指定警戒线的行为，支持设置警戒类型、拌线类型
穿越围栏	当目标按照设定的方向穿越设置的围栏线时，能够进行识别
物品搬移	检测区中的原有目标被拿走超过一定时间时，识别场景，生成报警信息
物品遗留	当检测区中的遗留目标超过设置的时间时，识别场景，生成报警信息
人脸识别	识别视频流中的人脸，并进行人脸比对
烟火识别	提供基于视频流的烟火检测与预警，支持以红色方框动态标注提醒，支持生成报警信息

AI公司的生存之道

1. 必须**解决规模复制效益**的问题；
2. 集中力量在某个**方向/赛道**上打造**有技术/市场壁垒的产品**；
3. 以产品思维开展项目，将项目开发变为**搭积木**；
4. 只有**可复制**的产品，才能分摊高昂的研发费用

PoC项目简介

PoC（Proof of Concept），即**概念验证**。通常是企业进行产品选型时或开展外部实施项目前，进行的一种产品或供应商能力验证工作。

1. 验证内容：
   a) 产品的功能；
   b) 产品的性能；
   c) 产品的API试用性；
   d) 产品相关技术文档的规范性、完整性；
   e) 产品API开放性；
   f) 企业资质规模及实施案例

2. PoC准备前提：
   a) 前期调研充分，并已经对产品或供应商有了较深入的沟通了解；
   b) 企业对自己的产品需求比较清晰

3. PoC测试工作参与者：

使用用户代表、业务负责人、项目负责人、技术架构师、测试工程师、商务经理等

PoC项目简介

4. PoC测试工作准备文档：
   a) PoC测试工作说明文档
   b) 功能测试用例
   c) 场景测试用例
   d) 技术测评方案
   e) 商务测评方案

5. PoC测试工作：
   a) 工作启动
   b) 产品宣讲及现场集中测试
   c) 技术测评
   d) 间歇性测试工作
   e) 商务验证
   f) 背书归档、分析总结

AI产品常见的三种交付形态

1) 服务 Service

AI模型的部署服务，客户端可以通过HTTP/REST API或gRPC的方式来请求服务；

2) 开发包SDK或者功能组件

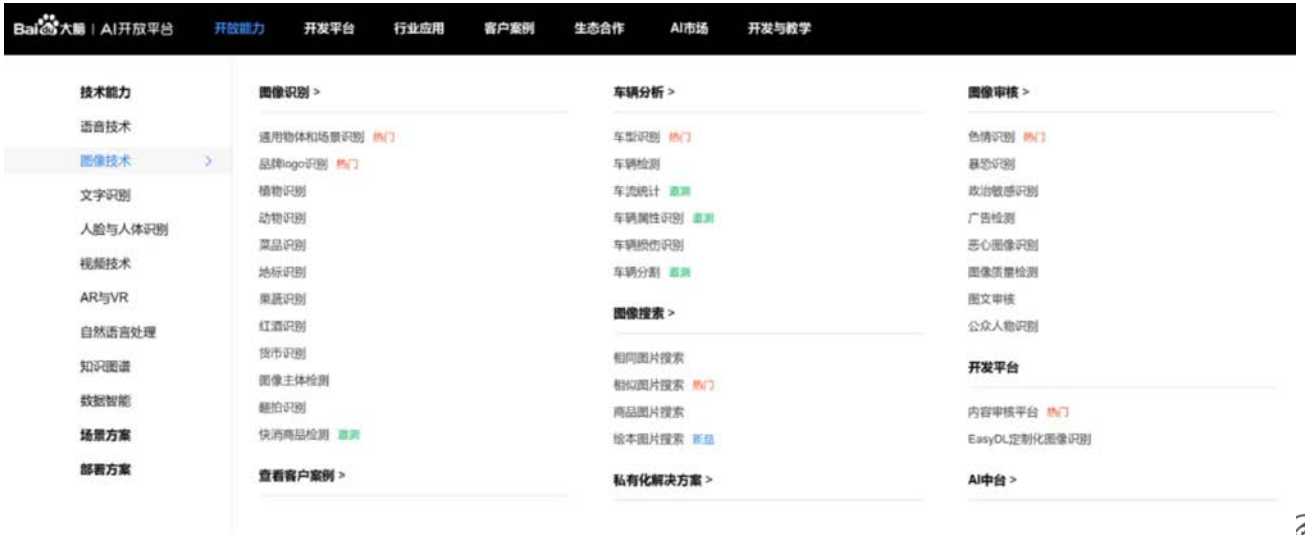
给非AI中间商或集成商的业务系统赋能AI算法能力；

3) 应用 Application

面向某个场景的最终用户。一整套包括交互界面在内的软件系统，有时也会将硬件一起捆绑交付

服务 Service

AI模型的部署服务，客户端可以通过HTTP/REST或gRPC的方式来请求服务。输入一张图片/一段视频，输出图片/视频的分析结果，通常按次数收费或者按时间段授权。比如百度AI市场上提供的各种API服务：



服务 Service

这种形式，业务流程相对是单一的，**主要需要考虑的是充分利用GPU算力资源，能够提供稳定的高吞吐量的服务。**这种服务通常部署在GPU服务器上，可能是客户局域网内的服务器，也可能是公有云上的服务器。市面上也有一些成熟的商用框架可以使用，如NVIDIA的Triton Inference Server, Google的TF Servering，百度的Paddle Serving等。Triton是 NVIDIA 推出的 Inference Server，专门做 AI 模型的部署服务。

而客户端可以通过HTTP/REST或gRPC的方式来请求服务，特性包括以下方面：

a) 支持多种框架，例如 Tensorflow、TensoRT、Pytorch、ONNX甚至自定义框架后端；
b) 支持 GPU 和 CPU 方式运行，能最大化利用硬件资源；
c) 容器化部署，集成 k8s，可以方便的进行编排和扩展；
d) 支持并发模型，支持多种模型或同一模型的不同实例在同一GPU上运行；
e) 支持多种批处理算法，可以提高推理吞吐量

开发包SDK/功能组件

有的中间商或集成商以及一些传统的非AI公司，需要**用深度学习解决问题的能力。**把基于深度学习的算法能力，集成到自己的业务系统中，为最终用户提供服务。这时，他们会寻找第三方的合作伙伴，**提供一套封装了深度学习算法能力的SDK或者功能组件。**向他们的业务系统赋能AI算法能力，比如百度的EasyDL-零门槛AI开发平台，云从科技的人脸识别服务等。

百度EasyDL开发平台的功能示意图

应用Application

这种形式的产品，通常面向的是**某个场景的最终用户**。因此交付的产品，是一整套包括交互界面在内的软件系统，有时也会将硬件一起捆绑交付。对这类产品，用户需要的其实只是**应用的分析输出结果**。比如绘制了违规提醒框的实时画面，web、邮件甚至手机短信联动的告警消息，某个时段或者满足某种条件的数据分析报表等。

这类产品一方面需要提供友好的操作界面供用户查看使用，同时可能还需要提供对接用户第三方平台的接口，将分析产生的告警结果等信息，推送到用户的业务管理平台。

应用类产品的基本框架

以**视频分析**为例，通常包括：

1. **视频结构化引擎**：通过分析视频内容，生成包含了目标坐标、类别、属性、特征、追踪id等信息的结构化数据，供业务中台做进一步的业务逻辑处理；
2. **业务中台**：主要对业务进行逻辑处理，通常由业务逻辑和对外接口构成；
3. **管理平台**：视频结构化引擎或者业务平台可能是分布式的，部署在同一局域网内的不同主机、甚至是不同局域网内的不同主机上，管理平台用来管理这些主机

视频结构化引擎

为了简化开发流程，提高代码复用率，降低代码维护难度，视频结构化引擎作为一个基础平台，应当适配不同硬件平台，屏蔽硬件差异，向管理平台提供统一的接口，同时支持根据不同需求灵活配置任务流程。主要分为三个部分：

① **视频源接入：**支持多种接入，图片，视频文件，rtsp流，GB28181流，海康SDK（工业相机）

② **流程Pipeline配置和创建：**

a) 输入数据预处理：对输入数据做尺寸缩放、归一化等；

b) 模型推理：使用多种硬件平台进行推理，如NVIDIA GPU、Jetson、Bitmain、Cambricon等

c) 输出数据后处理：对模型的结果，进行后处理，得到可以显示的结果；

③ **结果输出：**终端打印、写入Redis、输出画面到屏幕、保存结果到视频文件、推送rtsp流等。

管理平台

① 主机管理
② 视频源管理
③ 任务管理
④ 用户交互界面与结果查看
⑤ license认证与管理

为什么选择边缘计算设备？

边缘计算的优势：

① **低延迟：**计算能力部署在设备侧附近，设备请求实时响应；

② **低带宽运行：**将工作迁移至更接近于用户，或是数据采集终端的能力，能够降低站点带宽限制所带来的影响；

③ **隐私保护：**数据本地采集，本地分析，本地处理，有效减少了数据暴露在公共网络的机会，保护了数据隐私

GPU服务器的问题：

① 许多场景下，数据源（摄像头）是分布式的，可能分布在不同的子网内，甚至分布在不同的城市，使用GPU服务器集中处理延时大、带宽占用高、能耗高（因为传输的数据中大部分是无效信息）；

② 有的场景下使用GPU，会造成算力过程、资源浪费；

③ 相比较纯软件的产品，客户更倾向于为软硬件一体的产品买单

一些例子

边缘计算部署在设备侧附近，**可以通过算法即时反馈决策**，并可以过滤绝大部分的数据，有效降低云端的负荷，使得海量连接和海量数据处理成为可能

- 同一地点/内网上百路摄像头->x86服务器
- 不同地点/节点，每个节点摄像头10-20路以下->边缘计算设备

边缘计算设备的特点

算力有限：常常在几T～几十T INT8 OPS之间；

功耗低：常边缘计算设备的功耗在5-30W，可以通过太阳能供电，进行户外移动作业；

硬件接口丰富：便于与其他设备/系统对接；

体积小/重量轻：安装简便灵活，便于分布式部署和扩展

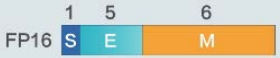
AI算力计算方式

FP16

名词释义:

FP16（半精度浮点数）是一种被计算机使用的二进制浮点数据类型。FP 即“Floating Point”，表示“浮点运算”，16 表示“半精度浮点数”。FP16 在计算机存储器中占用 2 个字节（16 bits），利用“浮点”（浮动小数点）的方法，可以表示一个范围很大的数值。

与FP32 （单精度浮点数） 相比，FP16的访存的消耗仅为其1/2，数据通信时间大幅减少，适合端侧AI运算。与INT8相比，FP16在高精度图像处理上有较大优势，可以使图片中更多颜色、对比度、质感和清晰度得以保留。
麒麟芯片支持的AI精彩瞬间、AI夜景降噪以及短视频网站常用的人像分割换背景、AI超分、AI-RAW等都是基于FP16机器学习模型计算的。



INT8

名词释义:

INT8是一种定点计算方式，代表了整数运算，一般是由浮点运算量化而来。二进制中的一个“0”或一个“1”叫1位（bit），INT8 指用8位（即8个bit ）表示一个数字。

与FP32和FP16相比，INT8精度相对较低，更多用于深度学习中的智慧识物、图库分类、人脸检测等分类问题。INT8的优势在于计算的数据量相对小，计算速度可以更快，并且能通过减少计算和内存带宽需求来提高能效。

FP16、INT8没有完全优劣之分，移动端的深度学习常常会需要根据不同场景、不同需求来综合考虑选择。

注：FP16和INT8是移动端AI计算中深度学习模型的常用数据格式，可以在损失少量精度的情况下，大幅降低算力要求。

边缘计算设备算力估算

表格中的前3项是NVIDIA的GPU，其峰值算力为根据CUDA核心数、主频，折算为FP16而计算的理论值，估算公式为：**计算能力的峰值 = 单核单周期计算次数 × 处理核个数 × 主频**

后三项为NVIDIA的边缘计算模组Jetson的不同产品，峰值算力为产品手册中给出的参考值。这些峰值算力虽然并不完全准确，但也基本代表了设备的算力情况。

不准确的原因主要有两方面：

① 这些是理论值，实际中还要考虑线程调度、数据拷贝、异构同步等，实际算力肯定达不到理论值；

② 除了CUDA核心，设备内还会有其他加速单元，比如Tensor Core、DLA（深度学习加速器）。

以Jetson AGX Xavier为例，他还有48个Tensor Core，以及DLA，(2x) NVDLA Engines* | 5 TFLOPS (FP16)，相当于额外的5TFLOPS算力。因此官方手册中会写AGX Xavier的AI算力是32TOPS INT8（16TOPS FP16）

	显卡	算力	架构	CUDA核心数	主频	典型功耗	峰值算力FP16
GPU	GTX 1080	6.1	pascal	2560	1607MHz	180W	16TFLOPS
	RTX 2080	7.5	turing	2944	1515MHz	225W	17TFLPOS
	RTX 3080	8.6	ampere	8704	1440MHz	320W	50TFLOPS
Jetson	AGX Xavier	7.2	Volta	512	/	10-30W	5.5-11TFLOPS
	NX	7.2	Volta	384	/	10-15W	6TFLOPS
	Nano	5.3	Maxwell	128	/	5-10W	0.5TFLOPS

几种边缘计算设备平台

训练平台：通常以英伟达Nvidia-GPU为主；

推理平台：云端和设备端，CPU(x86 arm)，GPU，NPU，TPU，FPGA，ASIC

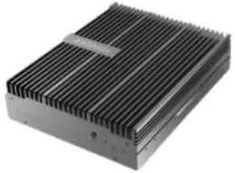
公司	类别	产品系列	芯片/模组型号	加速库/框架/转换工具	算力
NVIDIA英伟达	云	GeForce	RTX2080	CUDA/cuDNN/TensorRT	
	端	Jetson	Nano/TX2/Xavier NX/AGX Xavier	CUDA/cuDNN/TensorRT	0.5/1.3/6/5.5~11TFLOPs(FP16)
HUAWEI华为	云	Atlas	昇腾910AI处理器	MindSpore	521TOPS(INT8), 256TFLOPS(FP16)
	端	Atlas	昇腾310AI处理器	MindSpore	22/16/8 TOPS INT8
Amlogic	端	Verisilicon 芯原 Vivante NPU IP	C308	OpenVX OpenCL Acuity toolkit Acuity ovxlib	2TOPS(INT8)
	端		A311D		5TOPS(INT8)
Bitmain比特大陆	云	AI计算加速卡SC5+	BM1684*3	BMNNSDK BMLang编程	105.6T INT8（Winograd Enable），以及6.6T FP32
	端	算丰AI计算盒SE5	BM1684		17.6TOPS(INT8) 2.2TFLOPS(FP32)
Yitu依图科技	云	原子系列云端服务器	Questcore求索*6	QuestCore工具集 自定义算子需分割网络，仅支持CPU实现	
	端	前沿系列边缘盒子	Questcore求索		16TOPS(INT8)
Cambicon寒武纪	云	思元270	思元270-S4 思元270-F4	Neuware SDK CNML\CNPlugin\CNRT	128TOPS(INT8)、支持INT4、INT16，支持浮点混合运算
	端	思元220 T59边缘模组	思元220 M.2加速卡		8TOPS(INT8) 16TOPS
Xilinx赛灵思		Versal AI Core			

几种常见的边缘计算设备

Nvidia Jetson

华为 Atlas

Amlogic



Bitmain Sophon SE5

Yitu TBox

Cambricon MLU220

NVIDIA Jetson 硬件参数

NVIDIA Jetson 硬件参数

	Jetson Nano (B01)	Jetson TX2	Jetson Xavier NX	Jetson AGX Xavier
CPU	Quad-core Arm A57 @ 1.43 GHz	Hexa-core processor with 2x NVIDIA Denver 2 64-Bit CPU, 4x ARM Cortex-A57 cores	6-core NVIDIA Carmel ARM v8.2 64-bit CPU 6 MB L2 + 4 MB L3 cache	8-Core ARM v8.2 64-Bit CPU 8 MB L2 + 4 MB L3
GPU	128-core Maxwell GPU	256-core Pascal GPU	Volta GPU with 384 NVIDIA CUDA cores and 48 Tensor cores	512-core Volta GPU with Tensor Cores
AI加速	N/A	N/A	2x NVDLA Engines 7-Way VLIW Vision Processor	2x NVDLA Engines 7-Way VLIW Vision Processor
Memory	4 GB 64-bit LPDDR4 @ 25.6 GB/s	8GB 128-bit LPDDR4 @59.7 GB/s	8 GB 128-bit LPDDR4x @ 51.2GB/s	32 GB 256-Bit LPDDR4x @ 137 GB/s
存储	MicroSD slot	32GB eMMC flash SD card slot SATA	MicroSD card slot M.2 Key M for SSD	32 GB eMMC flash Micro SD/UFS card socket M.2 Key M for SSD eSATAp + USB 3.0 type A
视频输出	HDMI 2.0 and eDP 1.4 (video only)	HDMI 2.0 Display expansion header with 2x 4-lane DSI, eDP/DP/HDMI, backlight and touch	HDMI 2.0 DisplayPort	HDMI 2.0 2x DisplayPort via USB-C ports
摄像头 I/F	2x MIPI CSI-2	Camera expansion header with 6x 2-lane CSI or 3x 4-lane CSI, I2S, UART, I2C 5MP camera included in kit	2x MIPI CSI-2	Camera expansion header with 16x 2-lane CSI or 8x 4-lane CSI, conrol signals
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)	2x 4K @ 60 |4x 4K @ 30 | 7x 1080p @ 60 | 14x 1080p @ 30 (H.265)	2x 4Kp60 | 4x 4Kp30 | 12x 1080p60 | 32x 1080p30 (H.265) 2x 4Kp30 | 6x 1080p60 | 16x 1080p30 (H.264)	2x 4Kp60 (H.265)
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)	1x 4K @ 60 | 3x 4K @ 30 | 4x 1080p @ 60 | 8x 1080p @ 30 (H.265)	2x 4Kp30 | 6x 1080p60 | 14x 1080p30 (H.265 & H.264)	2x 4Kp60 (H.265)
网络	Gigabit Ethernet	Gigabit Ethernet 802.11ac WiFi 5 and Bluetooth	Gigabit Ethernet WiFi and Bluetooth via included M.2 Key-E card	Gigabit Ethernet
USB	4x USB 3.0 1x Micro USB 2.0 device	1x USB 3.0 1x Micro USB 2.0	4x USB 3.1 1x Micro USB 2.0	2x USB 3.1 Type C with optional DP, optional PD
Expansion	M.2 Key E socket (PCIe x1, USB 2.0, UART, I2S, and I2C) 40-pin expansion header with GPIO, I2C, I2S, SPI, UART signals	M.2 Key E socket PCIe x4 connector 40-pin expansion header with I2C, I2S, SPI, UART, D-MIC 30-pin expansion header with I2S, GPIOs, speaker	M.2 Key E socket (PCIe x1, USB 2.0, UART, I2S, and I2C) 40-pin expansion header with GPIOs, I2C, I2S, SPI, UART	M.2 Key E socket with PCIe x1, USB 2.0 + UART for WiFI/LTE PCIe x16 slot (x8 used) 40-pin expansion header with UART, SPI, CAN, I2C, I2S, DMIC, GPIOs HD audio header
电源	5V/4A via power barrel 5V/2A via micro USB optional PoE support	5.5V to 19.6V via power barrel 19V DC adapter included	9 to 19V via power barrel power adapter included	9 to 20V via power barrel 19V DC adapter included
功耗	5-10W	7.5-15W	10-15W	10-30W
散热	Heatsink	Fansink	Fansink	Fansink
尺寸大小	100 x 80 x 26 mm	170x170mm (Mini-ITX)	103 x 90.5 x 31 mm	105 x 105 x ~55 mm
AI性能	0.5 TFLOPS (FP16)	1.3 TFLOPS (FP16)	6 TFLOPS (FP16) 21 TOPS (INT8)	5.5-11 TFLOPS (FP16) 20-32 TOPS (INT8)
价格	700	3500	3500	5400

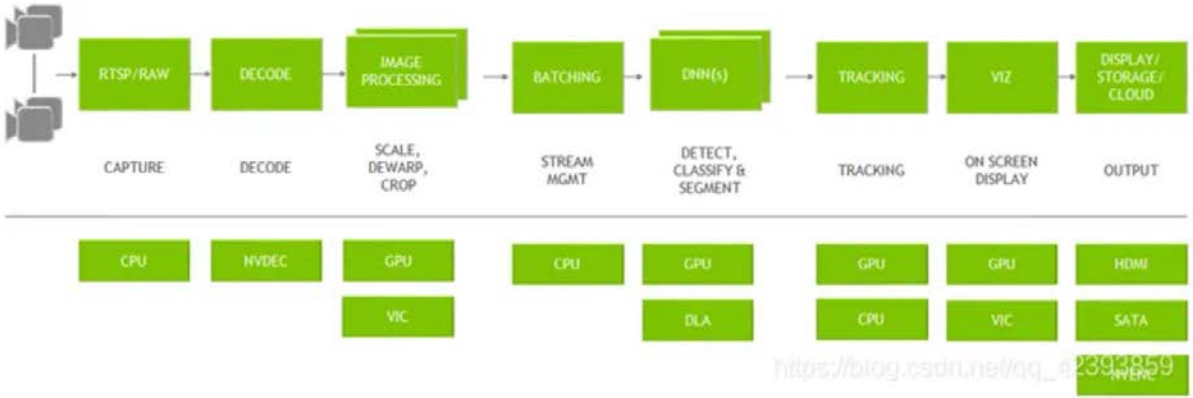
NVIDIA Jetson 开发工具链

```
NVIDIA Jetson Xavier NX (Developer Kit Version) - Jetpack 4.6 [L4T 32.6.1]

- Up Time:          0 days 0:38:52                          Version: 3.1.1
- Jetpack:          4.6 [L4T 32.6.1]                         Author: Raffaello Bonghi
- Board:                                                     e-mail: raffaello@rnext.it
   * Type:            Xavier NX (Developer Kit Version)
   * SOC Family:      tegra194      ID: 25
   * Module:          P3668         Board: P3509-000
   * Code Name:       jakku
   * Cuda ARCH:       7.2
   * Serial Number:   1422420036416
- Libraries:                                          - Hostname:     nx-desktop
   * CUDA:            10.2.300                        - Interfaces:
   * OpenCV:          4.1.1  compiled CUDA: NO          * eth0:       192.167.200.188
   * TensorRT:        8.0.1.6                           * docker0:    172.17.0.1
   * VPI:             ii libnvvpi1 1.1.12 arm64 NVIDIA Vision Programming Interface library
   * VisionWorks:     1.6.0.501
   * Vulkan:          1.2.70
   * cuDNN:           8.2.1.32
```

NVIDIA Jetson--Deepstream

DeepStream是Nvidia针对智能视频分析（IVA）应用提供的端到端解决方案。图像获取、解码、预处理、检测、追踪、分类、分析无缝连接。如果再配合Transfer Learning Toolkit （TLT），则连模型的训练、剪枝都一起解决了。DeepStream基于GStreamer搭建，其中各种功能都被实现为插件。推理部分基于TensorRT来完成，其它各个模块也大多在GPU上运行，并减少了数据传输开销。同时用户也可以开发所需的自定义插件。



华为 atlas 硬件参数

Atlas是华为基于昇腾系列AI处理器和业界主流异构计算部件，打造的智能计算平台。通过模块、板卡、小站、AI服务器等丰富的产品形态，打造面向“端、边、云”的全场景AI基础设施方案，可广泛用于“平安城市、智慧交通、智慧医疗、AI推理”等领域。

