XDF XILINX DEVELOPER FORUM

Deep Learning for ADAS on FPGA

Presented By

毛宁元
技术副总监
2018.10.16

XILINX

The rise of deep learning in ML

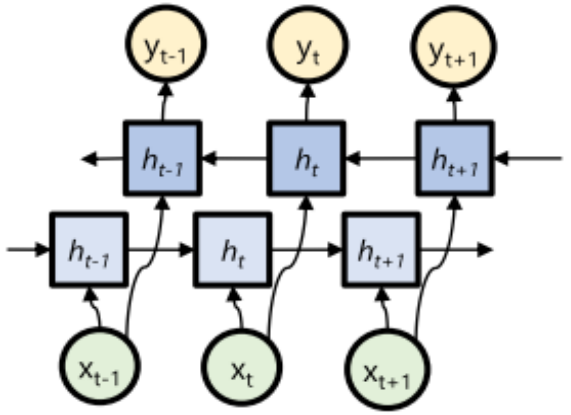
Deep neural networks have enabled major advances in machine learning and AI

Self-Driving

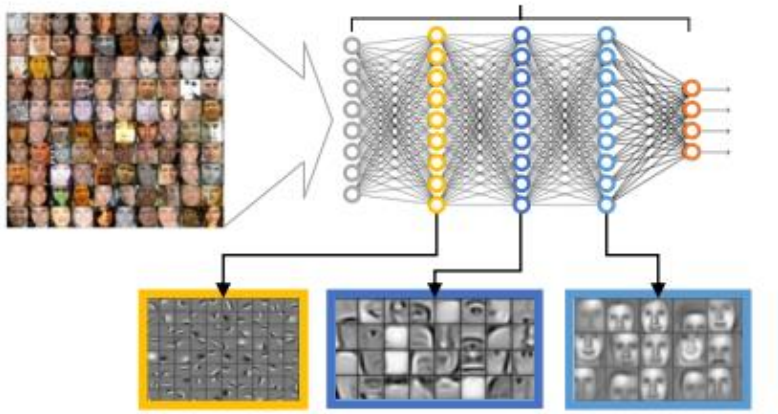
Machine Translation

Play Go

Artistic Style transfer



Recurrent Neural network

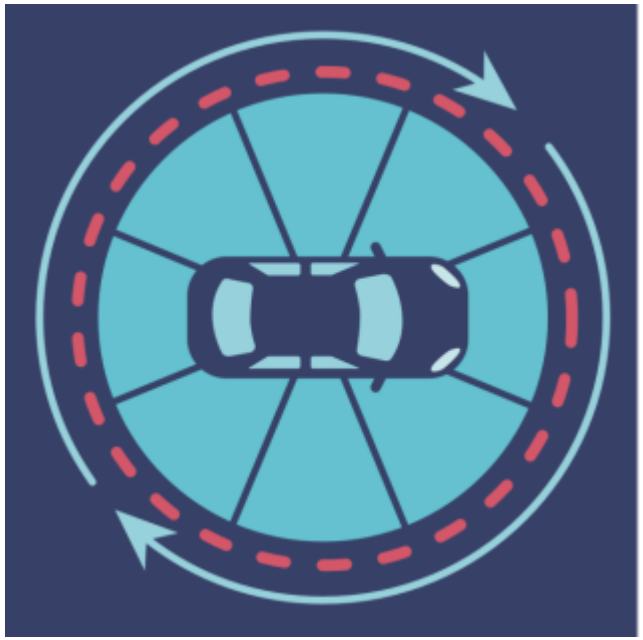


Convolutional Neural network

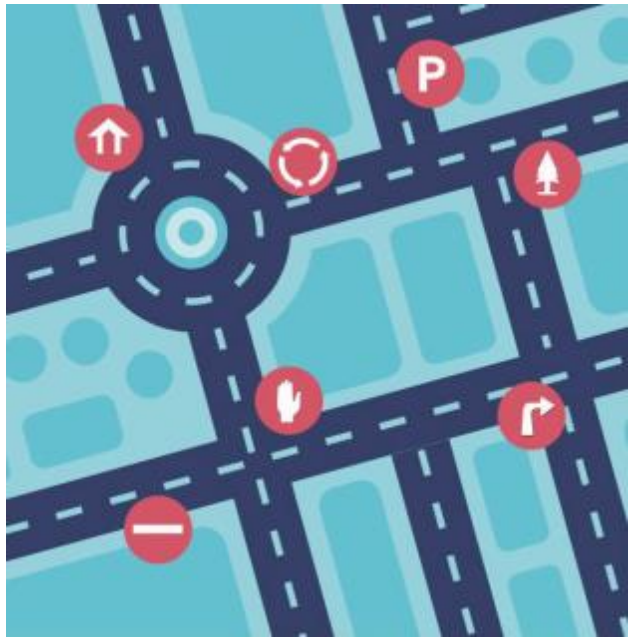
Autonomous driving：SAE/NHTSA

自动驾驶分级		名称	定义	驾驶操作	周边监控	接管	应用场景
NHTSA	SAE						
L0	L0	人工驾驶	由人类驾驶者全权驾驶汽车。	人类驾驶员	人类驾驶员	人类驾驶员	无