产品	Atlas 200 DK 开发者套件 型号：3000
AI芯片	昇腾310
AI算力	22 TOPS INT8 16 TOPS INT8 8 TOPS INT8
内存规格	LPDDR4X，8 GB，总带宽51.2 GB/s
编解码能力	• 支持H.264 硬件解码，16路1080P 30 FPS（2路3840*2160 60 FPS） • 支持H.265 硬件解码，16路1080P 30 FPS（2路3840*2160 60 FPS） • 支持H.264 硬件编码，1路1080P 30 FPS • 支持H.265 硬件编码，1路1080P 30 FPS • JPEG解码能力1080P 256 FPS，编码能力1080P 64 FPS，最大分辨率：8192*4320 • PNG解码能力1080P 24 FPS，最大分辨率：4096*2160
接口	• 网络：1个GE RJ45 • USB：1个USB2.0 / USB3.0 • Camera：2个15 pin raspberry pi 相机连接器 • 其他：1个40 pin IO连接器
电源	5~28 V DC，默认配置12 V / 3 A适配器
功耗	典型功耗20W
工作环境温度	0℃ ~ 45℃
结构尺寸	137.8 mm * 93.0 mm * 32.9 mm

产品	Atlas 300 推理卡 型号：3000
形态	半高半长PCIe卡
AI芯片	昇腾310
AI算力	88 TOPS INT8
内存规格	LPDDR4X 32 GB，总带宽204.8 GB/s
编解码能力	• 支持H.264硬件解码，64路1080P 30 FPS（8路 3840*2160 60 FPS） • 支持H.265硬件解码，64路1080P 30 FPS（8路 3840*2160 60 FPS） • 支持H.264硬件编码，4路1080P 30 FPS • 支持H.265硬件编码，4路1080P 30 FPS • JPEG解码能力4*1080P 256 FPS，编码能力 4*1080P 64 FPS，最大分辨率：8192*4320 • PNG解码能力4*1080P 48 FPS，最大分辨率： 4096*2160
PCIe	PCIe x16 Gen3.0
功耗	最大67W
结构尺寸	169.5 mm * 68.9 mm
工作环境温度	0℃~55℃（32°F ~ +131°F）

Atlas 200 DK 开发者套件（型号：3000）

是一款高性能AI应用开发板，集成了昇腾310 AI处理器，方便用户快速开发、快速验证，可广泛应用于开发者方案验证、高校教育、科学研究等场景。

Atlas 200 AI加速模块（型号：3000）

集成了昇腾310 AI处理器，可以在端侧实现图像识别、图像分类等，广泛用于智能摄像机、机器人、无人机等端侧AI场景。

Atlas 500 智能小站（型号：3000）

是面向边缘应用的产品，具有超强计算性能、体积小、环境适应性强、易于维护和支持云边协同等特点，可以在边缘环境广泛部署，满足在安防、交通、社区、园区、商场、超市等复杂环境区域的应用需求。

Atlas 500 Pro 智能边缘服务器（型号：3000）

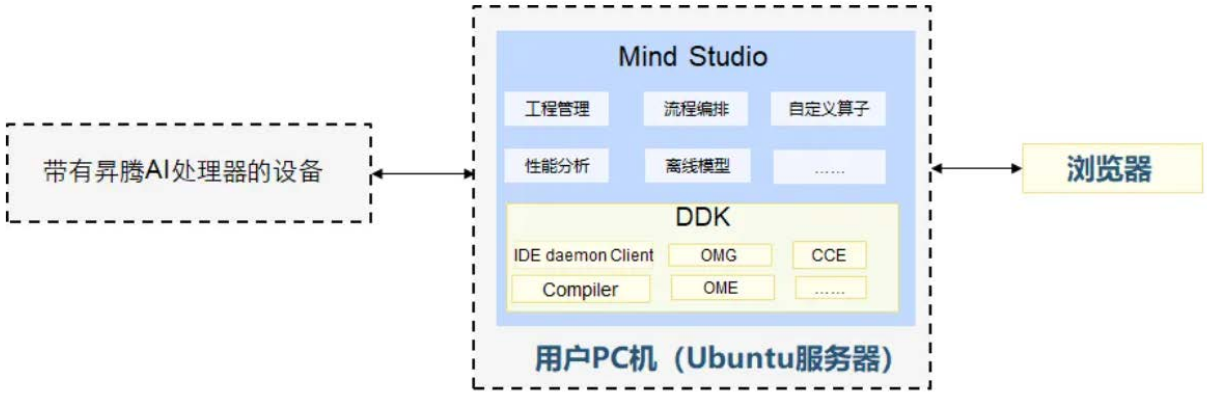
是面向边缘应用的产品，具有超强计算性能、高环境适应性、易于部署维护和支持云边协同等特点。

华为 atlas 开发工具链

华为atlas的开发工具链主要有两部分：

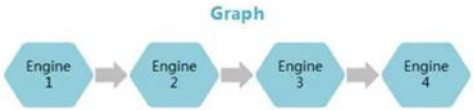
（1）Mind Studio：这是基于昇腾AI处理器的开发工具链平台，提供了基于芯片的算子开发、调试、调优以及第三方算子的开发功能。

（2）DDK（Device Development Kit）：设备开发工具包，为开发者提供基于昇腾AI处理器的相关算法开发工具包，旨在帮助开发者进行快速、高效的人工智能算法开发。



华为 atlas 开发工具链

华为提供了Matrix框架来完成推理业务迁移，有点类似gstreamer的味道。把每个功能节点抽象成流程的基本单元Engine，每个Engine对应着一个独立的线程。在Graph配置文件中配置Engine节点间的串接和节点属性（运行该节点所需的参数）。节点间数据的实际流向根据具体业务在节点中实现，通过向业务的开始节点输入数据激活Graph。每个Graph是一个独立的进程。



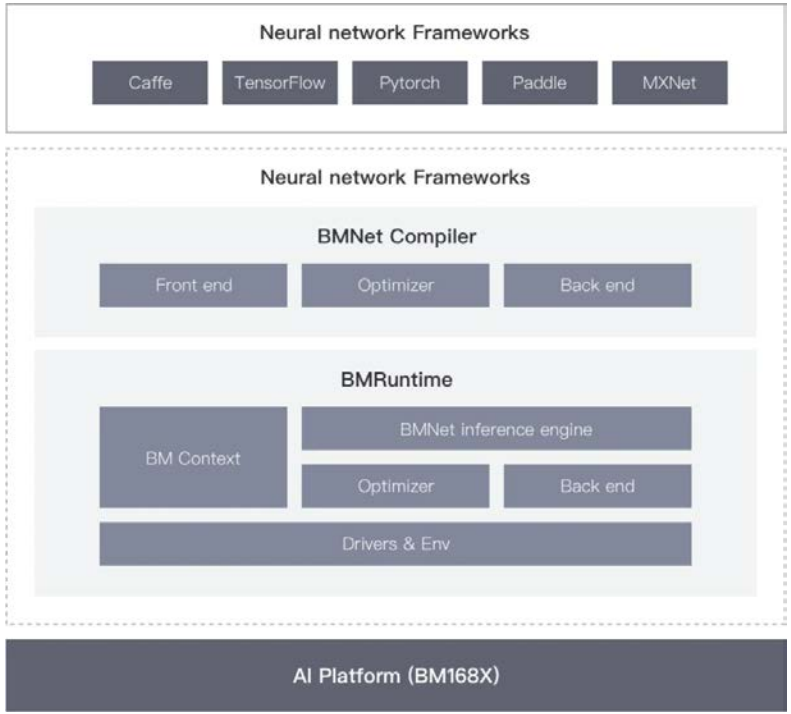
Mind Studio也提供了可视化的界面用于配置Graph：

比特大陆 Sophon SE5 硬件参数

- TPU　BM1684　峰值算力：17.6 TOPS (INT8) / 2.2 TFLOPS (FP32)
- CPU　8核 ARM A53 2.3GHz
- 视频解码　960FPS 1080p（38路1080P@25FPS）
- 视频编码　50FPS 1080p（2路1080P@25FPS）
- 内存　12GB
- 网口　10/100/1000Mbps自适应 *2
- 其他接口　USB3.1 *2 / MicroSD *1 / HDMI *1 / RS-232 *1 / RS-485 *1 / I/O *4
- 典型功耗　≤20W
- 尺寸　188mm*148mm*44.5mm（长*宽*高）
- 工作温度　-20℃ ~ +60℃（具体视配置而定）
- 供电　直流12V/5A适配器
- 可选功能　SATA硬盘 / LTE无线回传

比特大陆 Sophon SE5 开发工具链

比特大陆提供了BMNNSDK（SOPHON Neural Network SDK）一站式工具包，提供底层驱动环境、编译器、推理部署工具等一系列软件工具，涵盖了神经网络推理阶段所需的模型优化、高效运行时支持等能力，由**BMNet Compiler**和**BMRuntime**两部分组成。



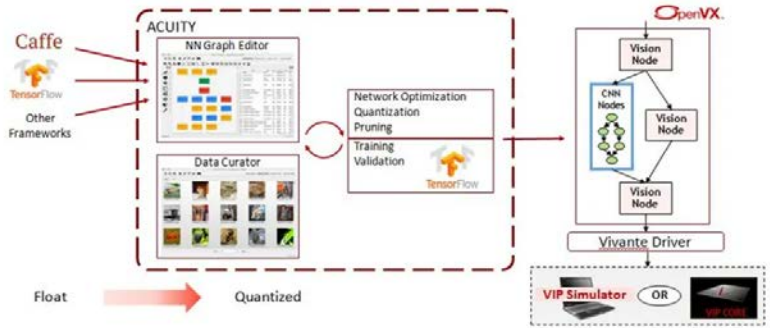
Amlogic NeuBoard

Amlogic，**晶晨半导体**是全球无晶圆半导体系统设计的领导者，为智能机顶盒、智能电视、智能家居等多个产品领域，提供**多媒体SoC芯片**和**系统级解决方案。**

NeuBoard智能分析盒

项目	组件	规格
硬件配置	CPU	AmlogicA311D，四核ARM Cortex A73 (2.2 GHz)+两核ARM Cortex A53 (1.8 GHz)
	NPU	独立AI神经处理单元，5 TOPS算力
	GPU	ARM Mali-G52 MP4
	VPU	4KH.265、VP9和AVS2视频解码，支持1080pH.265/H.264视频编码
	内存	2GB / 4GB DDR4
	存储	4GB / 8GB / 16GB / 32GB eMMC5.1，支持标准256G Micro SD/SDHC/SDXC卡扩展存储
	操作系统	NeucoreNeuSYS
	接口	100/1000M自适应RJ45×1，USB2.0×2，HDMI2.1×1
	指示灯	电源状态LED灯×1,启动状态LED灯×1
	电源	DC±12V，配220V交流电源适配器，设备功率≤5W

- Versilicon提供Acuity Toolset 用于模型转换(量化)以及推理代码生成



- Versilicon提供ovxlib用于自定义层的实现
- Neucore提供基于gstreamer的NeuSDK开发示例框架

寒武纪 MLU

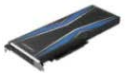
中科寒武纪其实是较早布局深度学习处理器的企业之一，也是目前国际上少数几家全面系统，掌握了通用型智能芯片及其基础系统软件研发和产品化核心技术的企业之一。

寒武纪的产品线也比较丰富，但由于它本身是专注做芯片、模组与基础软件的，并不提供成品的边缘计算设备或服务器，如果需要，可以从其合作的下游厂商处购买。

智能加速卡	智能加速系统	智能边缘计算模组	终端智能处理器IP	软件开发平台
思元290 MLU290-M5智能加速卡 思元270系列 MLU270-S4智能加速卡 MLU270-F4智能加速卡	玄思1000智能加速器	思元220系列 MLU220-SOM智能模组 MLU220-M.2边缘端智能加速卡	Cambricon-1M Cambricon-1H	Cambricon NeuWare®

寒武纪 MLU 硬件参数

智能加速卡-第一代架构



产品型号	思元100-C	思元100-D
核心频率	1GHz（平衡模式）	
半精度浮点运算速度（FP16）	16 TFLOPS（关闭稀疏模式时理论峰值性能） 64 TFLOPS（打开稀疏模式时理论峰值性能）	
整数运算速度（INT8）	32 TOPS （关闭稀疏模式时理论峰值性能） 128 TOPS（打开稀疏模式时理论峰值性能）	
内存容量	8GB/16GB	
内存位宽	256-bit	
内存带宽	102.4GB/s	
系统接口	x16 PCIe Gen.3	
外形	全高全长，单槽位	半高半长，单槽位
是否支持解码	支持解码	不支持解码
TDP功耗	110w	75w
ECC保护	是	
散热方式	被动散热	

智能加速卡-第二代架构

思元270-S4 产品规格		
产品性能	芯片型号 INT8理论峰值/TOPS INT4理论峰值/TOPS INT16理论峰值/TOPS	思元270（MLUv02 架构） 128 256 64
计算精度支持	低精度、混合精度	INT16，INT8，INT4，FP32，FP16
内存规格	内存容量 内存位宽 内存带宽	16GB DDR4，ECC 256-bit 102 GB/s
接口	PCIe接口	x16 PCIe Gen.3
功耗	最大热设计功耗 散热设计	70w 被动散热
形态	半高半长，单槽位	
尺寸	167.5mm x 68.9mm	
重量	310g	

智能加速卡-第二代架构

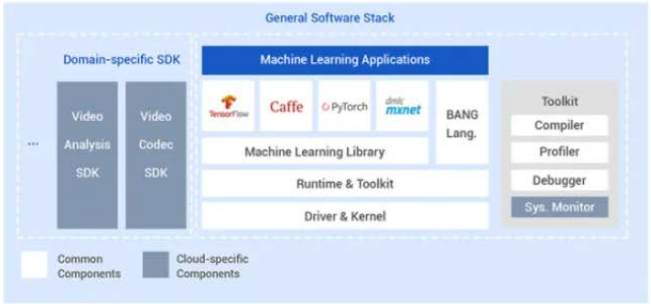
思元270-F4 产品规格		
产品性能	芯片型号 INT8理论峰值/TOPS INT4理论峰值/TOPS INT16理论峰值/TOPS	思元270（MLUv02 架构） 128 256 64
计算精度支持	低精度、混合精度	INT16，INT8，INT4，FP32，FP16
内存规格	内存容量 内存位宽 内存带宽	16GB DDR4，ECC 256-bit 102GB/s
接口	PCIe接口	x16 PCIe Gen.3
功耗	最大热设计功耗（TDP） 最大整板功耗（TBP） 散热设计	150w 160w
形态	全高，全长，双槽位	
尺寸	267mm x 111.15mm	
重量	874g	

边缘计算模组-第二代架构

思元220-M.2 规格参数	
参数	规格
型号	MLU220-M.2
内存	LPDDR4x 64bit，4GB
理论峰值性能	8TOPS（INT8）
编解码性能	支持H.264 ，HEVC (H.265) ，VP8 ，VP9
图片解码	JPEG解码 ，最大图片分辨率8192 x 8192
接口规格	M.2 2280 ，B+M key （PCIe3.0 X2）
功耗	8.25W
结构尺寸	长80mm ，宽22mm, 高7.3mm(无散热)/21.3mm（带散热）
散热	被动散热
表面温度	-20~80 度

寒武纪 MLU 人工智能开发平台

寒武纪人工智能开发平台（Cambricon NeuWare®）是寒武纪专门针对其云、边、端的智能处理器产品打造的软件开发平台， Neuware采用端云一体的架构，可同时支持寒武纪云、边、端的全系列产品。下图是寒武纪SDK的架构图及常用的工具包：



寒武纪提供全套易用的开发调试调优工具

- 开发 软件开发工具包
TensorFlow/Caffe/MXNet/PyTorch/AndroidNN
CNML（机器学习编程库）
CNRT（高性能运行时库）
CNCC（编译器）

- 调试 功能调试工具包
CNGDB（调试工具）
CNQual（硬件诊断工具）

- 调优 性能调优工具
CNPerf （性能剖析）
CNMon（系统监控）

边缘计算设备的选型

① **性能：**arm核心数量和主频、内存与AI模组专用内存、深度学习推理能力、视频编解码能力、jpeg编解码以及其他视觉运算硬件加速能力；

② **价格：**在性能都能满足要求的情况下，价格自然成为选型的决定因素。特别是同一个AI模组，其实会有多家厂商竞品可供选择。比如浪潮的EIS200和凌华的DLAP-211-JNX都是基于NVIDIA的Jetson NX模组；

③ **工具链：**支持主流框架模型情况，算子及网络模型支持情况，接口易用程度，SDK、技术论坛完备程度和技术支持力度；

④ **外围接口：**板载的外围接口是否能满足应用场景需求；

⑤ **功耗：**有的客户可能因为作业环境的限制对功耗要求比较严格，比如是在野外使用太阳能供电，但通常较低的功耗也意味着较低的算力。

边缘计算设备比较

	EIS200	Altas 500	Sophon SE5	NeuBoard	NE-V-200
实体图					
设备厂商	浪潮	华为	比特大陆	锘科	格兰泰克
AI模组厂商	NVIDIA	华为	比特大陆	Amlogic	寒武纪
AI模组	Jetson NX	Atlas 200	BM1684	A311D	MLU220 M.2
AI INT8算力	21TOPS	22T/16TOPS	17.6TOPS	5TOPS	8TOPS
CPU内存	8GB	4GB	4GB	2GB/4GB	4GB
AI内存		8GB	8GB		4GB
存储空间	16GB 可扩展	扩展硬盘	32GB 可扩展	4/8/16/32GB SD卡	32GB
OS	ubuntu18.04	ubuntu18.04	debian 9	定制linux	debian 10
CPU	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6MB L2 + 4MB L3	海思Hi3559A	8核 A53@2.3GHz	四核ARM Cortex A73 (2.2 GHz)+两核ARM Cortex A53 (1.8 GHz)	双核Cortex-A72+四核Cortex-A53大小核CPU、主频1.8GHz
视频解码	2x 4K @ 60 (HEVC) 12x 1080p @ 60 (HEVC) 32x 1080p @ 30 (HEVC)	16路1080P 30 FPS （2路3840*2160 60 FPS	960fps 1080p（38路1080P@25FPS）	4路1080P视频流	16路H264/H265 1080@30帧解码
视频编码	2x 4K @ 30 (HEVC) 6x 1080p @ 60 (HEVC)	1路1080P 30 FPS	50fps 1080p（2路1080P@25FPS）		
工具链体验	五颗星	四颗星	四颗星	三颗星	三颗星
技术支持力度	四颗星	四颗星	五颗星	三颗星	三颗星

岚马克边缘计算设备LM-EC101

1） LM-EC101（如下图）为一款基于NVIDIA® Jetson NX™系列模块设计的计算平台，内置集成NX模块，预装Ubuntu 18.04操作系统，具备21TOPS浮点运算的AI处理能力，采用风扇散热，具备优秀的散热能力，尺寸轻巧外观新颖，丰富IO接口类型，预留便于现场安装的底部支架，具备超长MTBF稳定运行能力，广泛适用于安防、物流、金融、零售、司法等行业，是边缘端部署AI算力进行深度学习的理想载体。

LM-EC101边缘计算平台概述

- 内嵌NVIDIA® Jetson NX™
- 支持M.2 KEY M（PCIex4 NVMe 2280）
- 支持CSI 摄像头
- 支持多种接口(如USB/以太网/SPI/串口/GPIO 等)
- 支持双频WiFi/4G 模组
- 支持20Pin I/O 扩展口
- 风扇主动散热设计
- 内置Ubuntu 18.04 系统和Jetpack SDKs

岚马克边缘计算设备LM-EC201

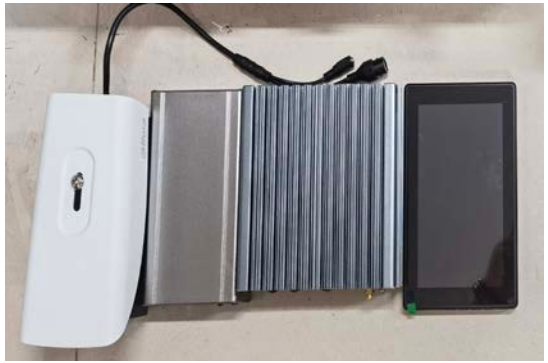
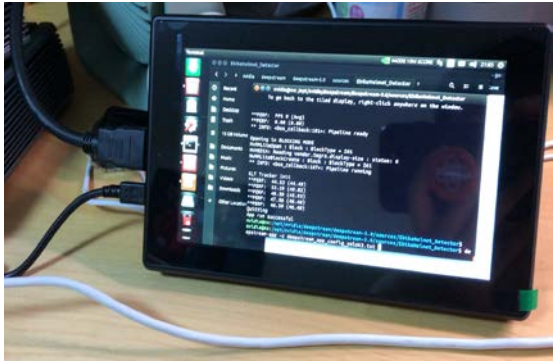
2） LM-EC201（如下图）为一款基于NVIDIA® Jetson AGX Xavier系列模块设计的计算平台，内置集成AGV Xavier模块，预装Ubuntu 18.04 操作系统，具备32TOPS浮点运算的AI处理能力，支持电源适配器和PoE网口两种电源输入方式，超强固轻型铝合金材料设计，无风扇结构传导被动散热，尺寸轻巧外观新颖，丰富IO接口类型，预留便于现场安装的底部支架，具备超长MTBF稳定运行能力，可应用于机器人、无人配送车、低空防御、智能巡检、智慧楼宇等自主化机器，是边缘端部署AI算力进行深度学习的理想载体。产品通过欧盟的CE/FCC 认证和CCC认证。

LM-EC201边缘计算平台概述

- 内嵌NVIDIA® Jetson AGX Xavier
- 支持M.2 KEY E (PCIex1 2230)或者M.2 KEY M (PCIex4 NVMe 2280)
- 支持多种接口(如CAN/USB/POE 以太网/SPI/串口/I2C/GPIO等)
- 支持双频WiFi/4G 模组
- 支持4xPoEPSE 供电端1000BASE-T RJ45 端口和1xPoEPD 受电端1000BASE-T RJ45端口
- 无风扇被动散热设计
- 内置Ubuntu 18.04 系统和Jetpack SDKs
- 宽压 12-36V 凤凰端子输入，LNA1 端口支持 PoE 输入供电。

基于岚马克边缘计算设备LM-EC201非机动车驾驶行为分析演示系统



边缘计算设备开发与GPU服务器开发比较

① **CPU架构不同：**GPU服务器是x86架构，GPU插在主板的PCIE插槽内；而边缘计算设备是基于aarch64的整体设备，其中有ARM CPU和GPU以及NPU、TPU、VPU等；

② **资源有限：**边缘设备由于资源有限，底层要使用C/C++推理，程序需要充分优化，以压榨硬件资源性能；

③ **交叉编译：**C/C++代码直接在边缘计算设备上编译比较耗时，有的甚至不支持在设备中编译，通常通过交叉编译的方式，在宿主机上生成代码，再拷贝到边缘计算设备中执行；

④ **远程调试：**由于需要使用gdb server远程调试，VSCode很好用；

⑤ **软件安装：**边缘计算设备通常运行的是裁剪/定制的linux，debian/ubuntu可以使用apt在线安装库包，但有的边缘计算设备内的linux是精简版的，没有包管理工具，安装软件只能源码交叉编译

边缘计算设备的一般开发流程

（1）基础平台开发： 深度学习分析引擎、业务中台、管理平台；

（2）模型转换、验证及优化： 使用硬件平台厂商提供的模型转换工具套件将caffe、tensorflow、pytorch、mxnet、darknet、onnx等模型转换为目标平台模型，必要时进行模型量化以及模型finetune；对不支持的模型或层，自定义算子、插件实现

（3）视频结构化引擎代码适配： 主要是视频流及图片编解码、推理等模块，任务管理、流程控制、前后处理等其他代码通常都是跨平台的；

（4）交叉编译及测试： 使用交叉编译工具链编译及调试代码，交叉编译工具工具链主要包括2部分内容，linaro gcc g++编译及调试器和包含了目标平台系统环境及软件库的所有代码文件；

（5）业务代码实现： 针对不同场景的业务需求开发业务逻辑处理代码；

（6）系统部署： 通常使用docker部署，使用docker-compose编排多个docker容器或使用K8S管理多个分布式节点。

常见问题

（1）模型转换失败：

解决方案：

① onnx模型转换失败，可能是**onnx和opset的版本不支持**，可以更换版本尝试；

② onnx模型转换失败，也可能是转换工具对onnx某些层**支持不好**，可以先使用onnx-simplifier简化模型，优化其中不必要的容易引起问题的层；

③ 如果是pytorch模型转换失败，需要注意pytorch有两种类型的保存格式，一种是只有权重的，一种是带有模型结构和权重的JIT模型；转换工具基本都要求JIT模型，应当使用torch.jit.trace保存。

④ 使用工具链提供的编程语言自定义算子实现不支持的层；

⑤ 将问题反馈给硬件厂商，询问是否有新版本的SDK或等待问题解决；

⑥ 反馈给算法同事，修改模型结构，尝试使用其他支持的等价算子，重新训练模型；

常见问题

（2）模型推理结果不对：

解决方案：

① 检查前后处理（包括输入、输出层的scale因子）；

② 检查模型转换后输出tensor的顺序；

③ 使用工具链中提供的工具保存中间层结果，逐步排查解决。

（3）模型量化精度损失：

解决方案：

① 量化是一定会有精度损失的，这个无法避免；

② 数量更多和内容更均衡的量化集，可以在一定程序改善量化模型的精度；

③ 如果仍无法满足要求，重新训练量化后的模型（不是所有的平台都支持）

常见问题

（4）程序优化：

解决方案：

① 首先，检查程序最耗时的部分是在哪里，找出制约性能的瓶颈：**视频解码？任务队列？数据拷贝？还是算力资源不够，模型需要进一步裁剪优化？**

② 然后，针对具体问题优化程序：使用更加高效的计算库或者硬件加速接口、优化多线程多进程、改进数据结构、使用多Batch推理或者根据任务实际设置合理的处理帧率等。总之，优化的主要原则就是**减少不必要的数据拷贝、充分利用计算单元资源。**

③ 通常，观察AIPU（GPU/NPU/TPU）的利用率情况，如果一直比较平稳，说明计算资源得到了充分的利用；如果起伏比较大，甚至有突然的高峰和低谷，说明某些时刻AIPU在等待数据；

④ 此外，某些AIPU可能对某种尺寸的数据、某些操作或特定参数的神经网络算子做了专门优化，在设计模型时应优先选用**高效的结构和参数。** 比如有的AI加速芯片的，若卷积层的输入不是8的倍数，底层会额外进行多次padding操作；stride为3的卷积核比其他卷积核要更高效；输入尺寸是512的倍数时的计算效率 > 256的倍数时的计算效率 > 128的倍数时的计算效率等。

互动问答1

Q：能简单说说交叉编译吗？典型场景是啥？

A：由于C/C++代码是依赖于硬件平台的二进制代码，源码需要经过编译器编译、链接，最终生成可执行的二进制代码。当我们在一个架构的平台上，编译生成在另一个架构的平台上运行的代码的过程，就叫交叉编译。交叉的意思就是编译源码的平台与代码运行的目标平台不同，比如我们要在x86的机器上编译生成在边缘计算盒子aarch64上的代码，就需要交叉编译。

互动问答2

Q：分享中从性能角度比较了几款边缘小站，从性价比角度，你觉得哪个更好？或者这么说，如果让你们公司挑选一个小站，把自有算法适配进去，以软硬一体的标准品卖出去，你会挑选哪个盒子？

A：关于盒子的选择，首选还是**推荐Nvidia的Jetson系列**。无论从价格还是生态来说，对开发者都是最友好的，同时代码移植成本也最低。这几款边缘计算设备，**华为官方的atlas 500小站**价格比较高，要1.2-1.6W；**amlogic的算力**相对比较低，要便宜一些；其他的几种差别不大，都在6000-8000，当然如果供货量大，价格应该可以商量。此外，**NVIDIA的Jetson**盒子成品虽然价格比较高，但是官方出的模组并不贵，比如NX是3500，AGX是5000多，对于初学者和个人开发者比较友好。总的来说 **，推荐按照英伟达、比特大陆、华为、寒武纪、Amlogic**的顺序考察选定。

互动问答3

Q：在x86上的gnu工具是编译不出aarch64的目标代码的，需要用到对应aarch64的编译器版本吧？这个就叫工具链?

A：编译器是工具链中很重要的一部分，但是工具链中还有一部分就是目标平台的系统环境，其中包含了程序依赖的运行库。这些依赖库分为2部分：（1）一部分是属于linux系统的基本库，这部分库通常都跟编译器集成在一起；（2）另一部分是特殊的库，比如我交叉编译在比特大陆边缘计算设备里的深度学习推理程序，还需要比特大陆的推理运行库，这部分库比特大陆会单独提供，包含在其提供的sdk BMNNSDK中。所以这些应该是一个整体，编译器+系统/依赖库。

互动问答4

Q：在x86上的gnu工具是编译不出aarch64的目标代码的，需要用到对应aarch64的编译器版本吧？这个就叫工具链？

A：编译器是工具链中很重要的一部分，但是工具链中还有一部分就是目标平台的系统环境，其中包含了程序依赖的运行库。这些依赖库分为2部分：（1）一部分是属于linux系统的基本库，这部分库通常都跟编译器集成在一起；（2）另一部分是特殊的库，比如我交叉编译在比特大陆边缘计算设备里的深度学习推理程序，还需要比特大陆的推理运行库，这部分库比特大陆会单独提供，包含在其提供的sdk BMNNSDK中。所以这些应该是一个整体，编译器+系统/依赖库。

互动问答5

Q：TensorRT部署比直接原生Pytorch，性能上有哪些优势?

A：首先，使用TensorRT部署和使用原生Pytorch部署，其实并不是一个层面的概念，因为Pytorch也可以使用TensorRT。与TensorRT部署相对应的，是直接使用CUDA；与原生Pytorch部署相对应的，是使用libtorch或者其他框架比如DeepStream部署。

互动问答6

Q：使用原生pytorch部署有什么问题?

A：主要有3个问题：

① 依赖于Python环境和Pytorch环境，这其实一个比较重的依赖，会导致程序比较庞大；

② Python的执行效率没有C/C++高，虽然单从模型推理这部分来说，效率差异也许不是那么明显，毕竟就算是Python，底层很多库也都是用C/C++实现的。但对于预处理、后处理以及视频文件或视频流的解码等操作，Python就和C/C++差别比较大了。

③ Python不利于代码的保护，虽然也有一些措施可以保护源码，比如使用Pyc或者通过修改Python解释器源码将py文件先加密解释执行时再解密，但效果有限，总的来说Python不如C/C++能够更高的保护代码。针对这一问题，Pytorch提供了libtorch库，以方便开发人员使用C/C++代码完成部署。

互动问答7

Q：那到底应该用什么方式部署最为高效?

A：理论上可以采用**TensorRT，**TensorRT是英伟达专门针对模型推理过程提供的优化推理引擎，它会将模型中的很多操作进行裁剪、整合、合并、并行化以及量化，使得模型推理速度提高2-10倍。

但在实际生产环境中，我们会更倾向于使用DeepStream，而不是libtorch，原因在于：

DeepStream使得构建任务流程变得更加简单灵活，他不仅仅可以执行推理，还可以利用英伟达优化过的插件执行视频编解码、多视频流分析等其他也很重要的工作。

在生产环境中部署时，我们应尽可能使用硬件厂商提供的库或SDK，而尽量少的依赖于其他第三方提供的库，因为只有硬件厂商提供的库才是更新和维护最及时的。

互动问答8

Q：DeepStream框架怎么应用部署，有没有典型的参考案例?

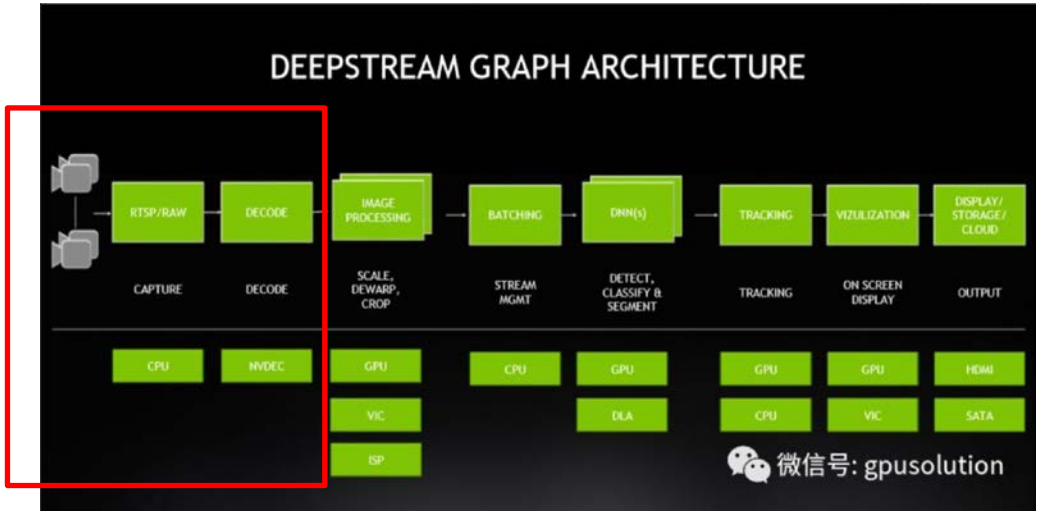
A：DeepStream SDK随附了多个测试应用程序，包括预训练的模型，示例配置文件和可用于运行这些应用程序的示例视频流，具体可以查看解压后的文件夹下Samples文件夹下的内容。DeepStream还内置了**人员检测、车辆检测、车辆分类、人脸检测识别、车牌识别等多个模型和组件。**

具体可以参考官方网站：https://developer.nvidia.com/deepstream-sdk。官方论坛：https://www.nvidia.cn/forums/，Github上也有许多丰富的样例工程，CSDN上也有很多人撰写的教程及说明。

Deepstream简介及部分实践

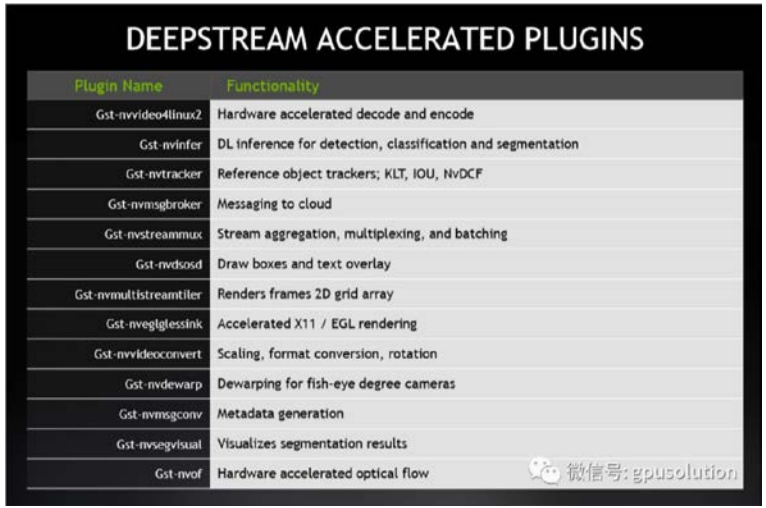
DeepStream 视频硬件编解码及推流

DeepStream SDK是一个流分析工具包。基于开源GStreamer平台的即插即用架构，是一个典型的IVA管道。第一步是捕获流数据。这可能来自RTSP流文件或USB或CSI摄像机；下一步是解码流。解码器插件使用NVIDIA的硬件加速解码引擎--NVIDIA Video CODEC编解码加速单元（与GPU上的CUDA内核不同）。

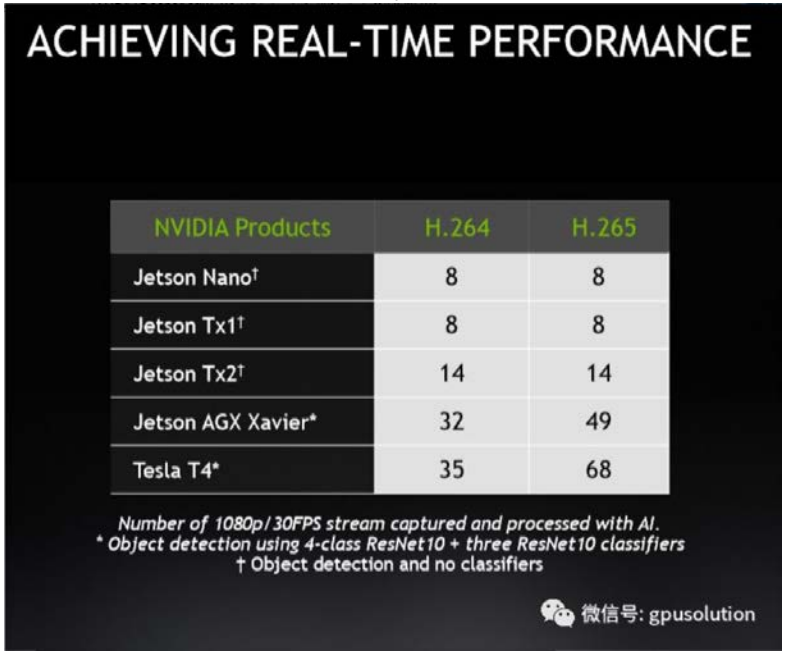


DeepStream 视频硬件编解码及推流

这是DeepStream SDK中包含的插件列表。
第一个是nvvideo4linux2，视频和图像解码和编码插件。
第二个是nvinfer，这是一个推理插件，在各种推理加速器上使用tensorRT，这样就可以对目标检测图像进行分类和分割
第三个是nvtracker，部署几个参考跟踪器，比如KLT、IOU
第四是nvmsgbroker，这允许使用各种协议向云发送消息



不同NVIDIA平台上利用Deepstream达到的实时性能

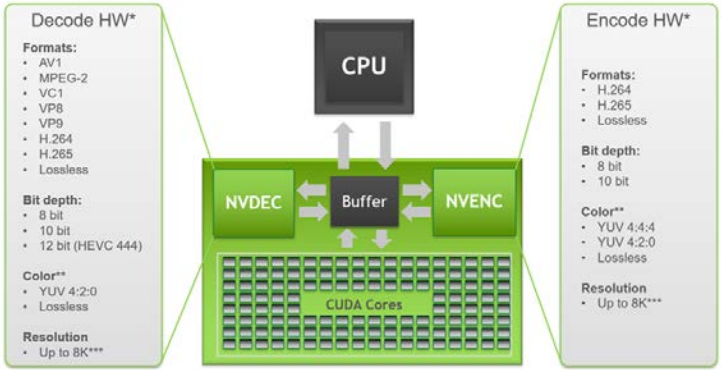


DeepStream 视频硬件编解码及推流

为什么选择显卡编解码?

NVIDIA Codec 硬件编解码有天生的优势。首先，编码性能好，NVDEC (之前叫NVCUVID API)和 NVENC 的性能不是 CPU 可以匹敌的，特别是多路编码的情况下。软件调用显卡去编解码会大大提高效率，节省时间。其次，CPU 在进行编解码时，硬件占用率会很高，这样就会影响其他应用的正常运行，特别是在做图形设计或者游戏过程中，编码占用太多 CPU 资源就会导致应用卡顿。而使用显卡编码不会占用太多系统资源，不会影响应用的使用性能。

NVIDIA 显卡通常都带有硬件编解码的芯片，可以支持各种格式的视频进行解码播放和转换。编码使用 NVENC 芯片，解码使用 NVDEC 芯片。NVIDIA 显卡已经支持大多数的视频格式的编解码了。详细参考下图：



DeepStream 视频硬件编解码及推流

显卡解码

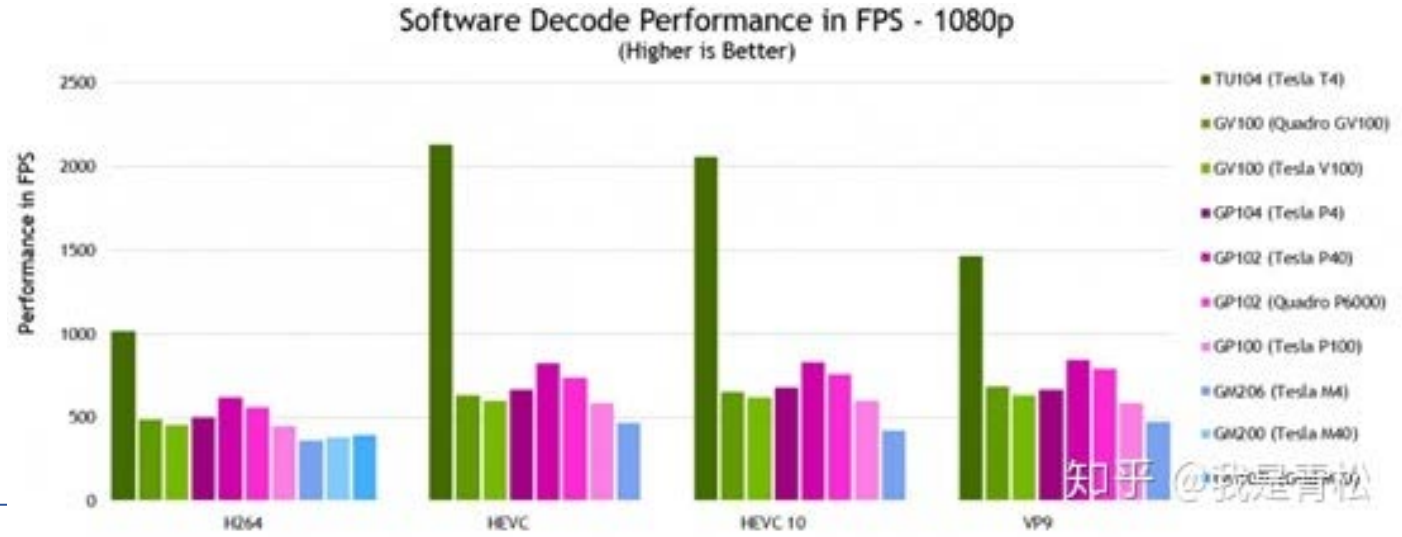
对于解码来讲相对简单一些，解码性能最主要看两个指标就可以了，一个是单解码器解码的帧数，一个是解码芯片数。显卡中带有的NVDEC解码芯片数量可以参考以下表格：

GPU	FAMILY	Total of NVDEC
Quadro P400/P600/P620	Pascal	1
Quadro P1000	Pascal	1
Quadro P2000	Pascal	1
Quadro P4000	Pascal	1
Quadro P5000	Pascal	1
Quadro P6000	Pascal	1
Quadro GP100	Pascal	1
Quadro GV100	Volta	1
Quadro RTX 4000/RTX 5000	Turing	2
Quadro RTX 6000/RTX 8000	Turing	1
Tesla P4	Pascal	1
Tesla T4	Turing	2

DeepStream 视频硬件编解码及推流

通过以下图表，我们了解一下显卡的解码性能：

Tesla T4 解码性能高，是因为他有两个 NVDEC 解码芯片。如果按照一路 1080P 的视频，编码格式为 HEVC，帧率是 30FPS 的话，那么一张 T4 就可以完成 70 路左右视频的解码工作。由于解码芯片的数量相同，所以使用 Quadro RTX 4000 和 RTX 5000 也能达到相同的效果。并且 Quadro 是主动散热带有显示输出的，可以灵活的在工作站上使用。T4 和 P4 由于是被动散热的，所以对环境的要求会更高。



DeepStream 视频硬件编解码及推流

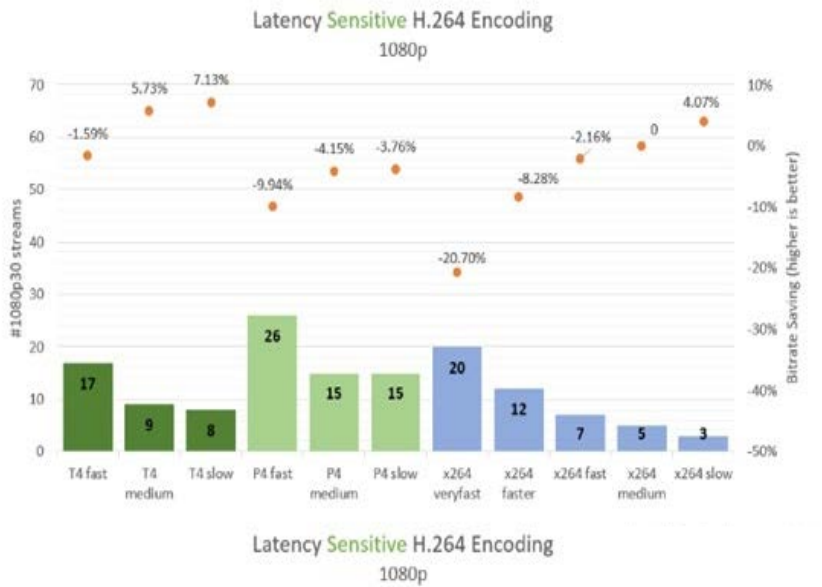
显卡编码

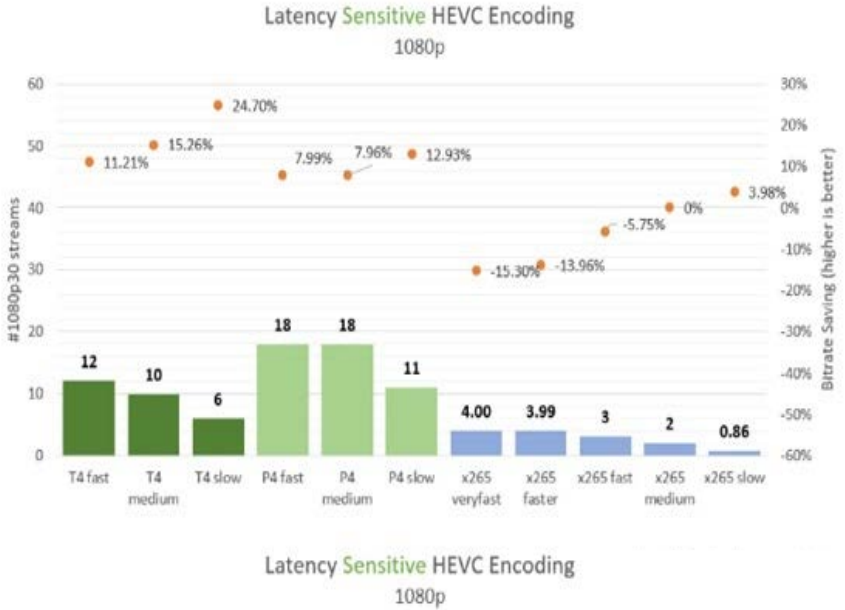
编码使用的是 NVENC 的编码芯片，NVENC 编码器数量越多，编码性能就越高。不同型号显卡 NVENC 编码芯片数量参考下表：

GPU	FAMILY	Total of NVENC	Max of concurrent session
Quadro P400/P600/P620	Pascal	1	3
Quadro P1000	Pascal	1	3
Quadro P2000	Pascal	1	Unrestricted
Quadro P4000	Pascal	1	Unrestricted
Quadro P5000	Pascal	2	Unrestricted
Quadro P6000	Pascal	2	Unrestricted
Quadro GP100	Pascal	3	Unrestricted
Quadro GV100	Volta	3	Unrestricted
Quadro RTX 4000/RTX 5000	Turing	2	Unrestricted
Quadro RTX 6000/RTX 8000	Turing	1	Unrestricted
Tesla P4/P6	Pascal	2	Unrestricted
Tesla T4	Turing	1	Unrestricted

DeepStream 视频硬件编解码及推流

编码性能如何？我们参考下图：





什么是GSteamer?