L1	L1	辅助驾驶	车辆对方向盘和加减速中的一项操作提供驾驶，人类驾驶员负责其余的驾驶动作。	人类驾驶员和车辆	人类驾驶员	人类驾驶员	限定场景
L2	L2	部分自动驾驶	车辆对方向盘和加减速中的多项操作提供驾驶，人类驾驶员负责其余的驾驶动作。	车辆	人类驾驶员	人类驾驶员	
L3	L3	条件自动驾驶	由车辆完成绝大部分驾驶操作，人类驾驶员需保持注意力集中以备不时之需。	车辆	车辆	人类驾驶员	
L4	L4	高度自动驾驶	由车辆完成所有驾驶操作，人类驾驶员无需保持注意力，但限定道路和环境条件。	车辆	车辆	车辆	
	L5	完全自动驾驶	由车辆完成所有驾驶操作，人类驾驶员无需保持注意力。	车辆	车辆	车辆	所有场景

Autonomous driving：Three Pillars



Sensing

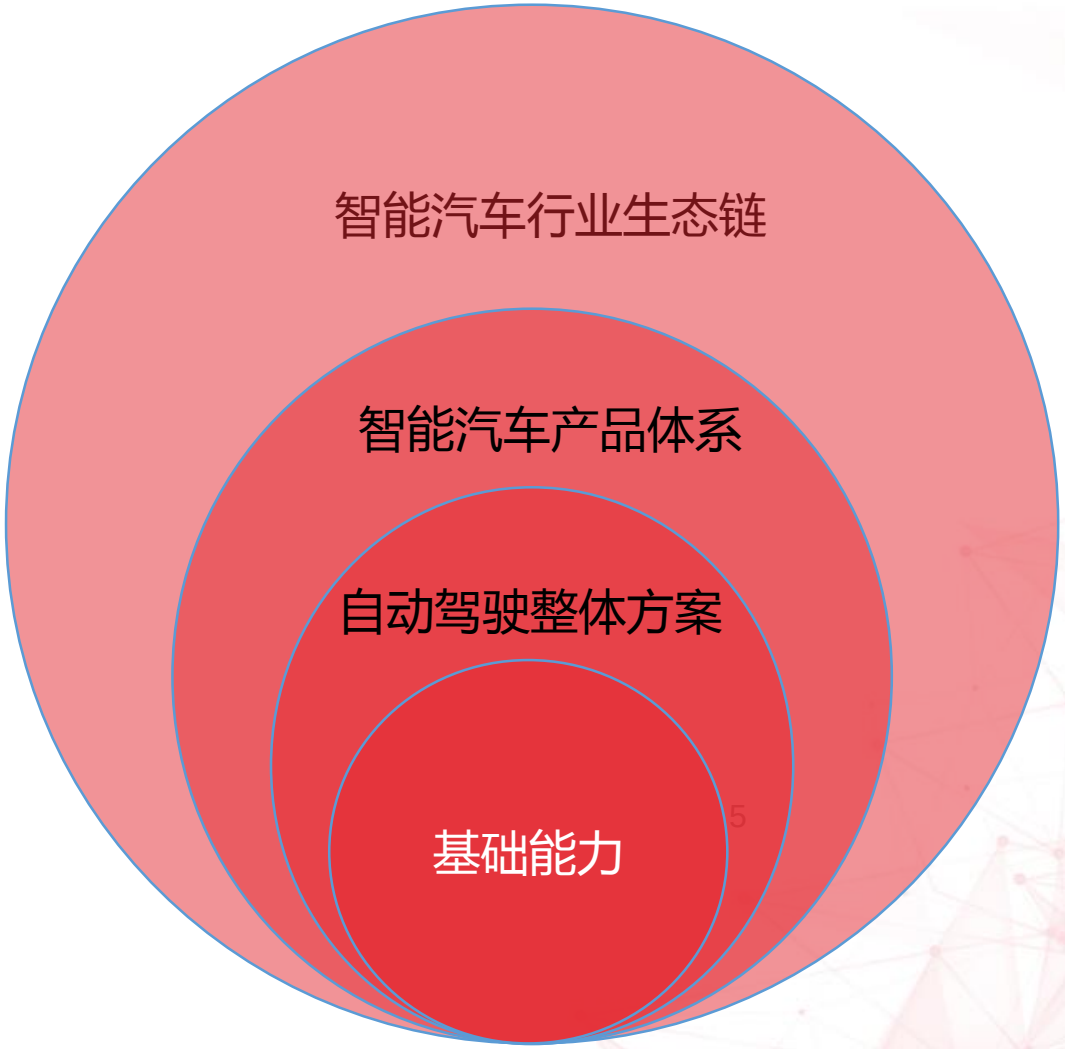


Path Planning

Driving Policy

Vision | 打造世界一流的智能汽车产品体系以及行业生态链

- **赋能车企，打造行业生态链**
  - 国际知名车企深度合作，联合开发
- **完善的产品体系**
  - DAS，ADAS，L2-L4自动驾驶产品
- **全方位的自动驾驶的解决方案**
  - 基于视觉感知以及多传感器融合
- **卓越的基础能力**
  - 全球领先的视觉算法
  - 人工智能GPU超算平台、深度学习平台