GStreamer是一个用于开发流式多媒体应用的开源框架，采用了基于**插件（plugin）和管道（pipeline）**的体系结构，框架中的所有的功能模块都被实现成可以插拔的组件（component）， 并且能够很方便地安装到任意一个管道上。由于所有插件都通过管道机制进行统一的数据交换，因此很容易利用已有的各种插件“组装”出一个功能完善的多媒体应用程序。Nvidia为Gstreamer开发了许多plugin，这些plugin能够利用Nvidia硬件进行加速。Nvidia的deepstream就是基于gstreamer开发的。

官网：https://gstreamer.freedesktop.org/

GSteamer 整体框架

1.1 Media Applications

最上面一层为应用，比如gstreamer自带的一些工具（gst-launch，gst-inspect等），以及基于gstreamer封装的库（gst-player，gst-rtsp-server，gst-editing-services等)根据不同场景实现的应用。

1.2 Core Framework

中间一层为Core Framework，主要提供：

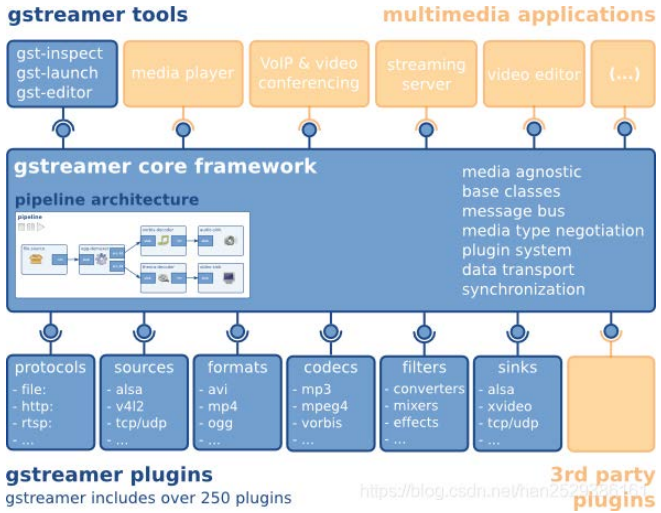
- 上层应用所需接口
- Plugin的框架
- Pipline的框架
- 数据在各个Element间的传输及处理机制
- 多个媒体流（Streaming）间的同步（比如音视频同步）
- 其他各种所需的工具库

1.3 Plugins

最下层为各种插件，实现具体的数据处理及音视频输出，应用不需要关注插件的细节，会由Core Framework层负责插件的加载及管理。
主要分类为：

- Protocols：负责各种协议的处理，file，http，rtsp等。
- Sources：负责数据源的处理，alsa，v4l2，tcp/udp等。
- Formats：负责媒体容器的处理，avi，mp4，ogg等。
- Codecs：负责媒体的编解码，mp3，vorbis等。
- Filters：负责媒体流的处理，converters，mixers，effects等。
- Sinks：负责媒体流输出到指定设备或目的地，alsa，xvideo，tcp/udp等。

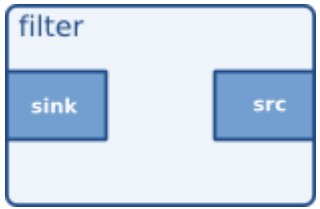
基于Gstreamer框架的应用的简单分层



GSteamer 组件

2.1 Element

Element是Gstreamer中最重要的对象类型之一。一个element实现了一个功能（读取文件，解码，输出等），程序需要创建多个element，并按顺序将其串连起来，构成一个完整的pipeline。



GSteamer 组件

2.2 Pad

Pad是一个element的输入/输出接口，分为src pad（生产数据）和sink pad（消费数据）两种。
两个element必须通过pad才能连接起来，pad拥有当前element能处理数据类型的能力（capabilities），会在连接时通过比较src pad和sink pad中所支持的能力，来选择最恰当的数据类型用于传输。如果element不支持，程序会直接退出。
在element通过pad连接成功后，数据会从上一个element的src pad传到下一个element的sink pad然后进行处理。当element支持多种数据处理能力时，我们可以通过Cap来指定数据类型.

例如，下面的命令通过Cap指定了视频的宽高，videotestsrc会根据指定的宽高产生相应数据：

```
1 gst-launch-1.0 videotestsrc ! "video/x-raw,width=1280,height=720" ! autovideosink
```

GSteamer 组件

2.3 Bin和Pipeline

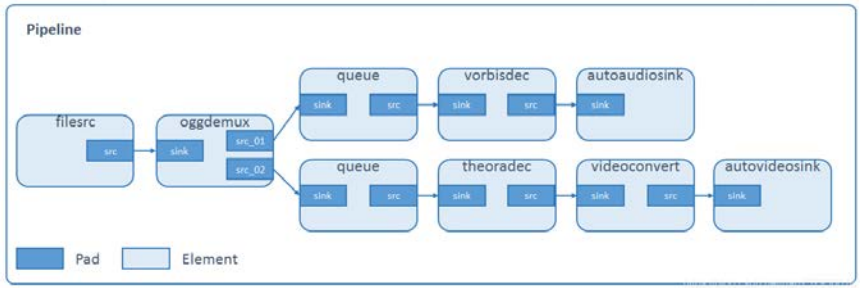
Bin是一个容器，用于管理多个element，改变bin的状态时，bin会自动去修改所包含的element的状态，也会转发所收到的消息。如果没有bin，我们需要依次操作我们所使用的element。通过bin降低了应用的复杂度。
Pipeline继承自bin，为程序提供一个bus用于传输消息，并且对所有子element进行同步。当将pipeline的状态设置为PLAYING时，pipeline会在一个/多个新的线程中通过element处理数据。

下面通过一个文件播放的例子来熟悉上述提及的概念：
测试文件：sintel_trailer-480p.ogv

```
1 | gst-launch-1.0 filesrc location=sintel_trailer-480p.ogv ! oggdemux name=demux ! queue ! vorbisdec ! autoaudiosink
demux. ! queue ! theoradec ! videoconvert ! autovideosink
```

GSteamer 组件

通过上面的命令播放文件时，会创建如下pipeline：



可以看到这个pipeline由8个element构成，每个element都实现各自的功能：

- filesrc读取文件
- oggdemux解析文件，分别提取audio，video数据
- queue缓存数据
- vorbisdec解码audio
- autoaudiosink自动选择音频设备并输出
- theoradec解码video
- videoconvert转换video数据格式
- autovideosink自动选择显示设备并输出

GSteamer 工具

Gstreamer自带了gst-inspect-1.0和gst-launch-1.0等其他命令行工具，我们可以使用这些工具完成常见的处理任务。

3.1 gst-inspect-1.0

查看gstreamer的plugin、element的信息。直接将plugin/element作为参数，会列出其详细信息，包括plugin的功能、Pad的输入输出类型、plugin的属性等。
如果不跟任何参数，会列出当前系统gstreamer所能查找到的所有插件。

3.2 gst-launch-1.0

用于创建及执行一个Pipline，因此通常使用gst-launch先验证相关功能，然后再编写相应应用。
通过上面ogg视频播放的例子，我们已经看到，一个pipeline的多个element之间通过“!”分隔，同时可以设置element及Cap的属性。
下面是解析RTSP视频流的pipeline：

```
gst-launch-1.0 -v rtspsrc location=rtsp://10.201.0.158:8554/vlc ! rtph264depay !  h264parse ! omxh264dec ! nvvidconv ! video/x-raw, width=(int)2048, height=(int)1536, format=(string)BGRx ! videoconvert !  ximagesink sync=false
```

GSteamer 工具

我们可以通过gst-inspect-1.0工具查看每个plugin的功能和属性，选择合适的插件来构成pipeline。

具体在python实现OpenCV+Gstreamer的方法是：OpenCV提供了cv2.VideoCapture()函数，只需把Gstreamer参数传给该函数即可。

具体代码如下：

```
def open_cam_rtsp(uri, width, height, latency):
    gst_str = ("rtspsrc location={} latency={} ! rtph264depay ! h264parse ! omxh264dec ! "
               "nvvidconv ! video/x-raw, width=(int){}, height=(int){}, format=(string)BGRx ! "
               "videoconvert ! appsink").format(uri, latency, width, height)
    return cv2.VideoCapture(gst_str, cv2.CAP_GSTREAMER)

def open_cam_usb(dev, width, height):
    # We want to set width and height here, otherwise we could just do:
    #     return cv2.VideoCapture(dev)
    gst_str = ("v4l2src device=/dev/video{} ! "
               "video/x-raw, width=(int){}, height=(int){}, format=(string)RGB ! "
               "videoconvert ! appsink").format(dev, width, height)
    return cv2.VideoCapture(gst_str, cv2.CAP_GSTREAMER)

def open_cam_onboard(width, height):
    # On versions of L4T previous to L4T 28.1, flip-method=2
    # Use Jetson onboard camera
    gst_str = ("nvcamerasrc ! "
               "video/x-raw(memory:NVMM), width=(int)2592, height=(int)1458, format=(string)I420, framerate=(fractic
               "nvvidconv ! video/x-raw, width=(int){}, height=(int){}, format=(string)BGRx ! "
               "videoconvert ! appsink").format(width, height)
    return cv2.VideoCapture(gst_str, cv2.CAP_GSTREAMER)
```

DeepStream视频结构化

视频结构化的定义

利用深度学习技术实时分析视频中有价值的内容，并输出结构化数据。相比数据库中每条结构化数据记录，视频、图片、音频等属于非结构化数据，计算机程序不能直接识别非结构化数据，因此需要先将这些数据转换成有结构格式，用于后续计算机程序分析。视频结构化最常见的流程为：**目标检测**、**目标分类（属性识别）**、**目标跟踪**、**目标行为分析**。

目标检测

对单张图片中感兴趣的目标进行识别、定位，注意两点，一个是检测的对象是静态图片，二是不但需要识别目标的类别，还需要给出目标在原图片中的坐标值，通常以（left, top, width, height）的形式给出。注意目标检测仅仅给出目标大概位置坐标（一个矩形区域），它跟图像分割不同，后者定位更加具体，能够给出图片中单个目标的轮廓边界。

DeepStream视频结构化

目标分类（属性识别）

通常目标被检测出来之后，会进行二次（多次）推理，识别出目标更加具体的属性，比如小轿车的颜色、车牌子奥迪还是奔驰等等。对于人来讲，可以二次推理出人的性别、年龄、穿着、发型等等外貌属性。这个环节主要对检测出来的目标进行更加具体的属性识别。

目标跟踪

前面两个环节操作的对象是静态单张图片，而视频有时序性，前后两帧中的目标有关联关系。目标跟踪就是为了将视频第N帧中的目标和第N+1帧中的同一目标关联起来，通常做法是给它们赋予同一个ID。经过目标跟踪环节后，理论情况下，一个目标从进入视频检测范围到离开，算法赋予该目标的ID固定不变。但是现实生产过程中，由于各种原因，比如目标被遮挡、目标漏检（第N帧检测到，第N+1帧没检测到）、跟踪算法自身准确性等等原因，系统并不能锁定视频中同一个目标的ID。目标ID不能锁定，会造成目标行为分析不准的问题，后面会提到。

DeepStream视频结构化

目标行为分析

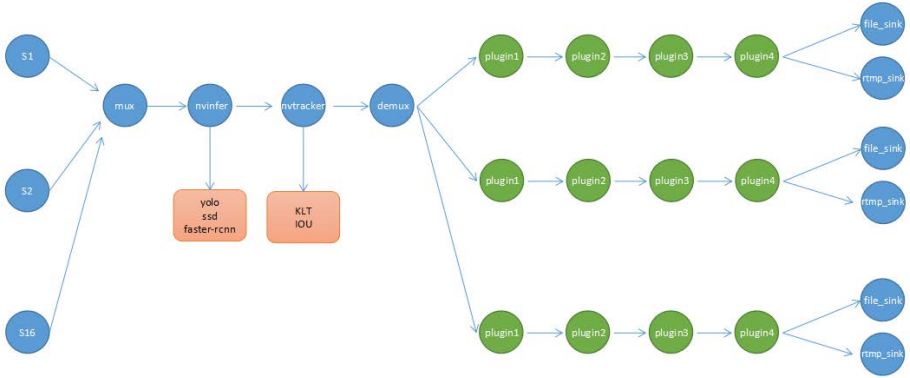
视频中目标被跟踪到，赋予唯一ID之后，我们可以记录目标在视频检测范内的运动轨迹（二维坐标点集合），通过分析目标轨迹点数据，我们可以做很多应用。比如目标是否跨域指定区域、目标运动方向、目标运动速度、目标是否逗留（逗留时长）、目标是否密集等等。该应用多存在于安防、交通视频分析领域。

目标检测算法

常见基于深度学习神经网络的目标检测算法有3种，SSD、YOLO以及Faster-RCNN，三者各有优劣，遵循一个原则：速度快的准确性不好，很多目标检测不准，很多小目标检测不到、容易漏检等；准确性好的速度不快，可能达不到实时检测的要求，或者需要更高的硬件条件。鱼和熊掌不可兼得，牺牲速度可以换来准确性。综合性比较好的是YOLO（YOLOv1/v2/v3/v4/YOLO X），准确性、小目标检测、检测速度上都可以接受。

DeepStream视频结构化

视频结构化处理流程框架



（1）内置推理加速插件nvinfer，不断支持各种目标检测算法（SSD、YOLO、Faster-RCNN）以及各种深度学习框架模型（自由切换），内部还自带tensorRT INT8/FP16加速功能，不需要你做额外操作；

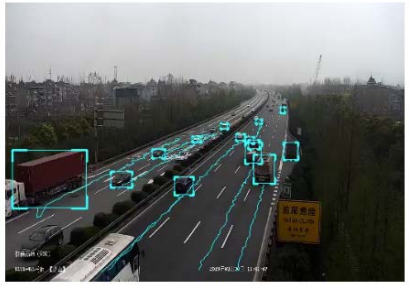
（2）内置目标跟踪插件nvtracker，目前DeepStream提供两种跟踪算法，一种基于IOU的，这种算法简单，但是快；另外一种KLT算法，准确但是相对来讲慢一些，而且由于这个算法是跑在CPU上，基于KLT的跟踪算法对CPU占用相对大一些；

（3）内置其他比较有用的插件，比如用于视频叠加（目标方框叠加到视频中）的nvosd、硬件加速解码插件nvdec_h264，专门采用GPU加速的解码插件，还有其他颜色转换的插件；

（4）提供跟视频处理有关的各种元数据类型以及API，方便你扩展自己的元数据类型

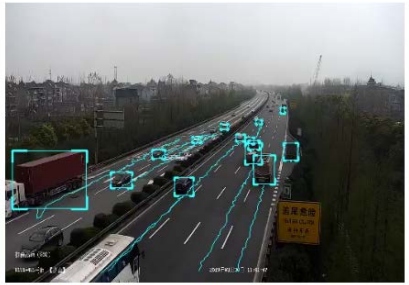
DeepStream视频结构化

http://192.168.77.21/flv?app=live&stream=10012



http://192.168.77.21/flv?app=live&stream=10013

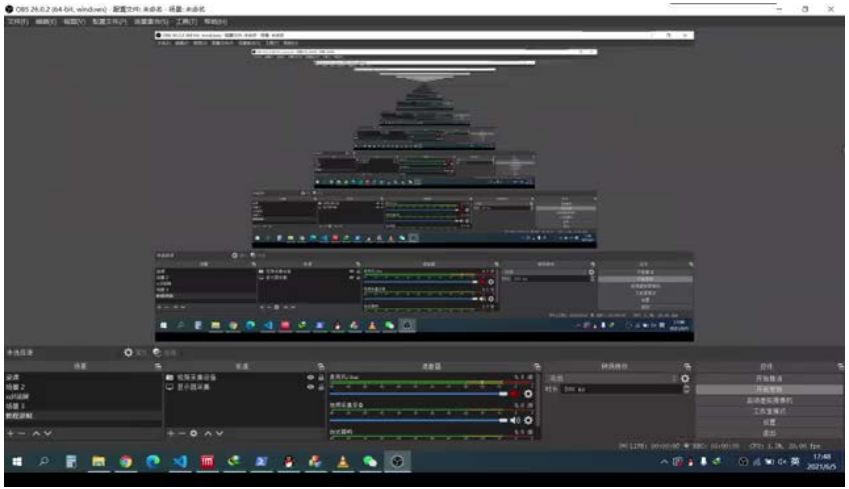
http://192.168.77.21/flv?app=live&stream=10012

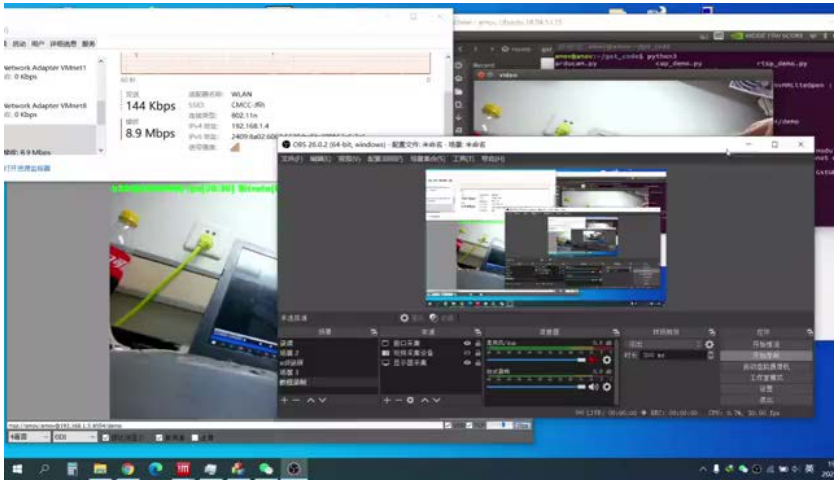


http://192.168.77.21/flv?app=live&stream=10013

DeepStream 视频硬件编解码及推流

实验配置：
编码工具：DeepStream中的NVENC；
视频压缩格式：H.264
分辨率：1080p (1920x1080)
推流格式：rtsp
平台：Jetson Xavier NX
测试网络环境：wifi





NoMachine远程控制软件

我们用Jetson开发的时候，会遇到远程显示的需求，常用的远程控制软件比如向日葵、teamviewer、RealVNC不支持arm版本，因此我们一般会选择支持arm linux的远程控制软件NoMachine。NoMachine基本覆盖了所有主流大操作系统，包括Windows，Mac，Linux，iOS，Android和Raspberry等等，还是很方便的，并且可以免费使用。

官网链接：https://www.nomachine.com/

直接播放海康rtsp流

如果这样粗暴简单的读取视频流，延迟很大（秒级），图像帧累加，画面不同步

```
import cv2
camera = cv2.VideoCapture("rtspsrc location=rtsp://169.254.160.104/av0_1
latency=0 drop-on-latency=true max-size-buffers=0 ! decodebin ! nvvidconv !
video/x-raw, format=I420 ! appsink sync=0", cv2.CAP_GSTREAMER)
```

解决办法：使用GSTEAMER实现GPU硬件解码视频流

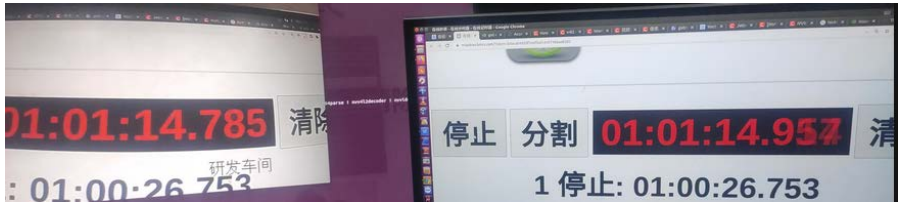
TIPS
如果你对ffmpeg比较熟悉，那你得改成gstreamer，在nvidia上，ffmpeg有些功能不支持。不过264/265没问题。GStreamer开发一般先用命令行进行验证，在进行代码编写，如果你只使用视频相关的业务，用c进行code就可以，如果你还需要进行视觉识别，深度计算…那就得c++和python了。我只进行了视频相关的命令行学习与验证。

使用gstreamer gst-launch-1.0命令用英伟达的硬件解码低延迟播放rtsp视频流的例子

```
gst-launch-1.0 rtspsrc location=rtsp://169.254.160.104:554/av0_0 latency=0 drop-on-latency=true max-size-
buffers=0 ! decodebin ! nvoverlaysink sync=false -e
```

同样的，你也可以使用opencv（cv2.VideoCapture）来实现低延迟gstreamer pipeline（后端）来拉流，如下
（opencv编译的时候必须在cmake中将-DWITH_GSTREAMER=ON打开，否则会报错）

```
import cv2
camera = cv2.VideoCapture("rtspsrc location=rtsp://169.254.160.104/av0_1 latency=0 drop-on-latency=true max-size-
buffers=0 ! decodebin ! nvvidconv ! video/x-raw, format=I420 ! appsink sync=0", cv2.CAP_GSTREAMER)
```



DeepStream 多路拉流

DeepStream4.0支持很方便地直接读取rtsp摄像头，只需更改config文件即可。
以自带例程为例，打开configs下要运行的txt设置文件，找到source块，如[source0]，修改type和uri即可。
例如，

```
[source0]
enable=1
type=4
uri=rtsp://你的地址
```

type可以为：

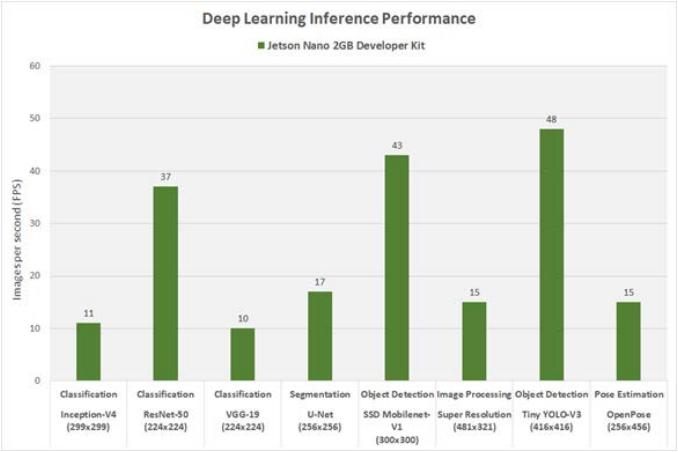
```
1: Camera (V4L2)
2: URI
3: MultiURI
4: RTSP
5: Camera (CSI) (Jetson only)
```

修改保存后运行如下命令即可看到效果，多路摄像头就多加几个source

```
deepstream-app -c 你的配置文件.txt
```

2GB版Jetson Nano基于DeepStream实现多路视频实时分析

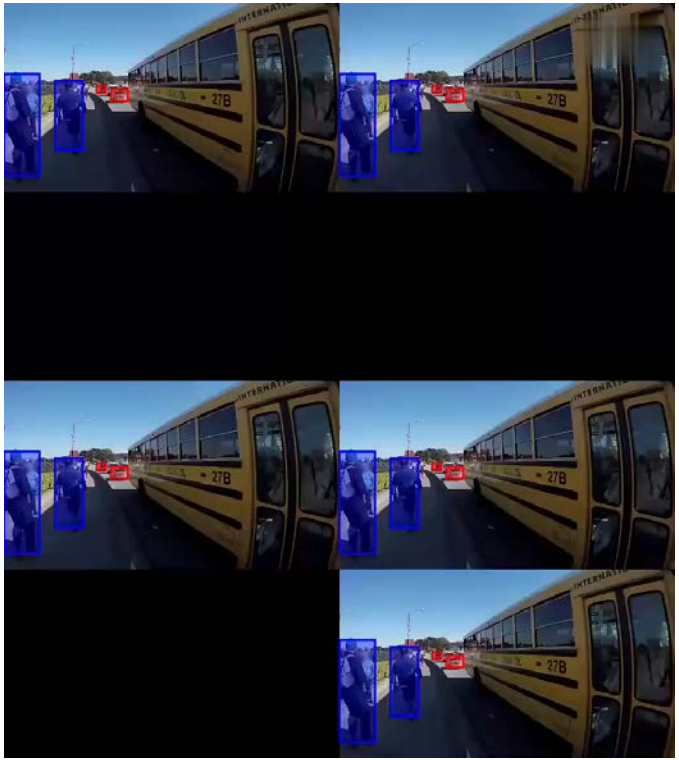
2GB版Jetson Nano基于DeepStream实现多路视频实时分析



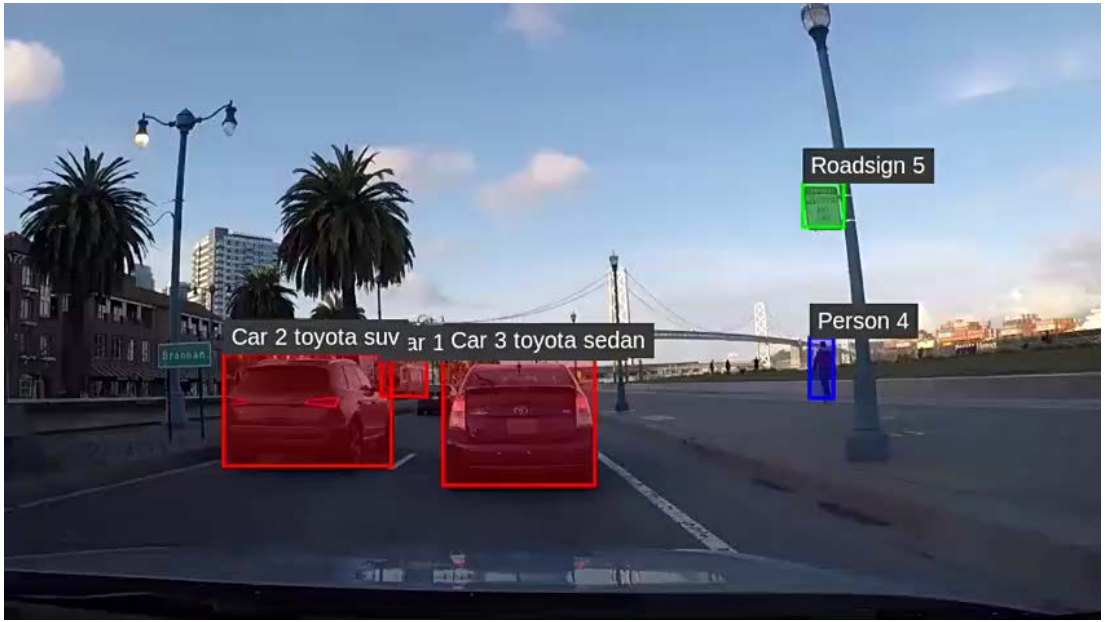
Jetson Nano 2GB Developer Kit inferencing performance (FP16) for various vision-based DNN models with JetPack 4.4.1 and TensorRT 7.1.

2GB版Jetson Nano基于DeepStream实现多路视频实时分析

在Jetson Nano 2GB上进行8路视频实时分析



2GB版Jetson Nano基于DeepStream实现多路视频实时分析



在Jetson Nano 2GB上用高精度预训练模型DashCamNet，VehicleMakeNet，VehicleTypeNet
进行检测+分类+追踪推理的效果

2GB版Jetson Nano基于DeepStream实现多路视频实时分析

在Jetson Nano 2GB上进行4路海康摄像头视频实战场景实时分析

2GB版Jetson Nano基于DeepStream实现多路视频实时分析

- 在Transfer Learning Toolkit（TLT）中对于大多数检测任务使用Nvidia的DetectNet_v2训练往往就能获得不错的效果。推理速度快并在不同尺寸的目标上都有较高的准确性。如果仍不能满足需求，还有其它主流模型可供选择，例如Faster-RCNN，YOLOv3，SSD，DSSD，RetinaNet，MaskRCNN，YOLOv4

- 总体来说，Jetson Nano的算力在当下众多AI硬件中或许并不算突出。但其价格持续下探，逼近树莓派，上手门槛越来越低，无疑对开发者是非常友好的。Nvidia显然也希望有更多开发者能参与进来，进一步挖掘Jetson的应用潜力。上面诸多示例也几乎不需要修改就可以跑在Jetson平台的其它硬件上。如果使用Jetson AGX Xavier，扩展到32路视频进行实时分析应该是没有问题的。

各种推理部署架构


ONNX
TNN
OpenVINO
MACE
Mobile AI Compute Engine
腾讯优图
NCNN
MNN
Mobile Neural Network
MediaPipe
NVIDIA TensorRT
可编程推理加速器
tvm

NCNN

- NCNN是腾讯优图实验室首个开源项目，于2017年7月正式开源，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；

- 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

NCNN

Current building status matrix

System	CPU (32bit)	CPU (64bit)	GPU (32bit)	GPU (64bit)
Linux (GCC)	build passing	build passing	—	build passing
Linux (Clang)	build passing	build passing	—	build passing
Linux (ARM)	build passing	build passing	—	—
Linux (MIPS)	build passing	build passing	—	—
Linux (RISC-V)	—	build passing	—	—
Windows (VS2015)	build passing	build passing	—	—
Windows (VS2017)	build passing	build passing	—	build passing
Windows (VS2019)	build passing	build passing	—	build failing
macOS	—	build passing	—	build passing
macOS (ARM)	—	build passing	—	build passing
Android	build passing	build passing	build passing	build passing
Android-x86	build passing	build passing	build passing	build passing
iOS	build passing	build passing	—	build passing
iOS Simulator	build passing	build passing	—	—
WebAssembly	build passing	—	—	—
RISC-V GCC/Newlib	build passing	build passing	—	—

NCNN

supported platform matrix

- ✅ = known work and runs fast with good optimization
- ✔ = known work, but speed may not be fast enough
- ? = shall work, not confirmed
- / = not applied

	Windows	Linux	Android	macOS	iOS
intel-cpu	✔	✔	?	✔	/
intel-gpu	✔	✔	?	?	/
amd-cpu	✔	✔	?	✔	/
amd-gpu	✔	✔	?	?	/
nvidia-gpu	✔	✔	?	?	/
qcom-cpu	?	✔	✅	/	/
qcom-gpu	?	✔	✔	/	/
arm-cpu	?	?	✅	/	/
arm-gpu	?	?	✔	/	/
apple-cpu	/	/	/	✔	✅
apple-gpu	/	/	/	✔	✔

NCNN的官方代码地址：
https://github.com/Tencent/ncnn

移动端的部署工具除了NCNN，还有**华盛顿大学的TVM、阿里的MNN、小米的MACE、腾讯优图基于NCNN开发的TNN**等推理部署工具。

OpenVINO

- OpenVINO工具套件全称是**O**pen **V**isual **I**nference & **N**eural Network **O**ptimization，是Intel于2018年发布的，**开源、商用免费**、主要应用于计算机视觉、实现神经网络模型优化和推理计算(Inference)加速的软件工具套件；
- OpenVINO是一个Pipeline工具集，同时可兼容各种开源框架训练好的模型，拥有算法模型上线部署能力，帮助用户将预训练模型在Intel的CPU上快速部署起来；
- OpenVINO提供了深度学习推理套件（DLDT)，包含了图片处理工具包OpenCV，视频处理工具包Media SDK，用于处理图像视频解码，前处理和推理结果后处理等；
- OpenVino目前支持Linux、Windows、macOS、Raspbian等系统平台。

Why OpenVINO?

- AI 应用原生的数据格式并不统一，大多数 AI 模型多采用传统的 FP32 数据格式，完全可以在损失很小精度的前提下转换成 BF16 或 INT8 格式，在 CPU 上实现 INT8 和 BF16 数据处理加速；
- 要想真正用好这些加速特性，要么得人为地对 AI 模型实施数据格式转换，要么就得用更为专业和省心的工具来帮忙。人为转换显然费时费力，不仅无法根据处理器平台特性实施优化，且转换后的模型也无法兼容不同的硬件平台；
- 由英特尔推出的 OpenVINO™ 工具套件，提供模型量化功能，能让基于不同 AI 框架，如 TensorFlow、MXNet、PyTorch 等构建的 FP32 数据格式 AI 模型，在损失很少精度的情况下转化为 INT8 和 BF16 数据格式

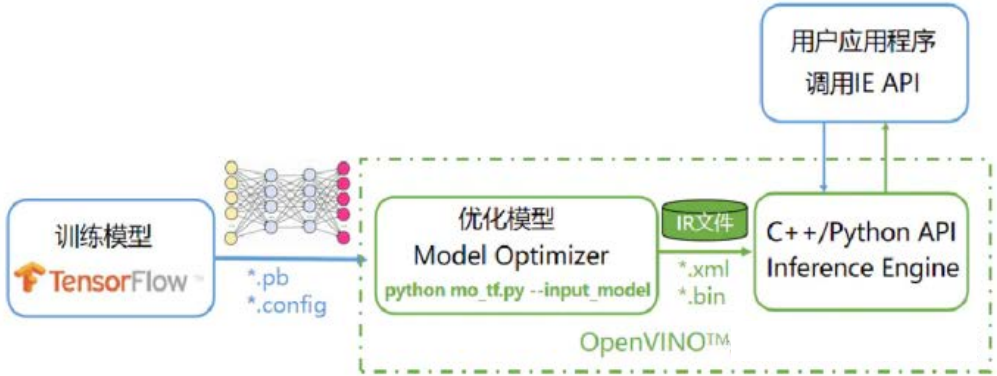
OpenVINO™ 工具套件提供的模型量化功能

OpenVINO

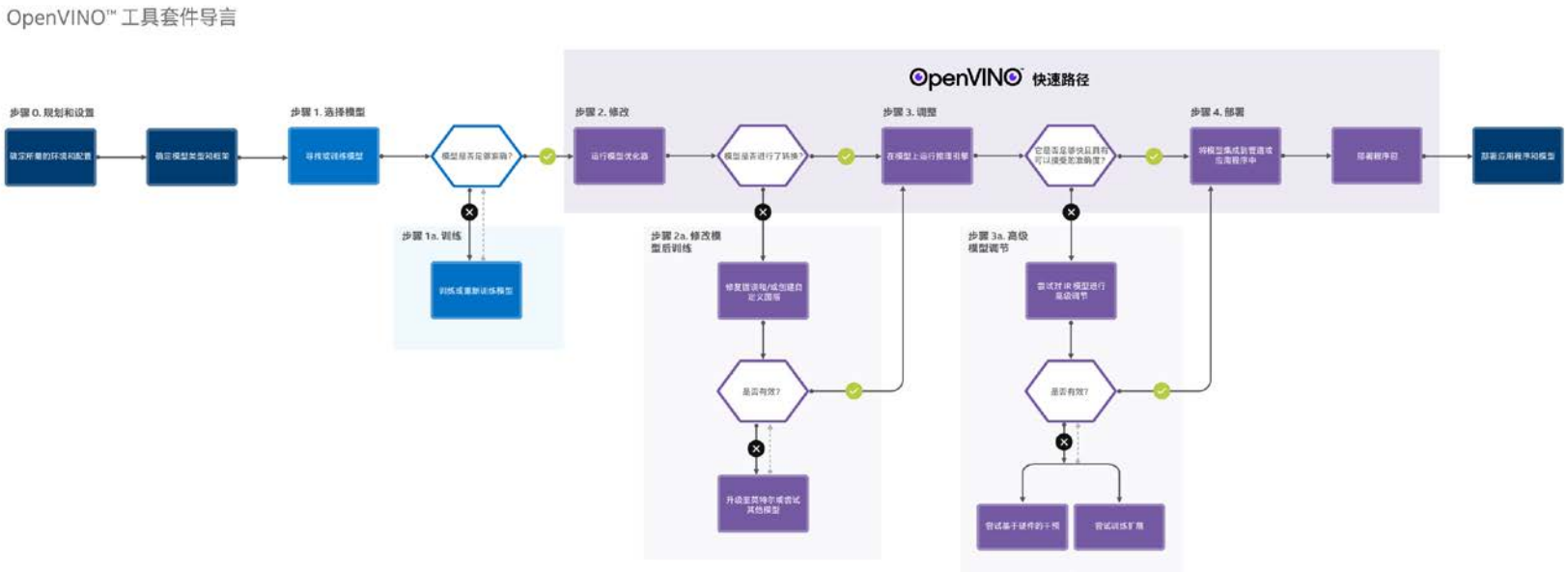
OpenVINO工具套件主要包括：Model Optimizer(模型优化器)——用于优化神经网络模型的工具，Inference Engine(推理引擎)——用于加速推理计算的软件包。

模型优化器的作用包括压缩模型和加速，比如，去掉推理无用的操作(Dropout)，层的融合(Conv + BN + Relu)，以及内存优化

推理引擎是一个支持C\C++和python的一套API接口，需要开发人员自己实现推理过程的开发。核心流程如下：
装载处理器的插件库->
读取网络结构和权重->
配置输入和输出参数->
装载模型->
创建推理请求->
准备输入Data ->
推理->
结果处理

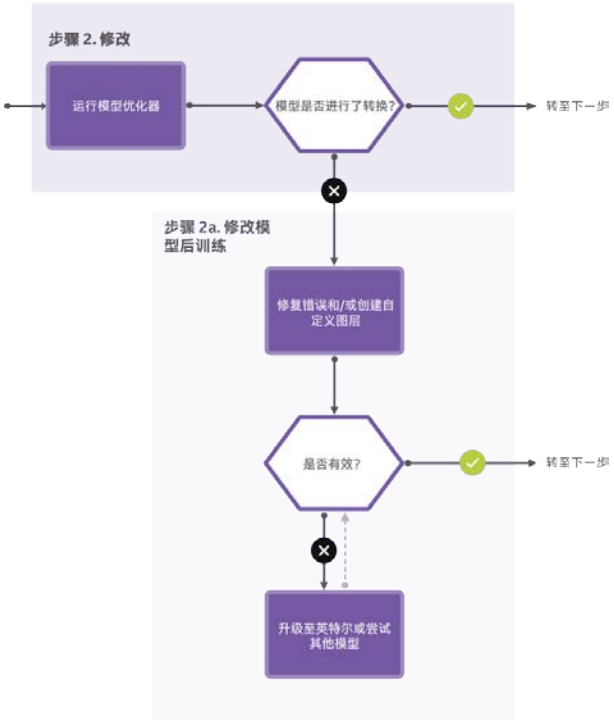


OpenVINO



OpenVINO™ 工具套件工作流程

OpenVINO



模型准备、转换和优化

使用自己选择框架来准备和训练深度学习模型，或从 Open Model Zoo 下载预训练模型。Open Model Zoo 包含针对各种视觉问题的深度学习解决方案，包括在一定程度的复杂度下的对象识别、人脸识别、姿态估计、文本检测和动作识别。这些预训练模型中的某些模型还用于代码样本和应用程序演示。要从 Open Model Zoo 下载模型，需要使用模型下载器工具。

OpenVINO™ 工具套件的一个核心组件是模型优化器，它是一个跨平台命令行工具，可将经过训练的神经网络从源框架转换为与 nGraph 兼容的开源中间表示 (IR)，用于推理运算。模型优化器导入在 Caffe*、TensorFlow*、MXNet*、Kaldi* 和 ONNX* 等常用框架中经过训练的模型，并执行几项优化，以尽可能删除过多的层和群运算，以更简单、更快速地形成图表。

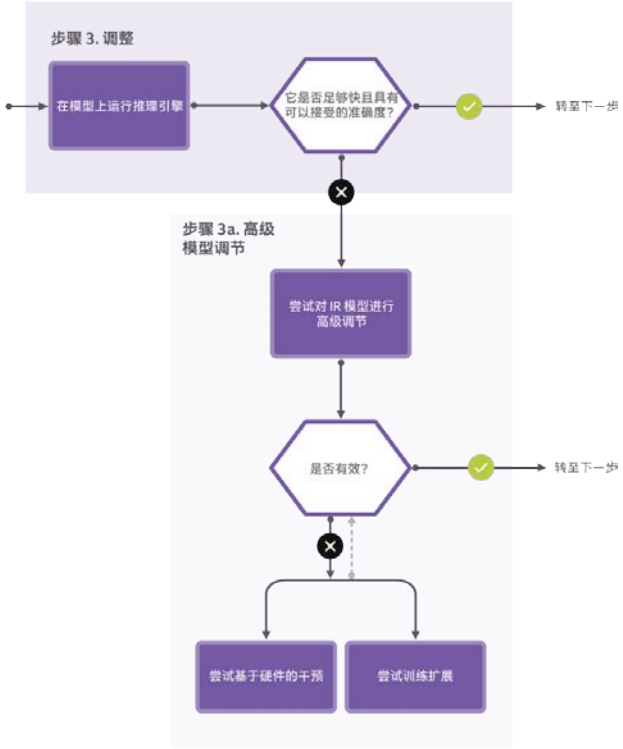
使用训练后优化工具，通过将其量化为 INT8 来加速深度学习模型的推理。

OpenVINO

推理的运行和调优

OpenVINO™ 的另一个核心组件是推理引擎，它管理经过优化的神经网络模型的加载和编译，在输入数据上运行推理运算，并输出结果。推理引擎可以同步或异步执行，其插件架构管理用于在多个英特尔® 设备上执行的适当编译，包括主力 CPU 以及专用显卡和视频处理平台

可以将 OpenVINO™ 调整实用程序与推理引擎一起使用，在模型上试用和测试推理。基准测试实用程序使用输入模型运行迭代测试，以检测吞吐量或延迟，交叉检查实用程序对不同配置的推理的性能进行比较。



OpenVINO

英特尔® 发行版 OpenVINO™ 工具套件包含以下组件：

•深度学习模型优化器：一种跨平台的命令行工具，用于导入模型并为推理引擎的优化执行准备模型。模型优化器会导入、转换、优化模型，这些模型在流行的框架中训练，比如 Caffe*、TensorFlow*、MXNet*、Kaldi* 和 ONNX*；

•深度学习推理引擎：一个统一的 API，可在多种硬件类型（包括英特尔® CPU、英特尔® 集成显卡、英特尔® 神经电脑棒 2、搭载英特尔® Movidius™ 视觉处理器 (VPU) 的英特尔® Vision Accelerator Design）上进行高性能推理；

•推理引擎样本：一套简单的控制台应用程序，展示了如何在您的应用程序中使用推理引擎；

•深度学习工作台：基于网页的图形环境，让您可以轻松使用各种复杂的 OpenVINO™ 工具套件组件；

•训练后优化工具：一种对模型进行校准，然后以 INT8 精度执行的工具；

•其他工具：一组用于配合您的模型的工具，包括基准测试应用程序、交叉检查工具、编译工具；

•Open Model Zoo
  - 演示 - 提供强大应用程序模板，帮助您实施特定的深度学习场景的一款控制台应用程序。
  - 其他工具：一组用于处理模型的工具，包括精度检查实用程序和模型下载器。
  - 适用于预训练模型的文档：适用于预训练模型的文档，可以在 Open Model Zoo 存储库中找到。

•深度学习流媒体播放器 (DL Streamer)：基于 GStreamer，用于构建媒体分析组件图的流媒体分析框架。DL Streamer 可以通过英特尔® 发行版 OpenVINO™ 工具套件安装程序安装。其开源版本可在 GitHub 上找到。

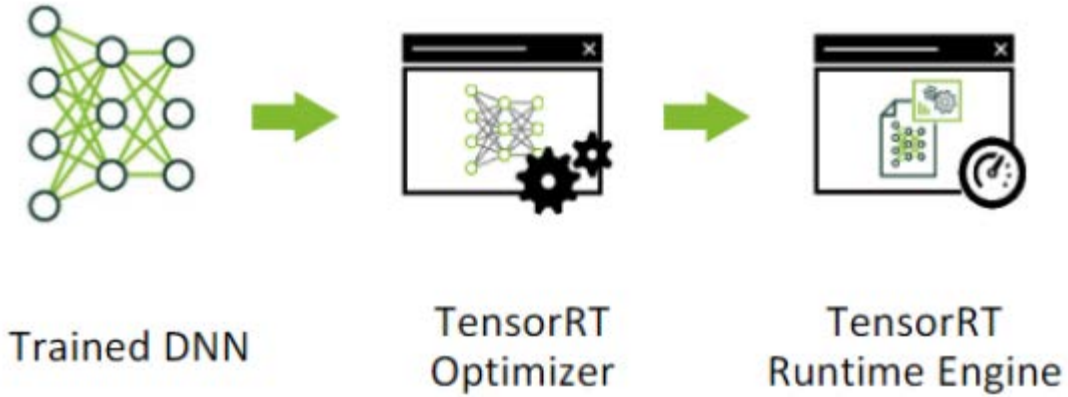
•OpenCV：为英特尔® 硬件编译的 OpenCV* 社区版本

•Intel® Media SDK（仅位于面向 Linux 的英特尔® 发行版 OpenVINO™ 工具套件中）

TensorRT

什么是TensorRT

TensorRT是由Nvidia推出的C++语言开发的高性能神经网络推理库，是一个用于**生产部署**的优化器和运行时引擎。其高性能计算能力依赖于Nvidia的图形处理单元。它专注于推理任务，与常用的神经网络学习框架形成互补，包括TensorFlow、Caffe、PyTorch、MXNet等。可以直接载入这些框架的已训练模型文件，也提供了API接口通过编程自行构建模型。

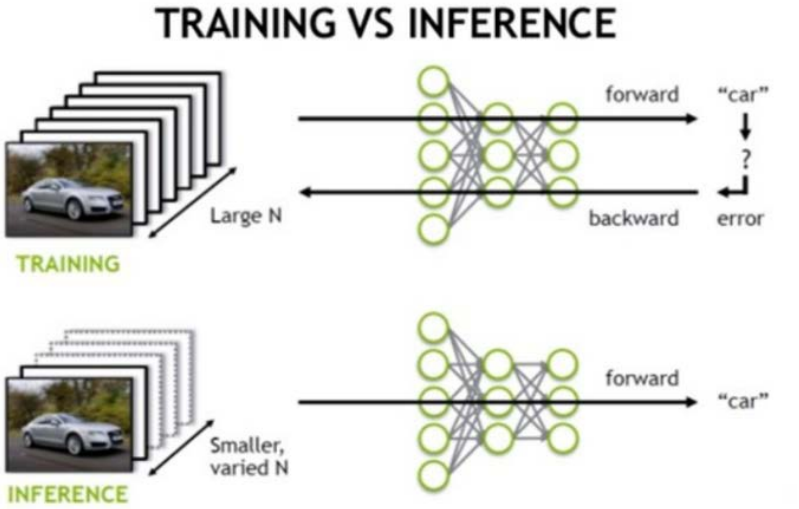


TensorRT

Why TensorRT

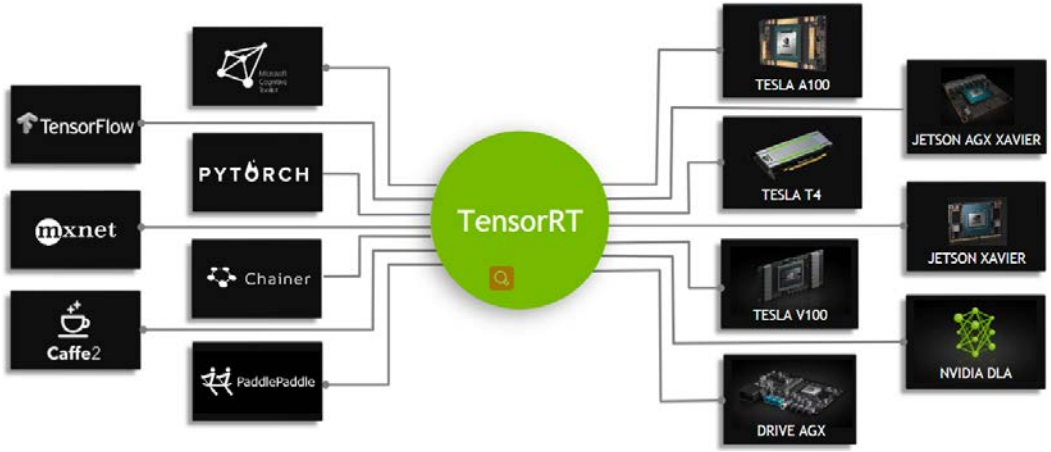
一般的深度学习项目，训练时为了加快速度，会使用多GPU分布式训练。但在部署推理时，为了降低成本，往往使用单个GPU机器甚至嵌入式平台（比如NVIDIA Jetson）进行部署，部署端也要有与训练时相同的深度学习环境，如caffe，TensorFlow等。由于训练的网络模型可能会很大（比如，inception，resnet等），参数很多，而且部署端的机器性能存在差异，就会导致推理速度慢，延迟高。这对于那些高实时性的应用场合是致命的，比如自动驾驶要求实时目标检测，目标追踪等。所以为了提高部署推理的速度，出现了很多轻量级神经网络，比如squeezenet，mobilenet，shufflenet 等。基本做法都是基于现有的经典模型提出一种新的模型结构，然后用这些改造过的模型重新训练，再重新部署。而 TensorRT 则是对训练好的模型进行优化。TensorRT就只是推理优化器。当你的网络训练完之后，可以将训练模型文件直接丢进TensorRT中，而不再需要依赖深度学习框架。