业务战略定位

人

智能驾驶舱

- 身份识别验证
- 驾驶行为检测（DAS）

- 出行服务运营企业
- 物流车队
- 保险公司

车

智能驾驶

- 高级辅助驾驶系统：车辆、行人、交通标志等识别、预警、控制
- L4\L5自动驾驶

- 整车制造企业
- Tier 1零部件商

基础设施

智能交通基础设施

- 高精度地图
- 交通大数据
- 智能交通服务

- 出行服务运营企业
- 交通管理机构

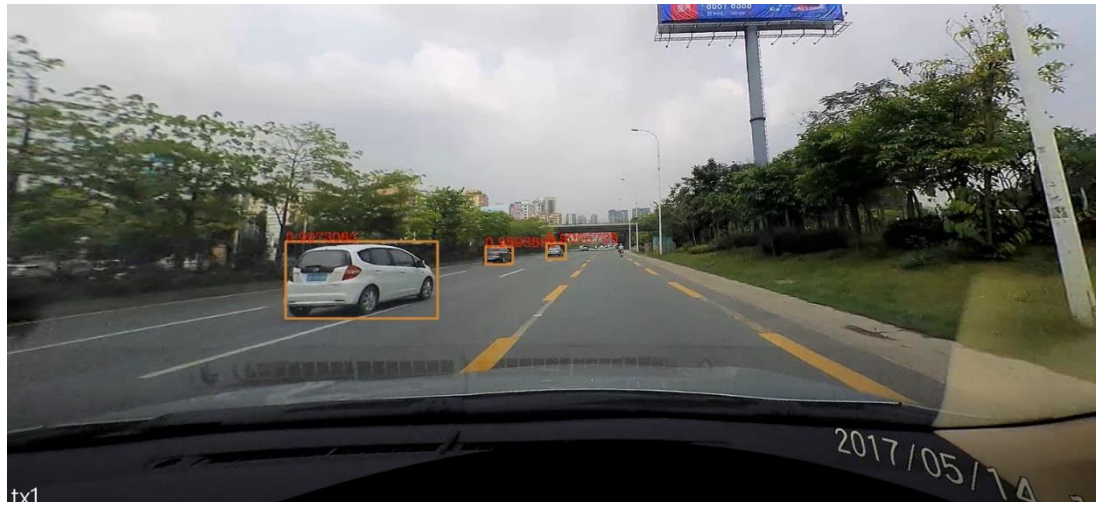
Deep learning for ADAS：LDW

Training

VGG

Res-like

Deep learning for ADAS：FCW/PCW



Vehicle detection & Pedestrian detection

Faster RCNN + Res like

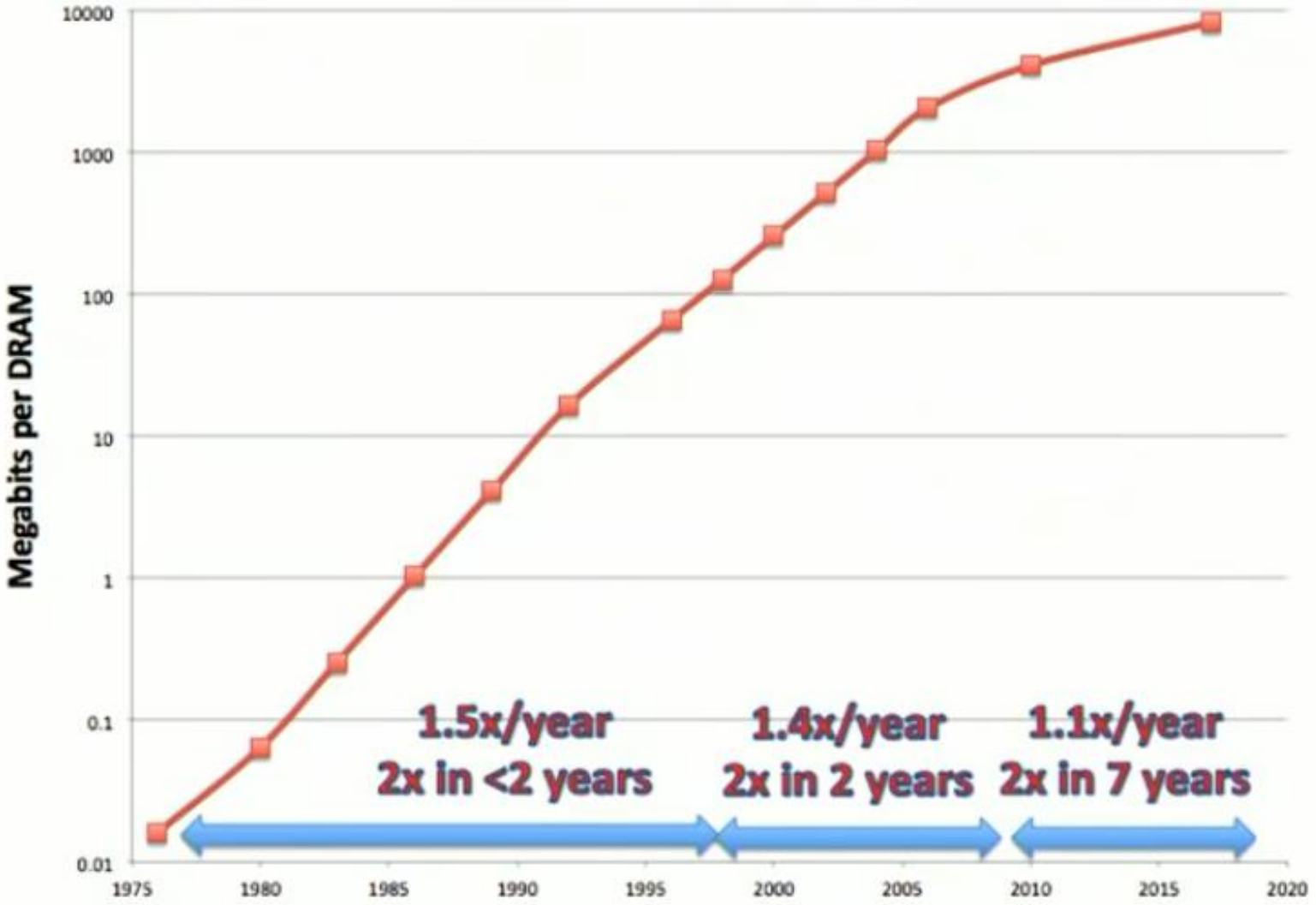
The problems of DNNs when deploying

Computation intensive **&** Memory intensive

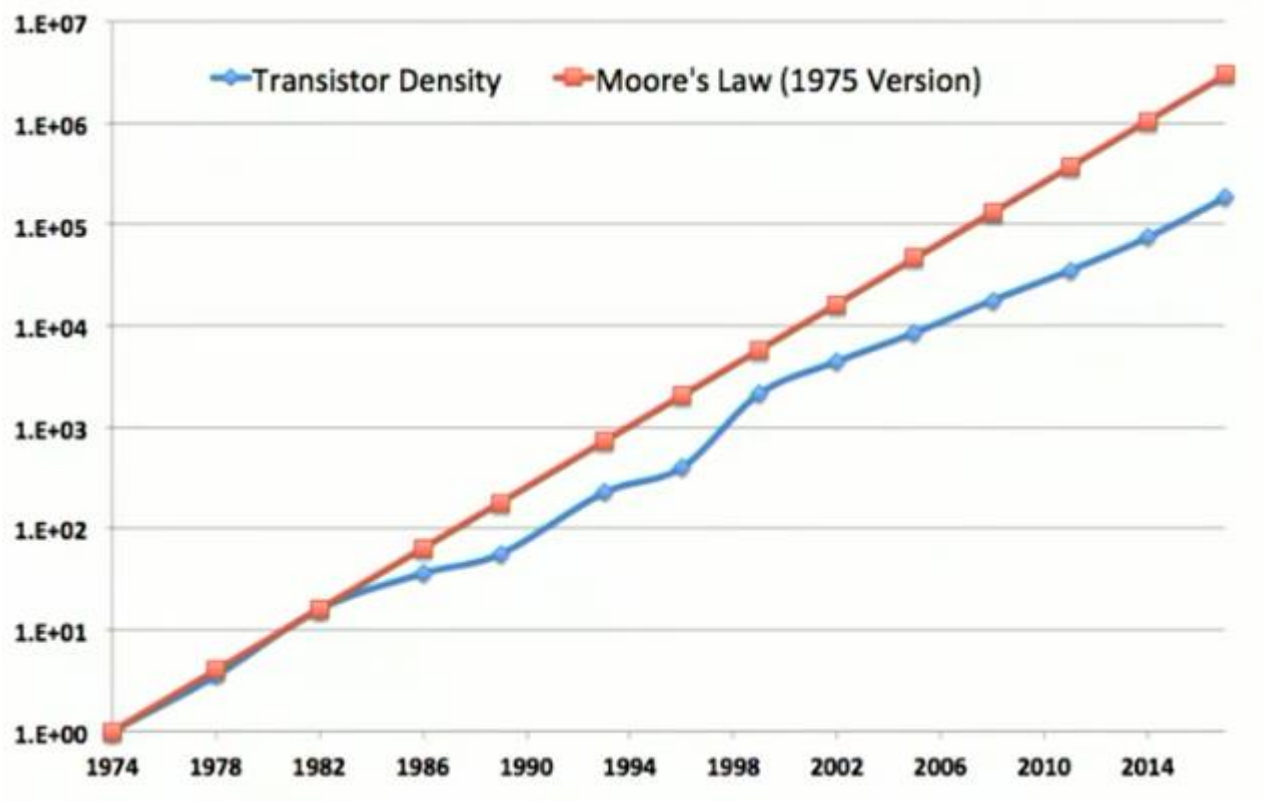
Challenging to deploy in end-devices

Cost & Latency & Power

Moore’s law in DRAMs

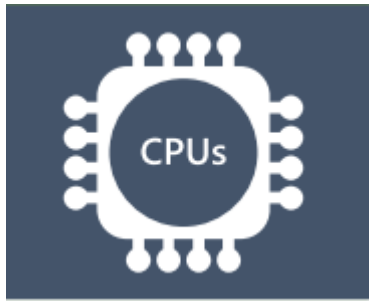


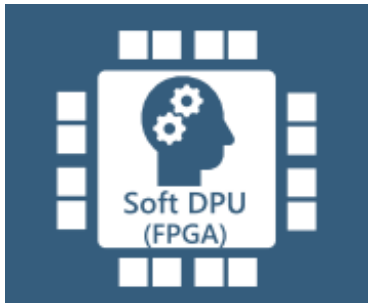
Moore’s law slowdown in intel processors

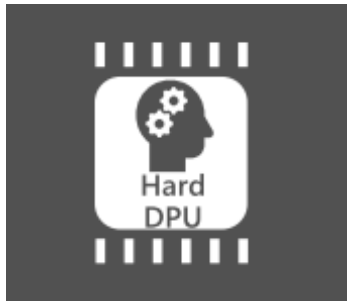


Cost per transistor is slowing down faster，due to fabrication costs

Silicon alternatives for DNNs







Flexibility ← → **Efficiency**

MS BrainWave
Baidu XPU
Teradeep
Deephi Tech
etc.

Google TPU
Intel Nervana
Wave Computing
Movidius
etc.

The power of deep learning on FPGA

Performance

- Excellent inference performance at low batch sizes
- Ultra-low latency serving on modern DNNs(>10X lower than CPUs and GPUs)
- Minimize memory bandwidth via data re-distribution
- Optimized algorithm to decrease the number of multipliers

Flexibility