TensorRT



可以认为TensorRT是一个只有**前向传播**的深度学习推理框架，这个框架可以将Caffe，TensorFlow，PyTorch等网络模型解析，然后与TensorRT中对应的层进行一一映射，把其他框架的模型统一全部转换到TensorRT中，最后在 TensorRT中可以针对 NVIDIA自家GPU 实施优化策略，并进行部署加速。

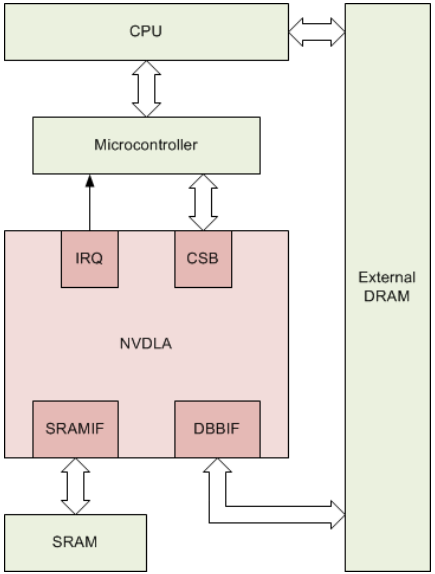
TensorRT



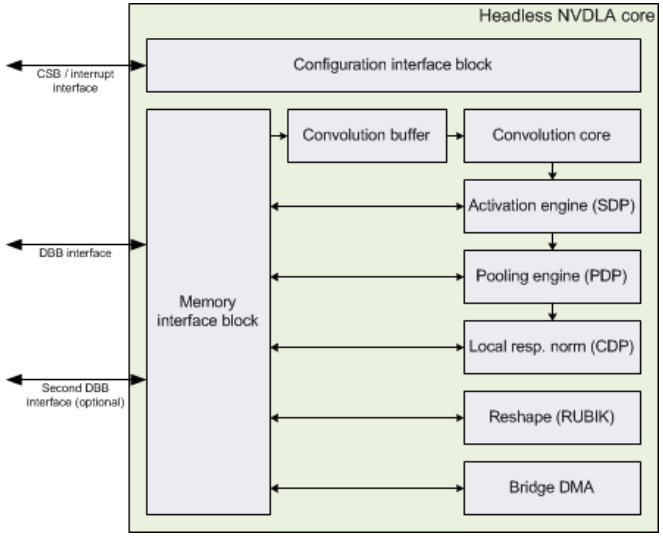
- TensorRT依赖于Nvidia的深度学习硬件环境，可以是GPU也可以是DLA，如果没有的话则无法使用；
- TensorRT支持目前大部分的神经网络Layer的定义，同时提供了API让开发者自己实现特殊Layer的操作

NVIDIA DLA

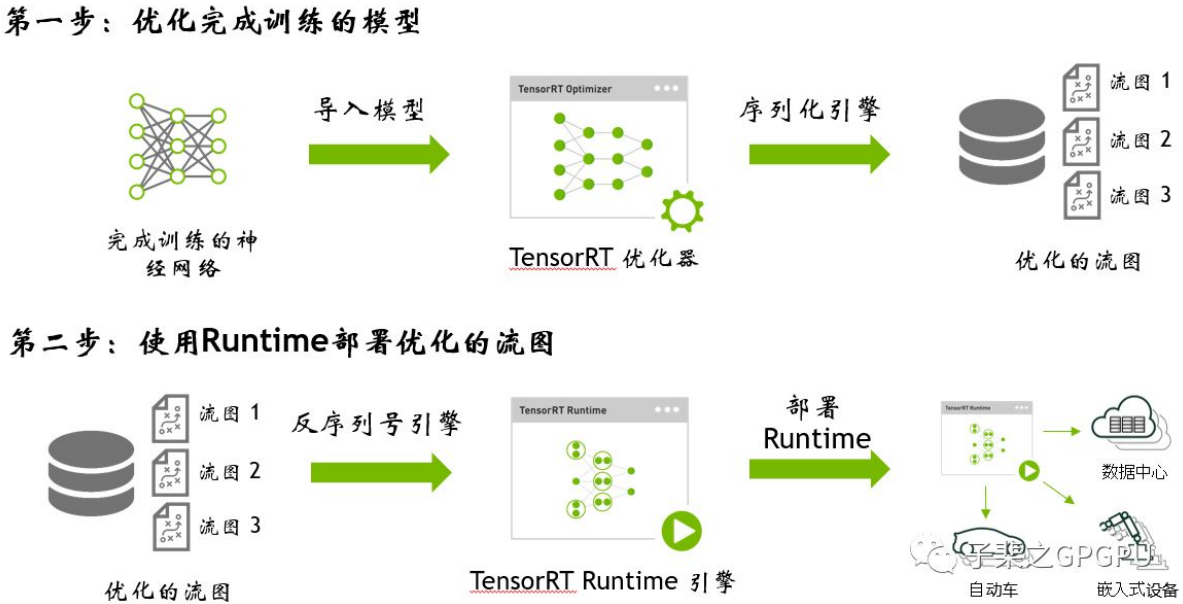
NVDLA（Deep Learning Accelerator）其实就是一个卷积神经网络加速器（只能推断，并不能进行训练），它还需要外部的CPU和内存单元才能完整驱动整个加速器，CPU通中断和CSB总线控制NVDLA加速器。



NVDLA的设计本身并没有特别创新之处，主要有卷积、池化、非线性激活函数操作等运算，为了降低带宽，权重阈值还进行了压缩



TensorRT的部署



TensorRT的部署

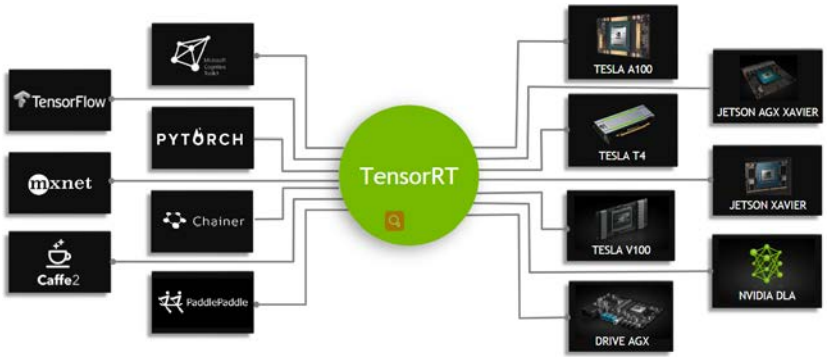
1. TensorRT支持什么框架训练出来的网络模型呢?

TensorRT3支持所有常见的深度学习框架包括Caffe、Tensorflow、Pytorch、MXnet、PaddlePaddle、Theano等

2. TensorRT优化好的计算流图可以运行在什么设备上?

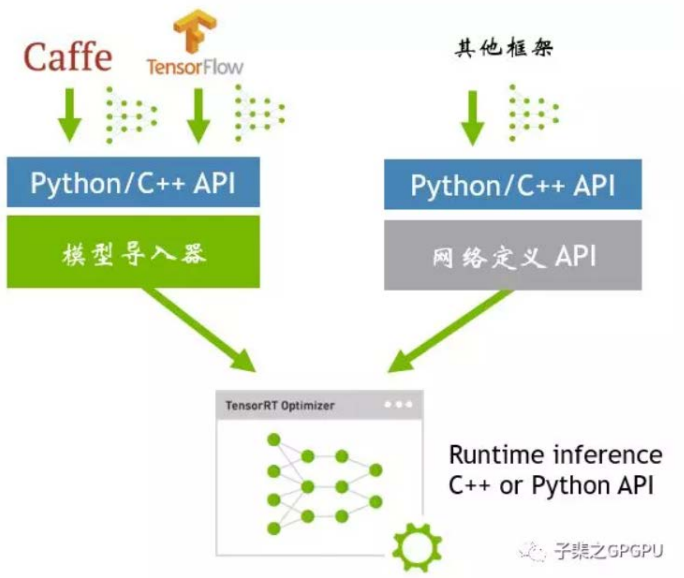
支持的系统平台：TensorRT3支持的平台包括Linux x86、Linux aarch64、Android aarch64和QNX aarch64

支持的硬件平台：TensorRT3可以运行在每一个GPU平台，从数据中心的Tesla P4/V100到自动驾驶和嵌入式平台的DrivePX及TX1/TX2



TensorRT的部署

TensorRT模型导入流程



如上图所示，模型导入方法可以根据框架种类分成三种：Caffe、Tensorflow和其他

caffe

1. 使用C++/Python API导入模型：通过代码定义网络结构，并载入模型weights的方式导入；
2. 使用NvCaffeParser导入模型：导入时输入网络结构prototxt文件及caffemodel文件即可

Tensorflow

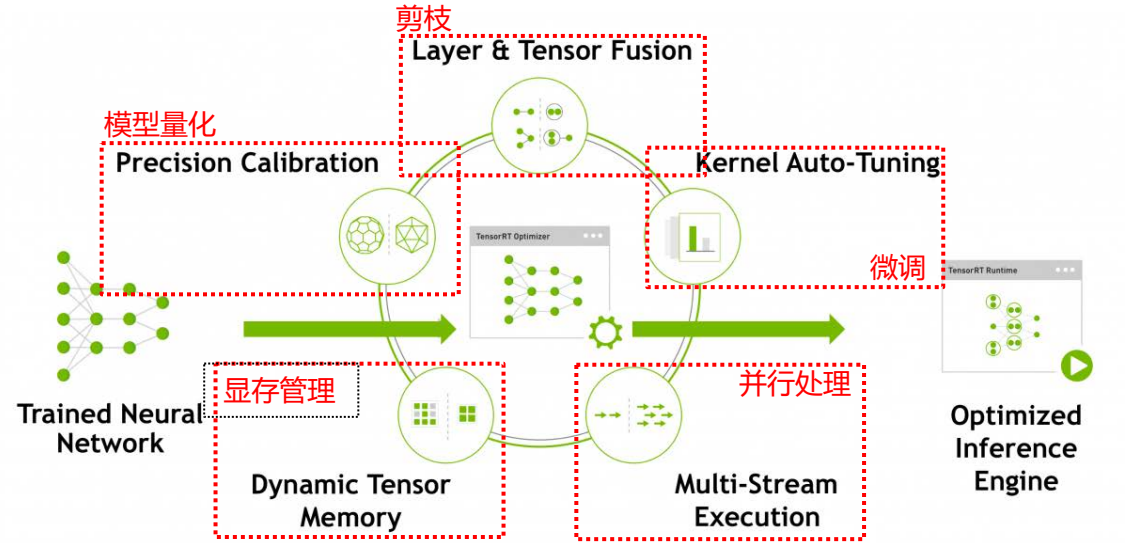
1. 训练完成后，使用uff python接口将模型转成uff格式，之后使用NvUffParser导入；
2. 对于Tensorflow或者keras的，利用Freezegraph来生成.pb(protobuf)文件，之后使用convert-to-uff工具将.pb文件转化成uff格式，然后利用NvUffParser导入

其他框架

使用C++/Python API导入模型：通过代码定义网络结果，载入模型weights的方式导入。以Pytorch为例，在完成训练后，通过stat_dict()函数获取模型的weights，从而定义网络结构时将weights载入

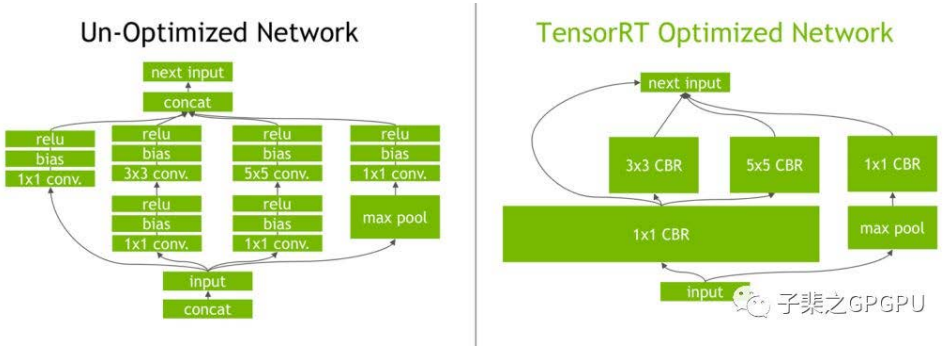
TensorRT的部署

TensorRT优化细节



TensorRT的部署

Layer & Tensor Fusion　层与张量的融合



1.Kernel纵向融合：如上图所示，卷积Conv、Bias和Relu层可以融合成一个Kernel，这里称之为CBR；

2.Kernel横向融合：如上图中超宽的1x1 CBR所示；

3.消除concatenation层，通过预分配输出缓存以及跳跃式的写入方式来避免这次转换。

TensorRT的部署

Kernel Auto-Tuning

TensorRT会针对大量的Kernel进行参数优化和调整。例如说，对于卷积计算有若干种算法，TensorRT会根据输入数据大小、filter大小、tensor分布、batch大小等等参数针对目标平台GPU进行选择和优化。

FP16 & INT8精度校准

大多数的网络都是使用FP32进行模型训练，因此模型最后的weights也是FP32格式。但是一旦完成训练，所有的网络参数就已经是最优的，在推理过程中无需进行反向迭代，因此可以在推理中使用FP16或者INT8精度计算从而获得更小的模型。低的显存占用率和延迟以及更高的吞吐率。

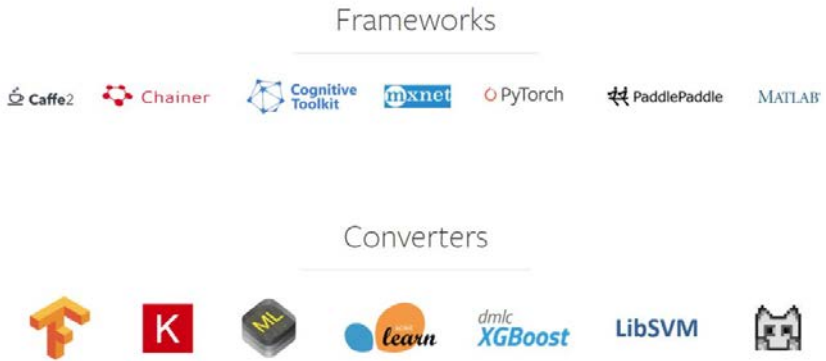
Dynamic Tensor Memory

TensorRT通过为每一个tensor在其使用期间设计分配显存来减少显存的占用，增加显存的复用率，从而避免了显存的过度开销以获得更快和更高效的推理性能。

ONNX

什么是ONNX

Open Neural Network Exchange（ONNX，开放神经网络交换）格式，是一个用于表示深度学习模型的标准，可使模型在不同框架之间进行转移。ONNX是一种针对机器学习所设计的开放式的文件格式，用于存储训练好的模型。它使得不同的人工智能框架（如Pytorch，MXNet）可以采用相同格式存储模型数据并交互。ONNX的规范及代码主要由微软，亚马逊 ，Facebook 和 IBM 等公司共同开发，以开放源代码的方式托管在Github上。目前官方支持加载ONNX模型并进行推理的深度学习框架有：Caffe2，PyTorch，MXNet，ML.NET，TensorRT 和 Microsoft CNTK，并且 TensorFlow 也非官方的支持ONNX



ONNX

典型的几个利用ONNX部署的路线：

- Pytorch -> ONNX -> TensorRT
- Pytorch -> ONNX -> TVM
- TensorFlow -> ONNX -> ncnn
- Pytorch -> ONNX -> TensorFlow

什么是Protobuf

ONNX既然是一个文件格式，那么我们就需要一些规则去读取它，或者写入它，ONNX采用的是protobuf这个序列化数据结构协议去存储神经网络权重信息。

Protobuf是个什么东西，如果大家使用过caffe或者caffe2，那么想必可能对Protobuf比较熟悉，因为caffe的模型采用的存储数据结构协议也是Protobuf。

这里简单介绍一些protobuf吧，Protobuf是一种平台无关、语言无关、可扩展且轻便高效的序列化数据结构的协议，可以用于网络通信和数据存储。我们可以通过protobuf自己设计一种数据结构的协议，然后使用各种语言去读取或者写入，通常我们采用的语言就是C++。

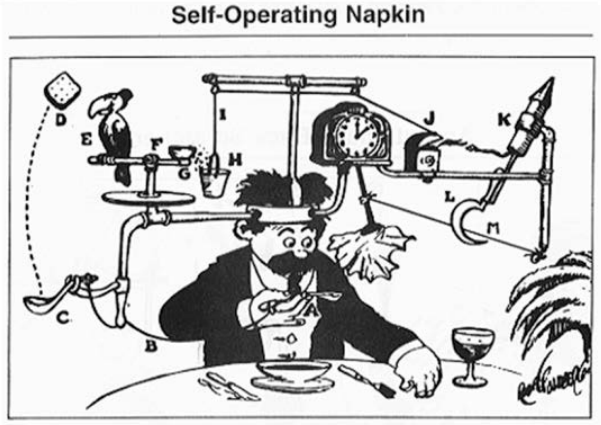
ONNX的数据格式内容

ONNX中最核心的就是 `onnx.proto` 这个文件，这个文件中定义了ONNX这个数据协议的规则和一些其他信息。

深度学习框架简介

深度学习框架就是使用**最精简的设计，灵活地**支持各类**神经网络，**并且不同硬件条件下，也能让**神经网络高效执行**

Rube Goldberg's Machine



Rube Goldberg's Machine将模型复杂化，而**深度学习框架**的目的是用最精简的方完成任务

深度学习框架简介

发展历程

- 早期率先推出的深度学习框架，应该得算是theano和Caffe, 大体的时间点如下

- 其中框架的整合或封装有如下实现方式：
  - caffe/caffe2/pytorch主要由Facebook负责开发

深度学习框架简介

发展历程

- 而早期的theano也基本转入形成tensorflow，并有Google主要负责开发

- 基于pytorch1.0进行高级封装诞生了FastAI，tensorflow进行封装成高级API则有了Keras

深度学习框架简介

发展趋势

- **支持并行计算以提高训练速度**，例如支持单机多卡/多机多卡等训练方式
- 网络优化减枝以减小训练耗时的同时，**提升底层计算硬件单元的计算能力也同等重要**，为此英伟达则推出了各种型号的GPU
- 分析现有的AI框架，大部分框架都有两方面同步提高的趋势，如百度的paddel paddel深度学习平台与生产的硬件进行结合；**在这方面而言，去年3月底华为推出的MindSpore做得更为到位一些**，Mindspore具有自动并行（auto-parallel)的特性，并与自研的达芬奇计算芯片相结合（进行芯片底层运算优化，提高计算速度）

深度学习框架简介

谷歌的TensorFlow可以说是当今最受欢迎的开源深度学习框架，可用于各类深度学习相关的任务中。TensorFlow = Tensor + Flow，**Tensor就是张量，代表N维数组；Flow即流，代表基于数据流图的计算。**

TensorFlow是目前深度学习的主流框架，其主要特性如下所述。

- TensorFlow支持Python、JavaScript、C++、Java、Go、C#、Julia和R等多种编程语言。
- TensorFlow不仅拥有强大的计算集群，还可以在iOS和Android等移动平台上运行模型。
- TensorFlow编程入门难度较大。初学者需要仔细考虑神经网络的架构，正确评估输入和输出数据的维度和数量。
- TensorFlow使用静态计算图进行操作。也就是说，我们需要先定义图形，然后运行计算，如果我们需要对架构进行更改，则需要重新训练模型。选择这样的方法是为了提高效率，但是许多现代神经网络工具已经能够在学习过程中改进，并且不会显著降低学习速度。在这方面，TensorFlow的主要竞争对手是PyTorch。

Keras是一个对小白用户非常友好且简单的深度学习框架。**如果想快速入门深度学习， Keras将是不错的选择。**

Keras是TensorFlow高级集成API，可以非常方便地和TensorFlow进行融合。Keras在高层可以调用TensorFlow、CNTK、Theano，还有更多优秀的库也在被陆续支持中。Keras的特点是能够快速搭建模型，是高效地进行科学研究的关键。

Keras的基本特性如下：

- 高度模块化，搭建网络非常简洁；
- API简单，具有统一的风格；
- 易扩展，易于添加新模块，只需要仿照现有模块编写新的类或函数即可。

深度学习框架简介

Caffe

Caffe是由AI科学家贾扬清在加州大学伯克利分校读博期间主导开发的，是以C++/CUDA代码为主的早期深度学习框架之一，比TensorFlow、MXNet、PyTorch等都要早。Caffe需要进行编译安装，支持命令行、Python和Matlab接口，单机多卡、多机多卡等都可以很方便使用。

Caffe的基本特性如下。

- 以C++/CUDA/Python代码为主，速度快，性能高。
- 工厂设计模式，代码结构清晰，可读性和可拓展性强。
- 支持命令行、Python和Matlab接口，使用方便。
- CPU和GPU之间切换方便，多GPU训练方便。
- 工具丰富，社区活跃。

同时，Caffe的缺点也比较明显，主要包括如下几点。

- 源代码修改门槛较高，需要实现正向/反向传播。
- 不支持自动求导。
- 不支持模型级并行，只支持数据级并行。
- 不适合非图像任务。

CNTK

CNTK（Microsoft Cognitive Toolkit）是微软开源的深度学习工具包，它通过有向图将神经网络描述为一系列计算步骤。在有向图中，叶节点表示输入值或网络参数，其他节点表示其输入上的矩阵运算。

CNTK允许用户非常轻松地实现和组合流行的模型，包括前馈神经网络（DNN）、卷积神经网络（CNN）和循环神经网络（RNN、LSTM）。与目前大部分框架一样，CNTK实现了自动求导，利用随机梯度下降方法进行优化。

CNTK的基本特性如下。

- CNTK性能较好，按照其官方的说法，它比其他的开源框架性能都要好。
- 适合做语音任务，CNTK本就是微软语音团队开源的，自然更适合做语音任务，便于在使用RNN等模型以及时空尺度时进行卷积。

深度学习框架简介

PyTorch是Facebook团队于2017年1月发布的一个深度学习框架，虽然晚于TensorFlow、Keras等框架，**但自发布之日起，其受到的关注度就在不断上升**，目前在GitHub上的热度已经超过Theano、Caffe、MXNet等框架。

PyTroch主要提供以下两种核心功能：

- 支持GPU加速的张量计算；
- 方便优化模型的自动微分机制。

PyTorch的主要优点如下。

- **简洁易懂：**PyTorch的API设计相当简洁一致，基本上是tensor、autograd、nn三级封装，学习起来非常容易。
- **便于调试：**PyTorch采用动态图，可以像普通Python代码一样进行调试。不同于TensorFlow，PyTorch的报错说明通常很容易看懂。
- **强大高效：**PyTorch提供了非常丰富的模型组件，可以快速实现想法。

MXNet框架允许混合符号和命令式编程，以最大限度地提高效率和生产力。MXNet的核心是一个动态依赖调度程序，可以动态地自动并行化符号和命令操作。其图形优化层使符号执行更快，内存效率更高。

MXNet的基本特性如下。

- **灵活的编程模型：**支持命令式和符号式编程模型。
- **多语言支持：**支持C++、Python、R、Julia、JavaScript、Scala、Go、Perl等。事实上，它是唯一支持所有R函数的构架。
- **本地分布式训练：**支持在多CPU/GPU设备上的分布式训练，使其可充分利用云计算的规模优势。
- **性能优化：**使用一个优化的C++后端引擎实现并行I/O和计算，无论使用哪种语言都能达到最佳性能。
- **云端友好：**可直接与S3、HDFS和Azure兼容。

深度学习框架简介

ONNX

ONNX（Open Neural Network eXchange，开放神经网络交换）项目由微软、亚马逊、Facebook和IBM等公司共同开发，旨在寻找呈现开放格式的深度学习模型。ONNX简化了在人工智能不同工作方式之间传递模型的过程，具有各种深度学习框架的优点。

ONNX的基本特性如下。

- ONNX使模型能够在一个框架中进行训练并转移到另一个框架中进行预测。
- ONNX模型目前在Caffe2、CNTK、MXNet和PyTorch中得到支持，并且还有与其他常见框架和库的连接器。