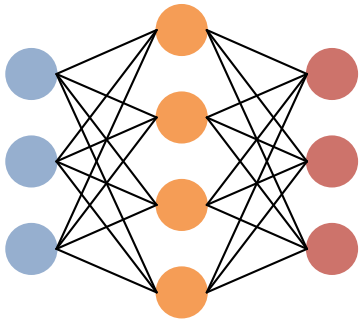
- Ideal for adapting to rapidly evolving ML
- CNNs, LSTMs, MLPSs, Reinforcement Learning, etc.
- Inference-optimized numerical precision
- Exploit sparsity, deep compression for larger, faster models

Scale

- Compose multiple computing-engines together to support more models at the same time

FPGA-powered platform solution for CNN

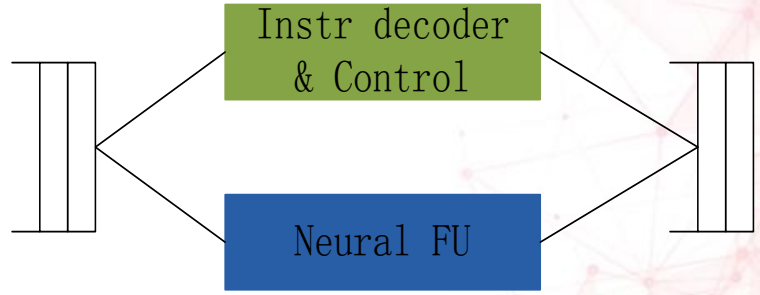
- ✓ Fast: ultra-low latency, high-throughput serving for CNN models at low batch sizes
- ✓ Flexible: adaptive numerical precision and custom operators
- ✓ Friendly: compiler tool-chain for faster development & deployment



Pre-trained DNN model
in CAFFE, etc.

Data Quantization
Network Optimization

Compiler tool-chain

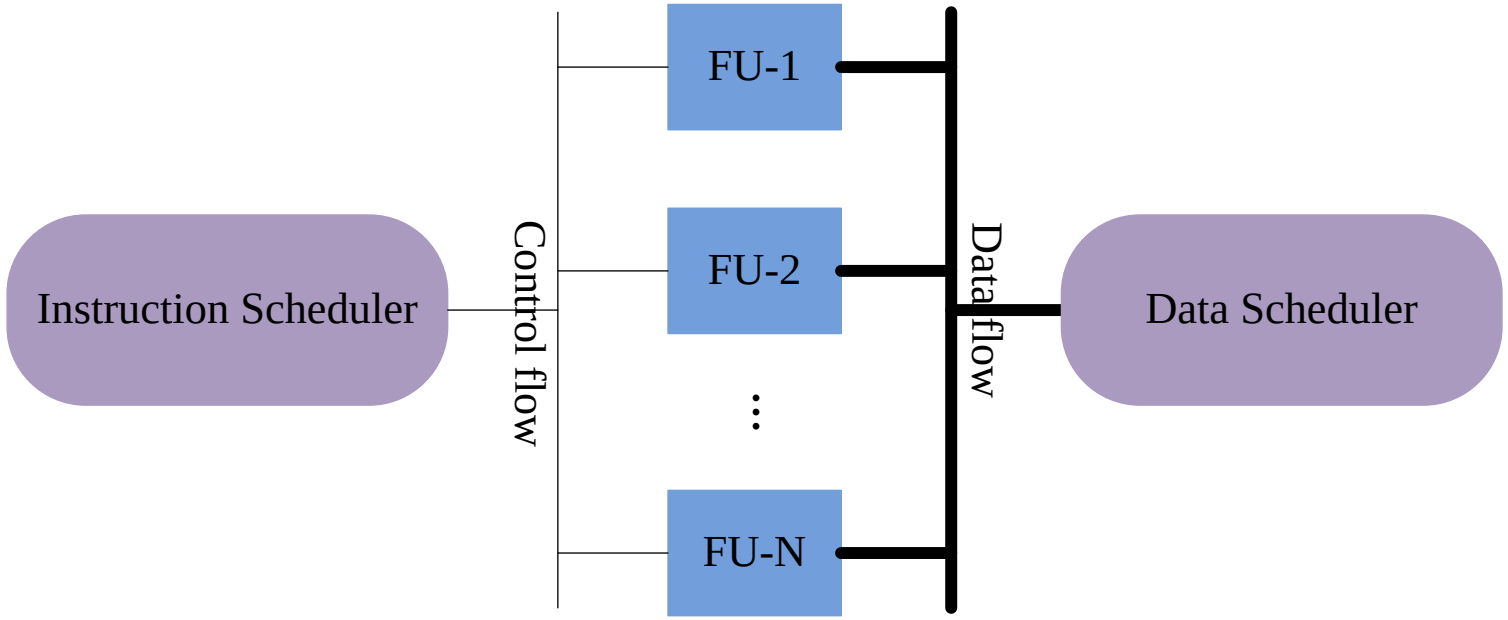


CNNiE
Soft DPU

How it works: The Solution Stack

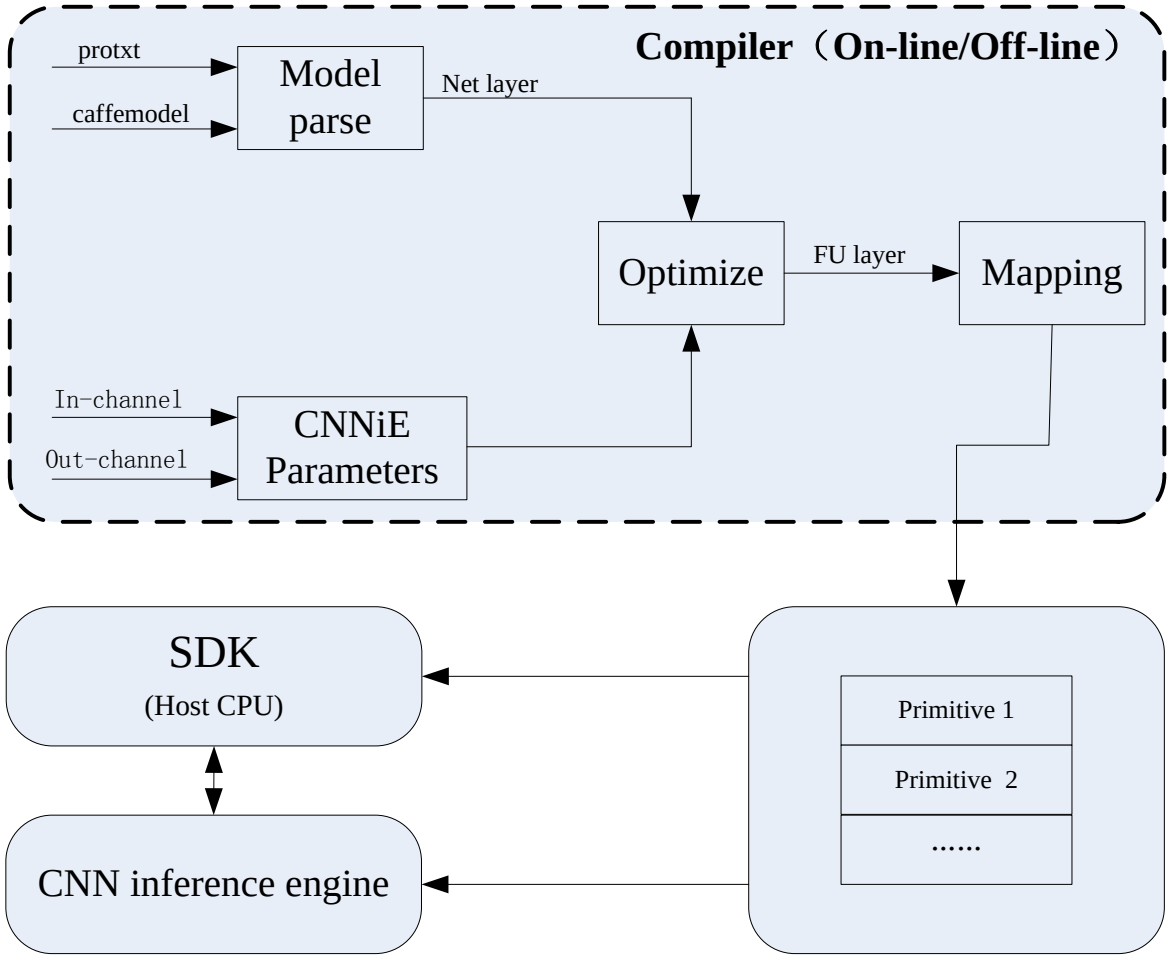
Compiler & RunTime	A framework-neutral federated compiler and runtime for compiling pre-trained DNN models to CNNiE (soft DPUs)
Architecture	Adaptive ISA for narrow precision DNN inference Flexible and extensible to support fast-changing AI algorithms
Microarchitecture	CNNiE Soft DPU microarchitecture Coarse-grain by layers, fine-grain with parallelism and pipeline Highly optimized for narrow precision and low batch