Darknet是一个较为轻型的完全基于C与CUDA的开源深度学习框架，其主要特点就是容易安装，没有任何依赖项（OpenCV都可以不用），移植性非常好，支持CPU与GPU两种计算方式。

相比于TensorFlow来说，darknet并没有那么强大，但这也成了darknet的优势：

- darknet完全由C语言实现，没有任何依赖项，当然可以使用OpenCV，但只是用其来显示图片、为了更好的可视化；
- darknet支持CPU（所以没有GPU也不用紧的）与GPU（CUDA/cuDNN，使用GPU当然更块更好了）；
- 轻量，灵活、没有像TensorFlow那般强大的API，适合用来研究底层，可以更为方便的从底层对其进行改进与扩展；
- darknet的实现与caffe的实现存在相似的地方，熟悉了darknet，相信对上手caffe有帮助；

深度学习框架简介

2016年8月底百度开源了内部使用多年的深度学习平台PaddlePaddle，PaddlePaddle 100% 都在Github上公开，没有内部版本。PaddlePaddle能够应用于自然语言处理、图像识别、推荐引擎等多个领域，其优势在于开放的多个领先的预训练中文模型。PaddlePaddle的2013年版本是百度杰出科学家徐伟主导设计和开发的，其设计思路是每一个模型的表示方式都是“一串Layers”，Caffe的作者贾扬清称赞了百度的 PaddlePaddle，并说“整体的设计感觉和 Caffe 心有灵犀”。三年后，百度AI团队在徐伟的指导下作了两次升级，2017年4月推出PaddlePaddle v2，v2参考TensorFlow增加了Operators的概念，把Layers打碎成更细粒度的Operators，同时支持更复杂的网络拓扑图而不只是“串”。2017 年底推出PaddlePaddleFluid。Fluid类似PyTorch，提供自己的解释器甚至编译器，所以不受限于 Python 的执行速度问题。

2018年10月10日，华为在上海全联接大会上首次发布华为AI战略与全栈全场景AI解决方案，包括Ascend(昇腾)系列AI芯片以及CANN算子库、MindSpore深度学习框架、AI开发平台ModelArts。华为MindSpore支持端、边、云独立的和协同的统一训练和推理框架。但是目前仍然在开发中，以华为在中国科技界地位和研发投入，自然是最受大家期待的。华为云虽然可以支持其它所有主流的深度学习框架，但就如同Amazon选择MXNet一样，这不是一个可以讨论的问题，为了不受制于人，是一定要有的。我相信为了与其它主流框架进行竞争，MindSpore将来也一定会开源的。

这么多框架，用哪个好？

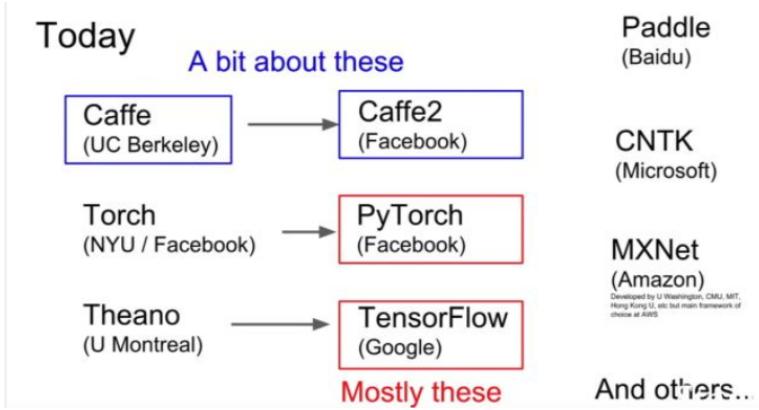
各大开源框架的一个总览

框架	发布时间	维护组织	底层语言	接口语言	Git star
Caffe	2013/9	BVLC	C++	C++/Python/Matlab	27000+
Tensorflow	2015/9	Google	C++/Python	C++/Python/Java等	124000+
Pytorch	2017/1	Facebook	C/C++/Python	Python	26000+
Mxnet	2015/5	DMLC	C++	C++/Python/Julia/R等	16000+
Keras	2015/3	Google	Python	Python	39600+
Paddlepaddle	2016/8	Baidu	C++/Python	C++/Python	8300+
Cntk	2014/7	Microsoft	C++	C++/Python/C#/.NET/Java	15900+
Matconvnet	2014/2	VLFeat	C/Matlab	Matlab	1100+
Deeplearning4j	2013/9	Eclipse	C/C++/Cuda	Java/Scalar等	10000+
Chainer	2015/4	Preferred networks	Python/Cython	Python	4600+
Lasagne/theano	2014/9	Lasagne	C/Python	Python	3600+
Darknet	2013/9	Joseph Redmon	C	C	12000+

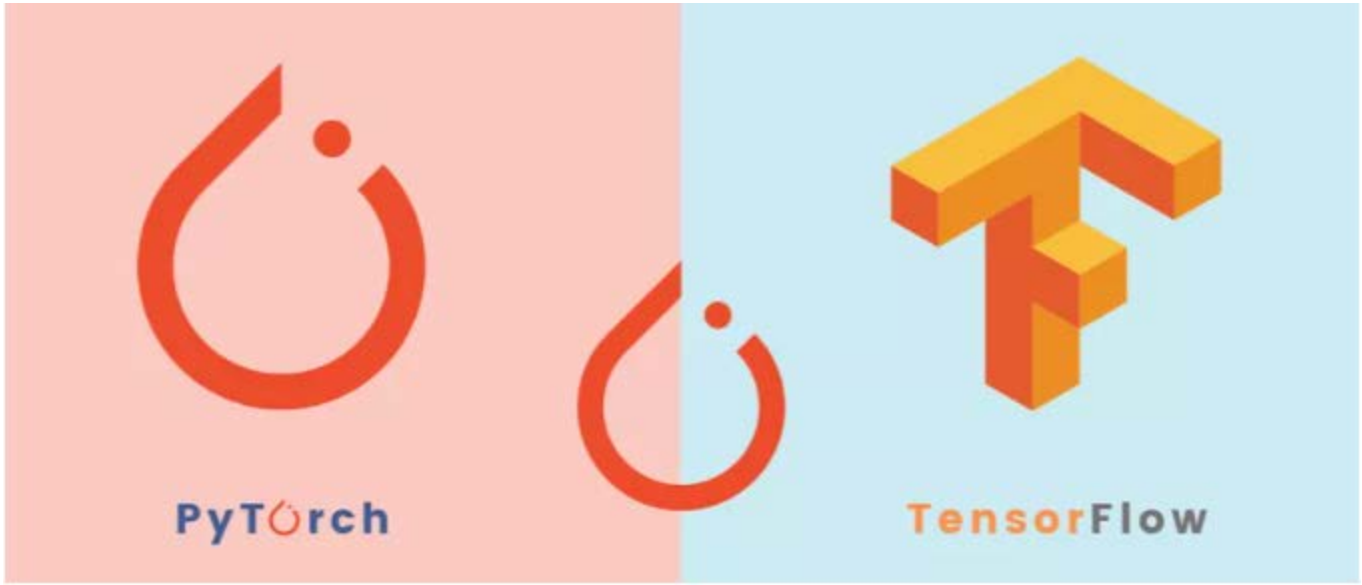
作为AI领域专业从业人员

1.不管怎么说，tensorflow/pytorch 你都必须会，是目前开发者最喜欢，开源项目最丰富的框架。
2.如果你要进行移动端算法的开发，那么 Caffe 是不能不会的。
3.如果你非常熟悉 Matlab，matconvnet 你不应该错过。
4.如果你追求高效轻量，那么 darknet 和 mxnet 你不能不熟悉。
5. 如果你很懒，想写最少的代码完成任务，那么用 keras 吧。
6.如果你是 java 程序员，那么掌握 deeplearning4j 没错的。

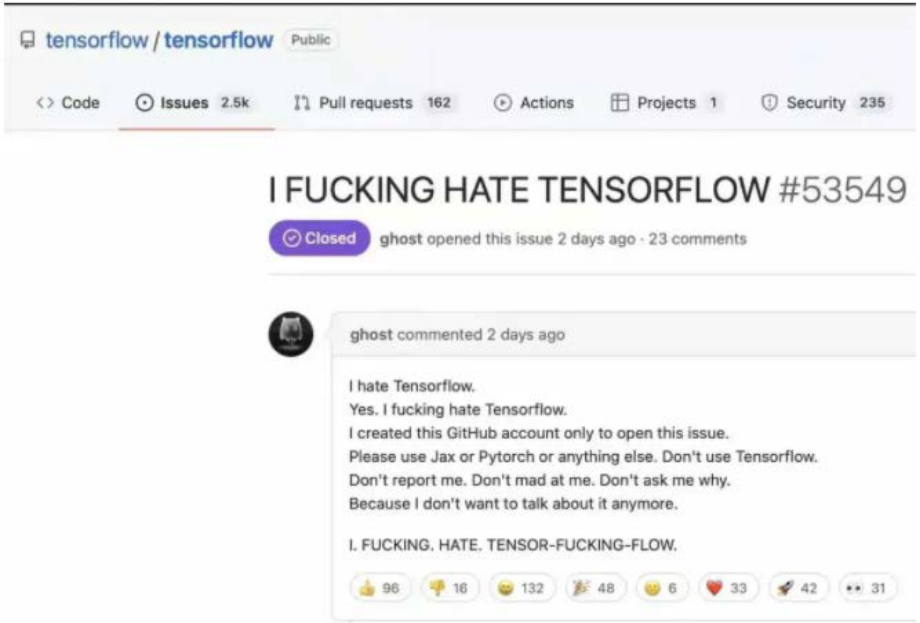
- 如果你相去百度工作，用Paddle
- Amazon：MXNet
- 谷歌：TensorFlow
- Facebook：PyTorch or Caffe
- 或许会增加面试官对你的一丢丢好感，见下图（2017年版本）



Pytorch vs TensorFlow



Pytorch vs TensorFlow



Pytorch vs TensorFlow

早在2015年11月9日，TensorFlow依据阿帕奇授权协议（Apache 2.0 open source license）就开放了源代码，其前身是谷歌的神经网络算法库DistBelief。TensorFlow是一个基于数据流编程（dataflow programming）的符号数学系统，被广泛应用于各类机器学习（machine learning）算法的编程实现。

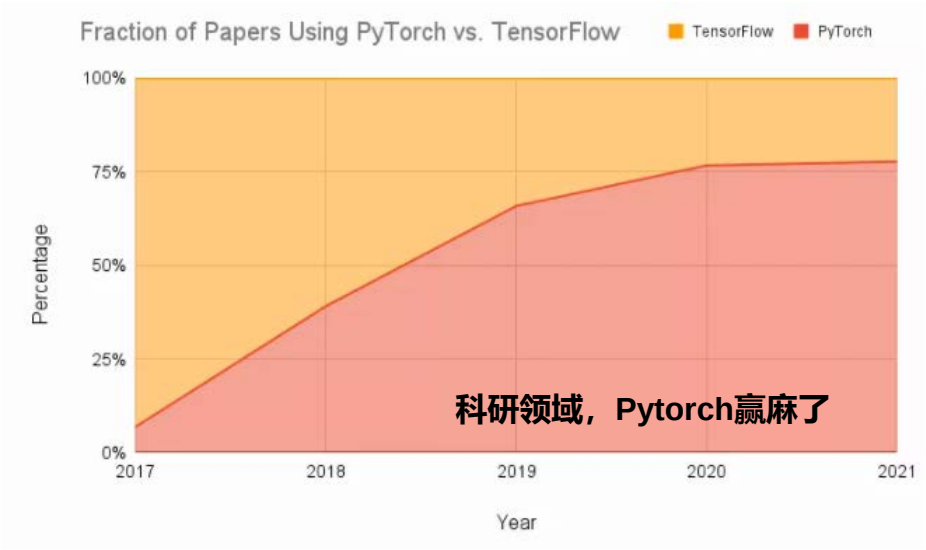
到了2017年1月，由Facebook人工智能研究院（FAIR，现在是MAIR）基于Torch推出了PyTorch，主要提供两个高级功能：

1. 具有强大的GPU加速的张量计算（如NumPy）
2. 包含自动求导系统的深度神经网络

Pytorch vs TensorFlow

模型可用性--源起

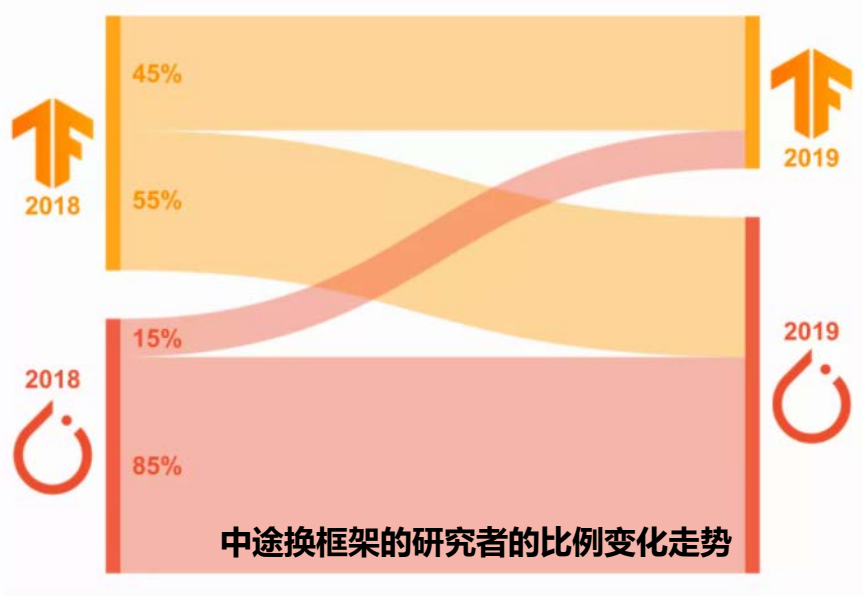
PyTorch 和 TensorFlow 分歧很大，PyTorch 和 TensorFlow 都有自己的官方模型存储库，但从业者可能希望利用多个来源的模型

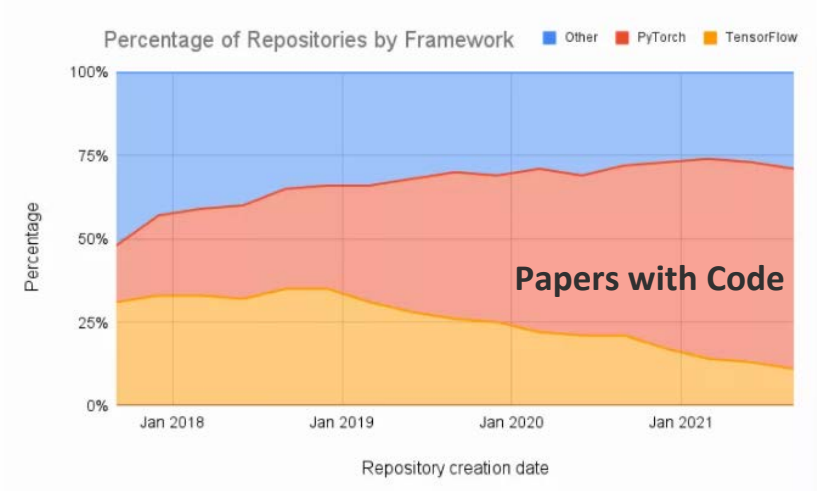


Pytorch vs TensorFlow

模型可用性--发展

很多转向 PyTorch 的研究者都表示 TensorFlow 1 太难用了。尽管 2019 年发布的 TensorFlow 2 改掉了一些问题，但彼时，PyTorch 的增长势头已经难以遏制。





Pytorch vs TensorFlow

模型可用性--小结

PyTorch 目前在**研究领域**占据主导地位。虽然 TensorFlow 2 解决了研究者使用该框架进行研究的一些痛点，但 PyTorch 却没有给研究者回头的理由。此外，TensorFlow 两大版本之间的**向后兼容性问题**只会让这种趋势愈演愈烈。

Google AI：谷歌发布的论文自然会用 TensorFlow。鉴于在论文方面谷歌比 Facebook 更高产，一些研究者可能会发现掌握 TensorFlow 还是很有用的。

DeepMind：DeepMind 也用 TensorFlow，而且也比 Facebook 高产。他们创建了一个名叫 Sonnet 的 TensorFlow 高级 API，用于研究目的。有人管这个 API 叫「科研版 Keras」，那些考虑用 TensorFlow 做研究的人可能会用到它。此外，DeepMind 的 Acme 框架可能对于强化学习研究者很有用。

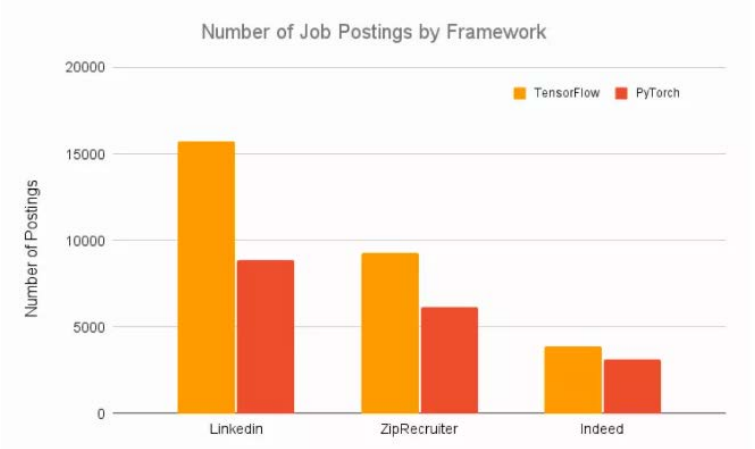
OpenAI：OpenAI 在 2020 年宣布了全面拥抱 PyTorch 的决定。但他们之前的强化学习基线库都是在 TensorFlow 上部署的。基线提供了高质量强化学习算法的实现，因此 TensorFlow 可能还是强化学习从业者的最佳选择。

Pytorch vs TensorFlow

部署便捷性--开端

TensorFlow 自成立以来一直是面向部署的应用程序的首选框架，TensorFlow Serving和TensorFlow Lite可让用户轻松地在云、服务器、移动设备和 IoT 设备上进行部署

以前PyTorch 用户需要使用 Flask 或 Django 在模型之上构建一个 REST API



Pytorch vs TensorFlow

部署便捷性--发展

TorchServe：
TorchServe 是 AWS 和 Facebook 合作的开源部署框架，于 2020 年发布。它具有端点规范、模型归档和指标观测等基本功能，但仍然不如 TensorFlow。TorchServe 同时支持 REST 和 gRPC API。

PyTorch Live：
PyTorch 于 2019 年首次发布 PyTorch Mobile，旨在为部署优化的机器学习模型创建端到端工作流，适用于 Android、iOS 和 Linux。
PyTorch Live 于 12 月初发布，以移动平台为基础。它使用 JavaScript 和 React Native 来创建带有相关 UI 的跨平台 iOS 和 Android AI 应用。设备上的推理仍然由 PyTorch Mobile 执行。

Pytorch vs TensorFlow

部署便捷性--小结

目前，TensorFlow 依然在部署方面占有优势。Serving 和 TFLite 比 PyTorch 的同类型工具要稳健一些。而且，将 TFLite 与谷歌的 Coral 设备一起用于本地 AI 的能力是许多行业的必备条件。相比之下，PyTorch Live 只专注于移动平台，而 TorchServe 仍处于起步阶段。因此综合来看，第二轮（部署便捷性）的胜出者是 TensorFlow。

如果你既想用 TensorFlow 的部署基础设施，又想访问只能在 PyTorch 中使用的模型，作者推荐使用 ONNX 将模型从 PyTorch 移植到 TensorFlow。

Pytorch vs TensorFlow

生态系统--两者都做得很好

TensorFlow

TensorFlow Hub：一个经过训练的机器学习模型库，包含 TensorFlow、TensorFlow Lite 和 TensorFlow.js 模型；

TensorFlow Extended： TensorFlow 用于模型部署的端到端平台，用户可以加载、验证、分析和转换数据；训练和评估模型；

MediaPipe：用于构建多模式、跨平台应用机器学习管道的框架，可用于人脸检测、多手跟踪、对象检测等。开源，多种语言，包括 Python、C++ 和 JavaScript。

TensorFlow.js：一个用于机器学习的JavaScript库，允许使用 Node.js 在浏览器和服务器端训练和部署模型。

TensorFlow Cloud：可让用户将本地环境连接到Google Cloud。

Google Colab：一个基于云的笔记本环境，与 Jupyter 非常相似。

Pytorch

PyTorch Hub：包括用于音频、视觉和 NLP 的模型。它还有一些生成模型，包括用于生成名人面孔的高质量图像的GAN；

SpeechBrain：PyTorch 的官方开源语音工具包，支持 ASR、说话人识别、验证和分类等；

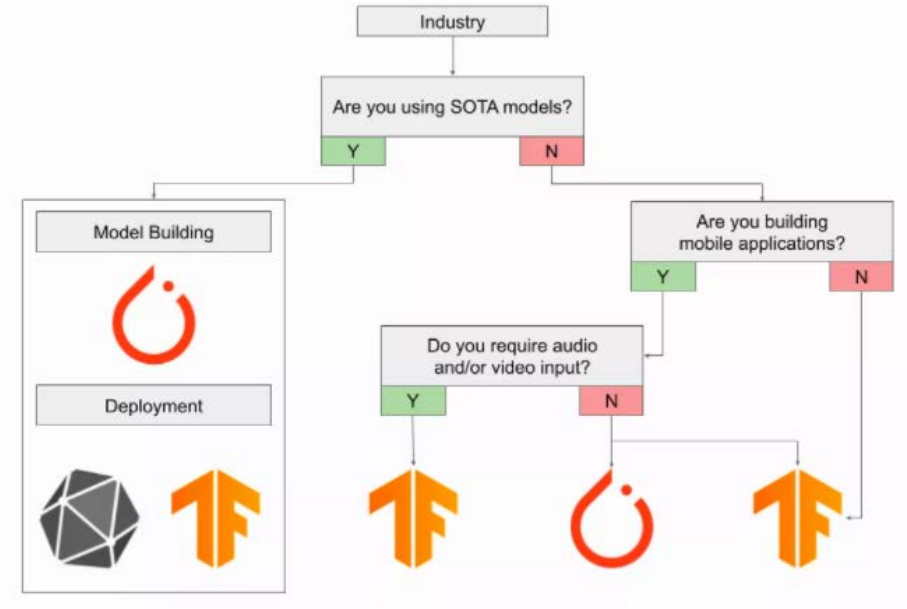
PyTorch Lightning： PyTorch 的 Keras，可以简化 PyTorch 中模型工程和训练过程；

TorchElastic：分布式训练工具，可管理工作进程并协调重启行为，以便用户可以在计算节点集群上训练模型；

TorchX：用于快速构建和部署机器学习应用程序的 SDK

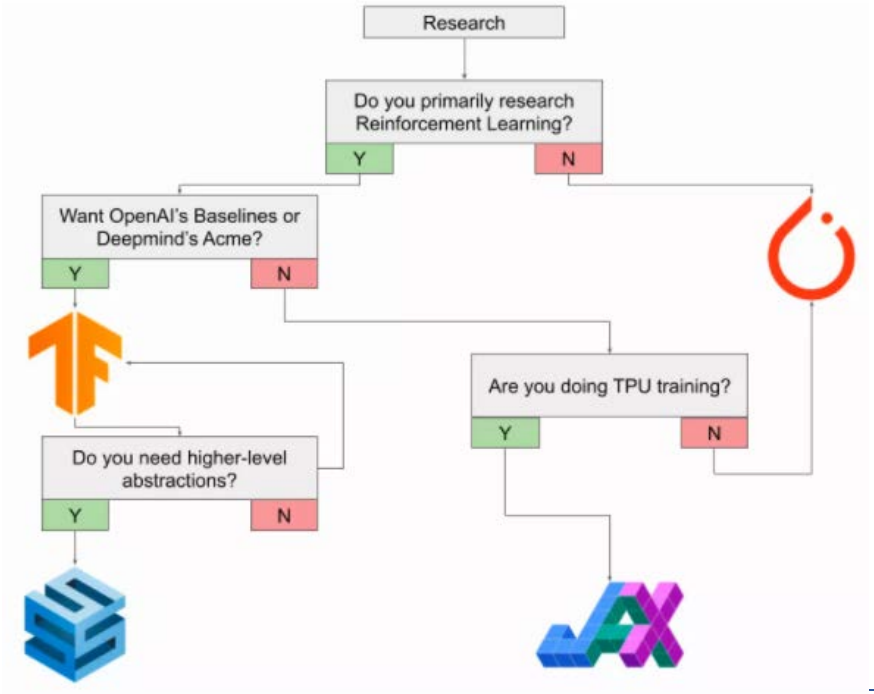
Pytorch or TensorFlow?