CNNiE：CNN inference engine



- Inter-Layers: Combined via FUs, coarse-grain parallelism
- Intra-Layers: Fully-pipelined/Multi-dims(input/output feature maps/kernel) parallelism, fine-grain
- Supported layers: Convolution/Pooling/ReLU/Interp/Fully connected/Element wise/Concat/Softmax
- Flexible numerical precision: fp16/int16/int8
- Winograd Optimization，the same performance with ~4x less DSP unit

Compiler toolchain：faster efficiency



- Faster development
- Auto-mapping from model to primitives
- Compile mode: Online/Offline

Run-Time SDK API：Easy to use

```
CHECK_RESULT(cnnieOpenDevice(&dev), "open cnnie device");
CHECK_RESULT(cnnieLoadModel(dev, (const char **)models, mode), "load models");
CHECK_RESULT(cnnieRun(dev, 0, &blob), "write input data, run net model");
CHECK_RESULT(cnnieGetOutputNum(dev, &output_num), "Get output num");
for(int i = 0; i < output_num; i++) {
    CHECK_RESULT(cnnieGetOutputData(dev, i, &blob), "Get output data");
}
CHECK_RESULT(cnnieCloseDevice(dev), "close cnnie device");
```

```
./test_cnnie models/6scales-max/net_train_test_max.prototxt models/6scales-max/UMDFace_SSD_512x512_iter_44000.caffemodel imgs/face_01.jpg
```

Initialization

- cnnieOpenDevice: Open device, alloc resource
- cnnieLoadModel: Load model

Run in time

- cnnieRun: prepare data, launch cnnie
- cnnieGetOutputNum: get net output layer number
- cnnieGetOutputData: get net output data

End

- cnnieCloseDevice: Close device, release resource

Platform performance

Direct Convolution -> Wino1D 1x2 -> Wino2D 2x2

	WI	HI	CI	WO	HO	CO	Time cost(us)			GOPS		
							Dire	1D 1x2	2D 2x2	Dire	1D 1x2	2D 2x2
1	640	480	3	320	240	4	1562	795	393	42.48092	83.46566	168.8427
2	320	240	4	160	120	8	850	496	246	52.04329	89.1871	179.8244
3	160	120	8	160	120	16	836	558	379	52.91483	79.27742	116.7198
4	160	120	16	80	60	16	1669	1024	649	53.00995	86.4	136.323
5	80	60	16	80	60	32	853	428	227	51.86026	103.357	194.8758
6	80	60	32	40	30	32	1693	851	410	52.25848	103.9643	215.7893
7	40	30	32	40	30	32	495	297	127	44.68364	74.47273	174.1606
8	40	30	32	40	30	32	494	297	129	44.77409	74.47273	171.4605
9	40	30	32	40	30	32	494	297	130	44.77409	74.47273	170.1415
10	40	30	32	40	30	6	123	62	30	33.71707	66.89032	138.24
11	40	30	32	40	30	12	186	92	45	44.59355	90.15652	184.32
Total							**9255**	**5197**	**2764**			

Platform performance & resources

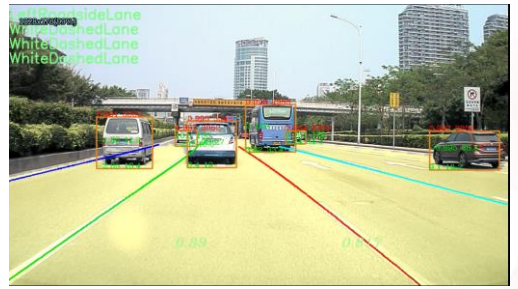
Direct Convolution -> Wino1D 1x2 -> Wino2D 2x2

	Zynq-7045	Dire	1D 1x2	2D 2x2
LUT	218600	12599	20719	28276
FFs/Registers	437200	18788	21906	30594
DSP	900	144	192	256
RAM	545	66	66	42