如果你在工业界应该怎么选?



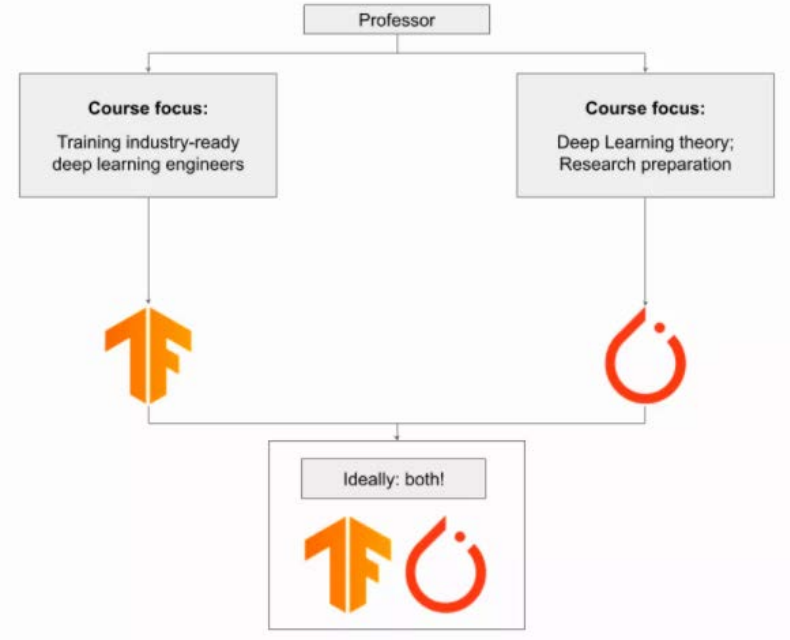
Pytorch or TensorFlow?

如果你是研究者应该怎么选?



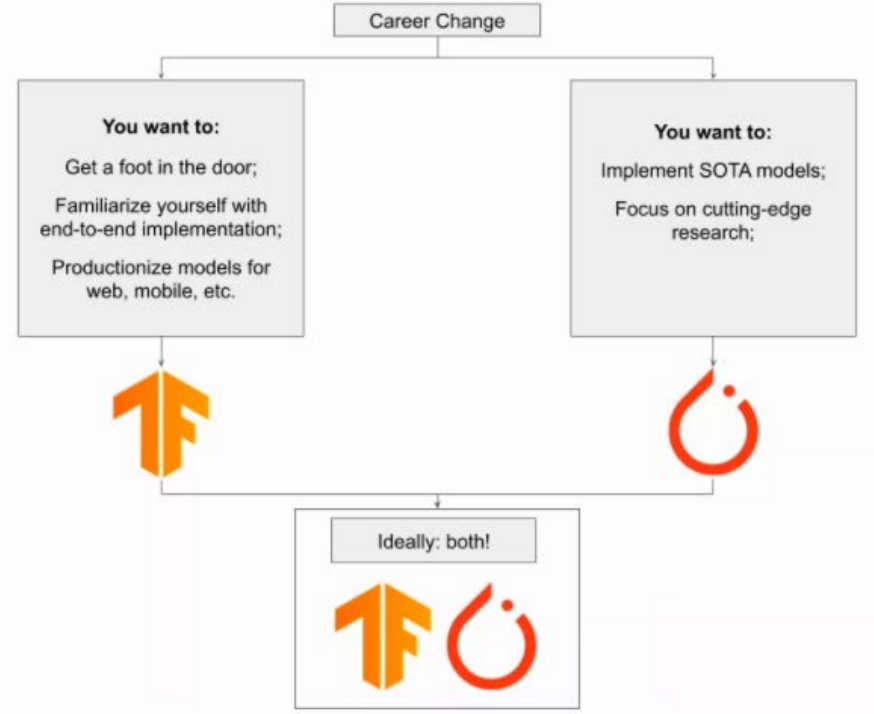
Pytorch or TensorFlow?

如果你是一名教授应该怎么选?



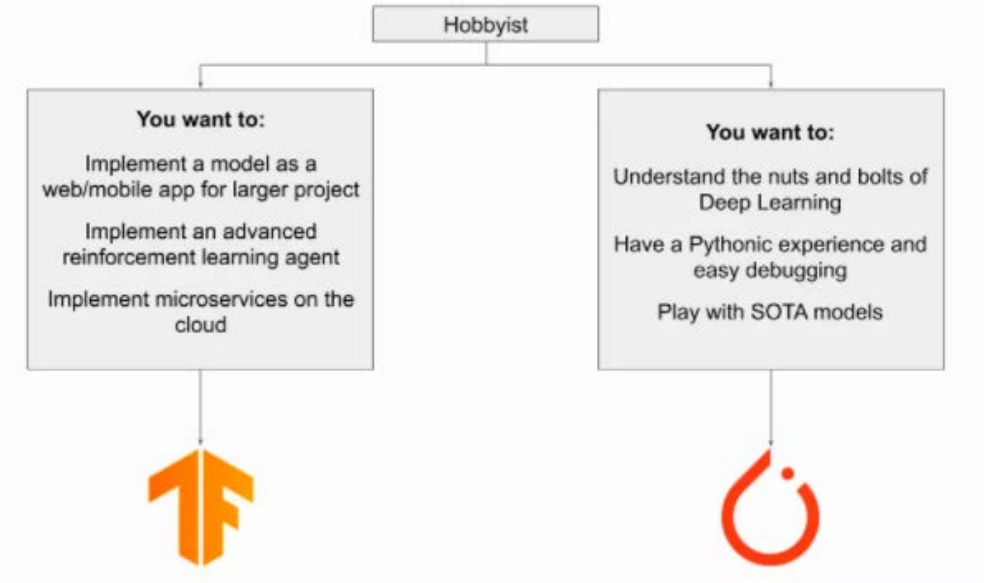
Pytorch or TensorFlow?

如果你正在寻求职业上的转变，应该怎么选?



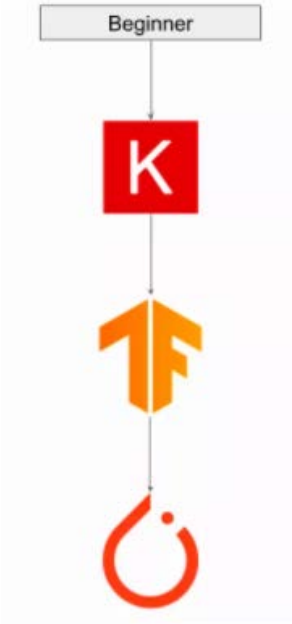
Pytorch or TensorFlow?

如果你是业余爱好者，应该怎么选?

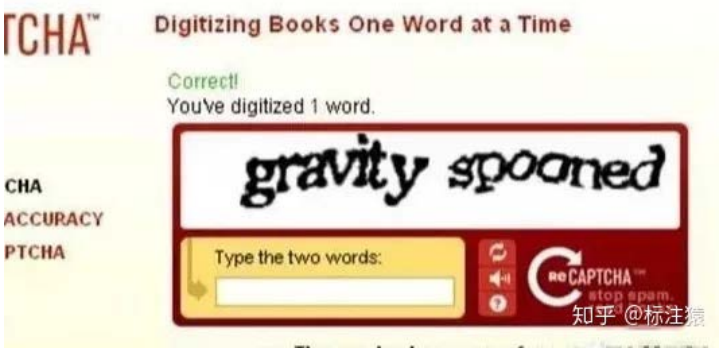


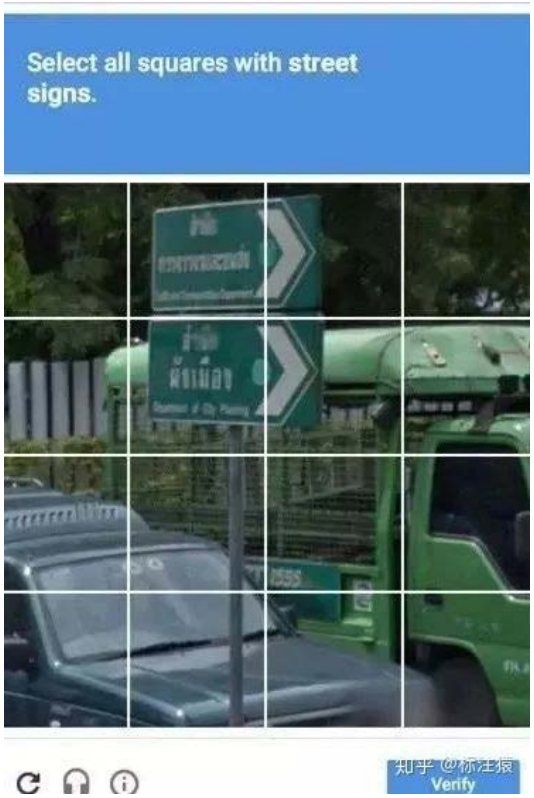
Pytorch or TensorFlow?

如果你初学者，应该怎么选?



数据标注工具


TCHA™
Digitizing Books One Word at a Time
Correct!
You've digitized 1 word.
CHA
ACCURACY
PTCHA
gravity spooned
Type the two words:
reCAPTCHA™
stop spam.
知乎 @标注猿


Select all squares with street signs.
知乎 @标注猿
Verify

数据标注工具

1.LabelImg

主页地址:https://github.com/tzutalin/labelImg

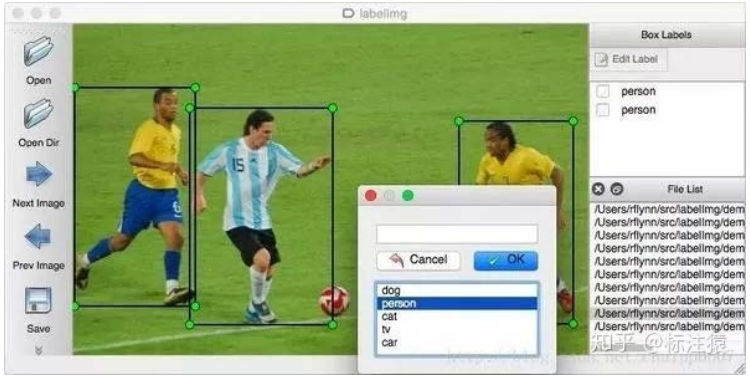
下载以后根据作者提供的安装指南即可安装。

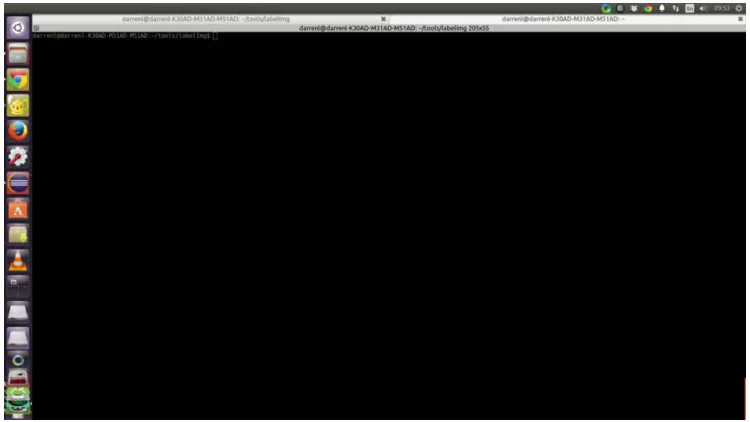
如果安装不上怎么办，不用这么麻烦，下面这个地址提供了直接下载的地址，下载预编译exe即可：

https://github.com/zhaobai62/labelImg

支持VOC2012与TFRecord文件自动生成

同时支持YOLO、CreatML等格式





数据标注工具

2.Labelme

主页地址:https://github.com/wkentaro/labelme

支持对象检测、图像语义分割数据标注，实现语言为Python与QT。

支持导出VOC与COCO格式数据实例分割

做**实例分割、语义分割**等任务比较推荐此工具。

数据标注工具

3.OpenCV/CVAT

官方主页:https://github.com/opencv/cvat

高效的计算机视觉注释工具，支持图像分类，对象检测框，图像语义分割

4.VOTT

官方主页:https://github.com/microsoft/VoTT

Microsoft发布的基于WEB的可视化数据注释工具，可本地部署

支持图像和视频数据注释

数据标注工具

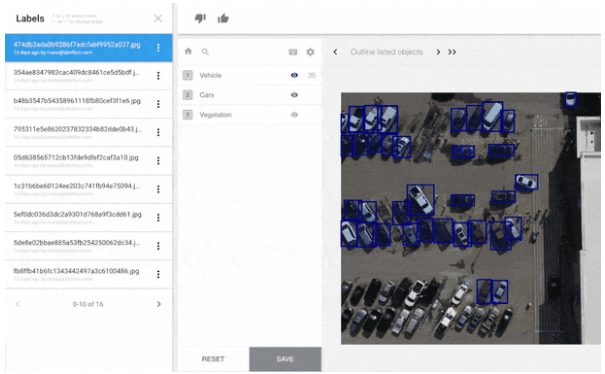
5.LabelBox

官方主页:https://github.com/Labelbox/Labelbox

WEB模式下的标记工具

提供自定义注释API支持

纯JS + HTML支持



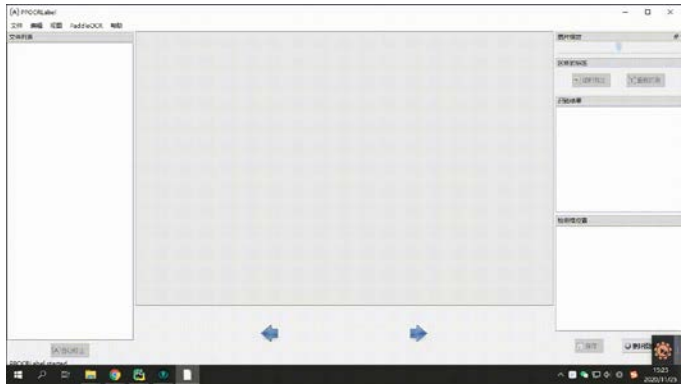
6.PPOCRLabel

官方主页: https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/PPOCRLabel/README_ch.md

适用于OCR领域的半自动化图形标注工具

实现语言Python3与PyQT5

支持矩形框标注和四点标注模式，导出格式可直接用于PaddleOCR检测和识别模型的训练



数据标注工具

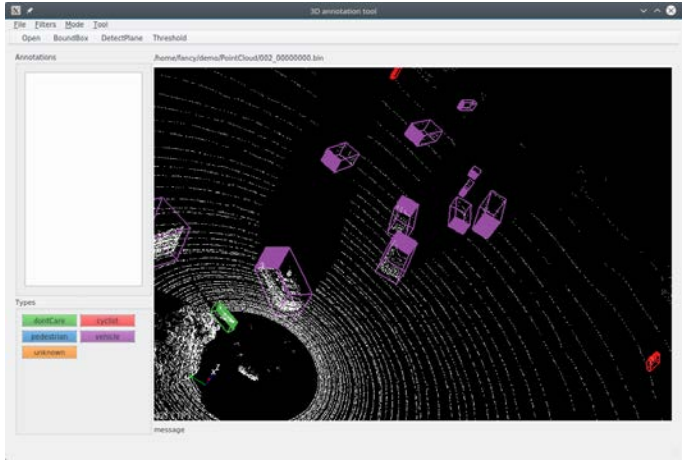
7.point-cloud-annotation-tool

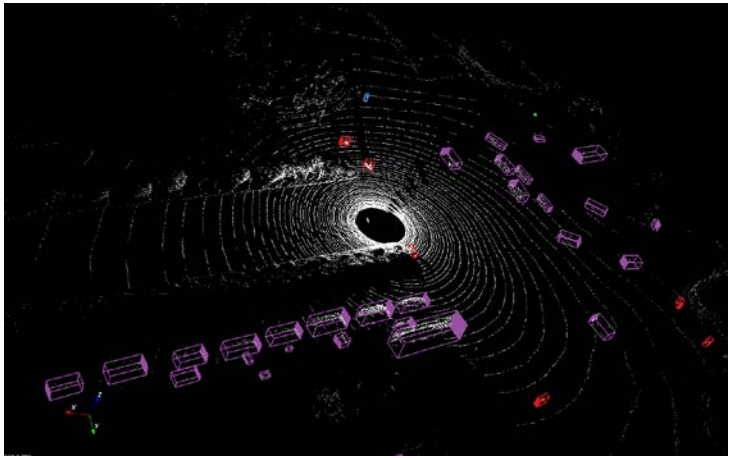
官方地址:https://github.com/springzfx/point-cloud-annotation-tool

3D点云数据注释工件

支持3D BOX盒子生成

支持KITTI-bin格式数据





数据标注工具

8. Semantic Segmentation Editor

GitHub链接：https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor

支持普通相机拍摄的2D图像（.jpg和.png文件）

支持LIDAR生成的3D点云（.pcd文件）中目标的标注

支持3D BOX框生成

数据标注工具

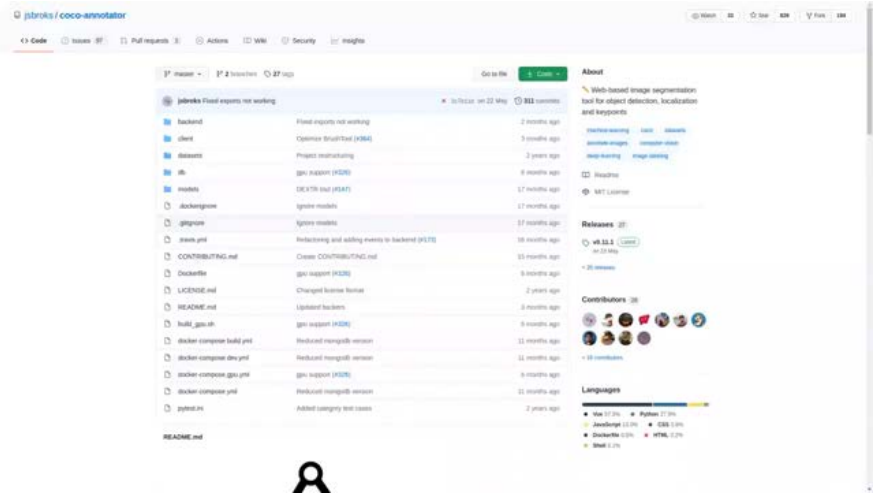
9. coco-annotator

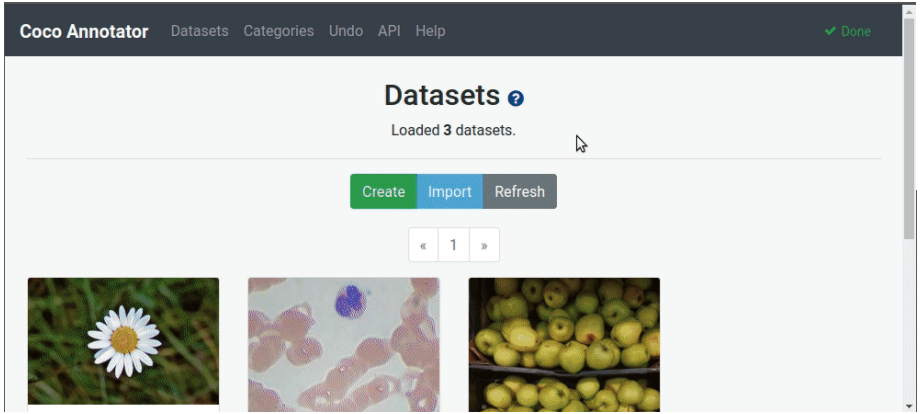
GitHub链接：https://github.com/jsbroks/coco-annotator

基于WEB的可视化数据注释工具，可本地部署

支持图像定位和目标检测，特别适用于细粒度的目标分割定位任务

导出coco格式标注数据文件





数据标注工具

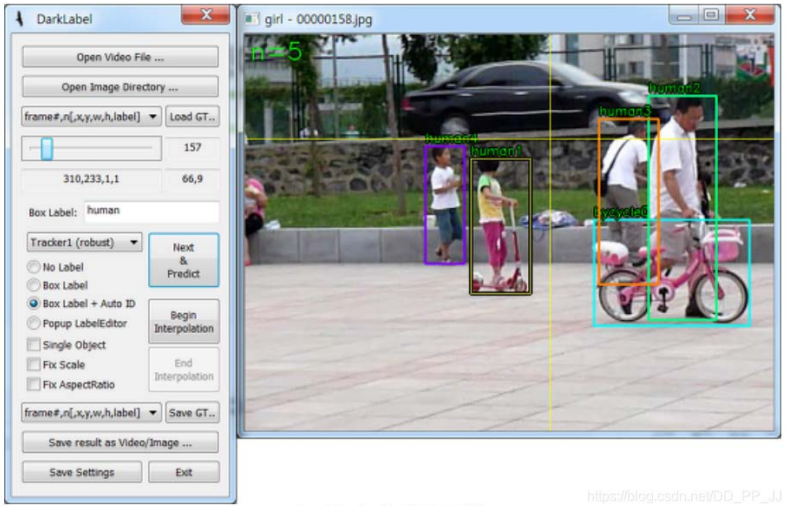
10. DarkLabel

GitHub链接：https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor

Windows平台下的轻量视频标注软件

特别适用于目标检测、多目标跟踪、ReID

导出的格式可以通过脚本转化，变成标准的**目标检测**数据集格式、**ReID数据集**格式、**MOT数据集**格式



数据标注工具

主要功能和特点

•支持各种格式的视频（avi，mpg等）和图像列表（jpg，bmp，png等）

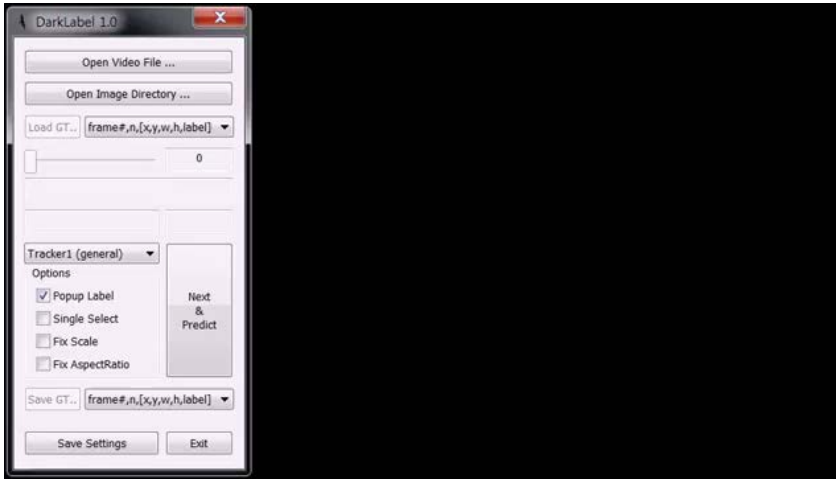
•多框设置和标签设置支持

•支持对象识别和图像跟踪中使用的各种数据格式

•使用图像跟踪器自动标记（通过跟踪标记）

•支持使用插值功能的间隔标签

•自动标记功能，可按类别自动为每个对象分配唯一的ID



追踪功能

这是这个软件比较好的功能之一，可以用传统方法（KCF类似的算法）跟踪目标，只需要对不准确的目标进行人工调整即可，大大减少了工作量。

•通过使用图像跟踪功能设置下一帧的框（分配相同的ID /标签）

•多达100个同时跟踪

•tracker1（稳健）算法：长时间跟踪目标

•tracker2（准确）算法：准确跟踪目标（例如汽车）

•输入键/下一步和预测按钮

Thanks!

Do you have any questions?

gonghao@landmarkvision.cn
+86 188 1809 9180
www.landmarkvision.cn

Please keep this slide for attribution