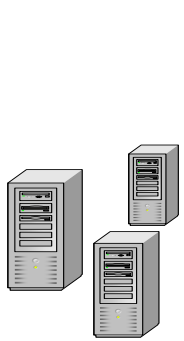
	Dire	1D 1x2	2D 2x2
Ideal GOPS(200MHz)	57.6	115.2	230.4
Test GOPS(MAX)	53.0	104.0	215.8
Efficiency(MAX)	92.0%	90.3%	93.7%
DSP	1	1.3	1.8

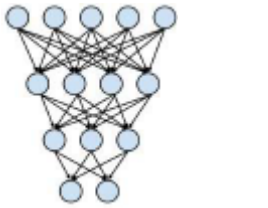
When DSP increased by 30% and 80%，the computing performance increased by 100% and 300% respectively

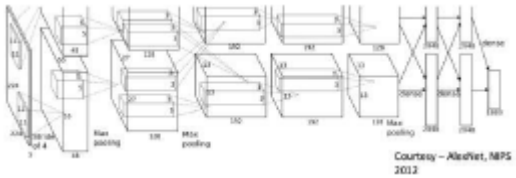
Deep learning for ADAS with FPGA



Weight/Parameters







Model

Product solution

vision based application

LDW/FCW/UFCW/PDW/HMW/SLI/TLR/AEB

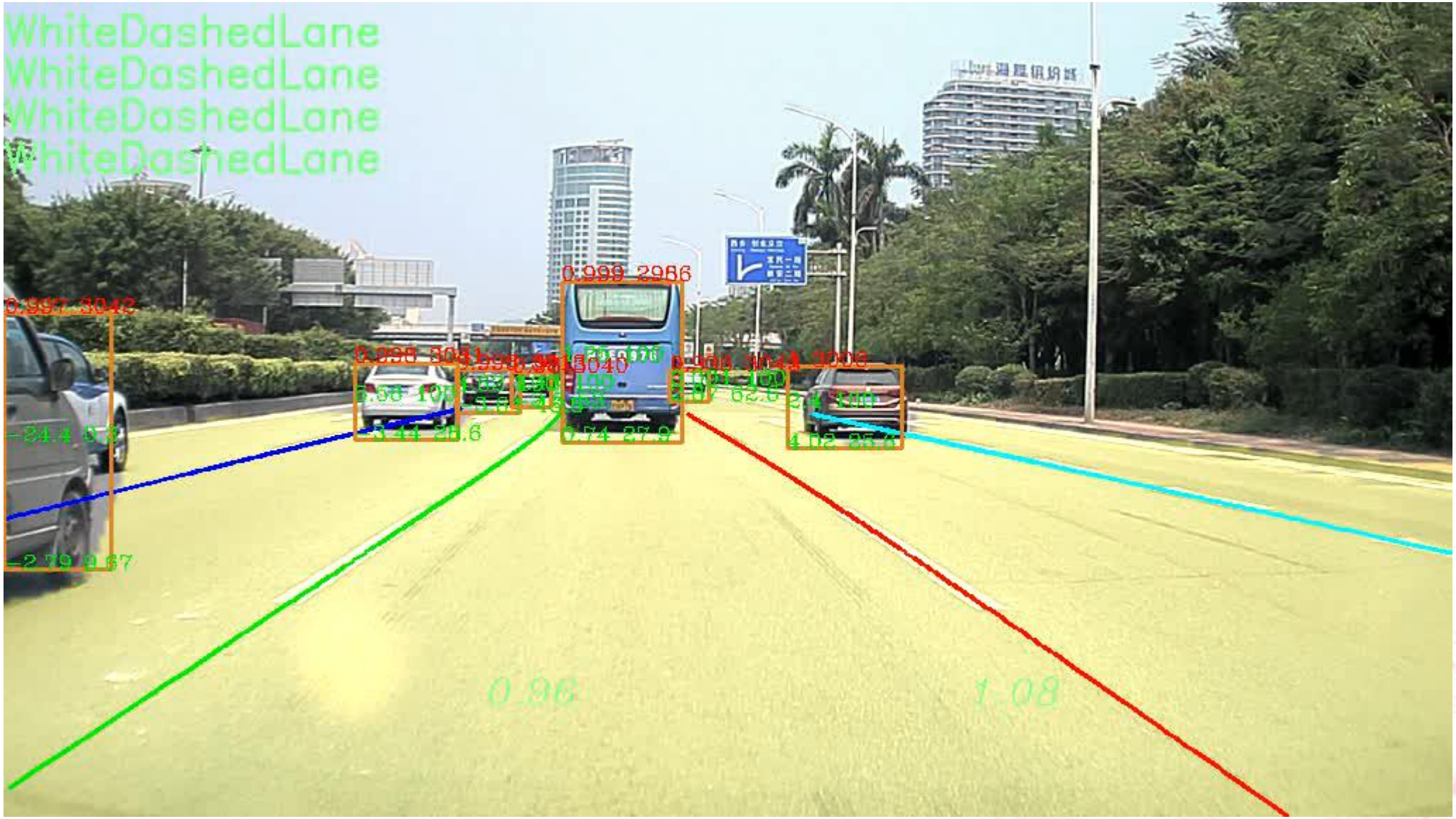
lane/type/free space

car/nonMotorVehicle/human

traffic light/sign

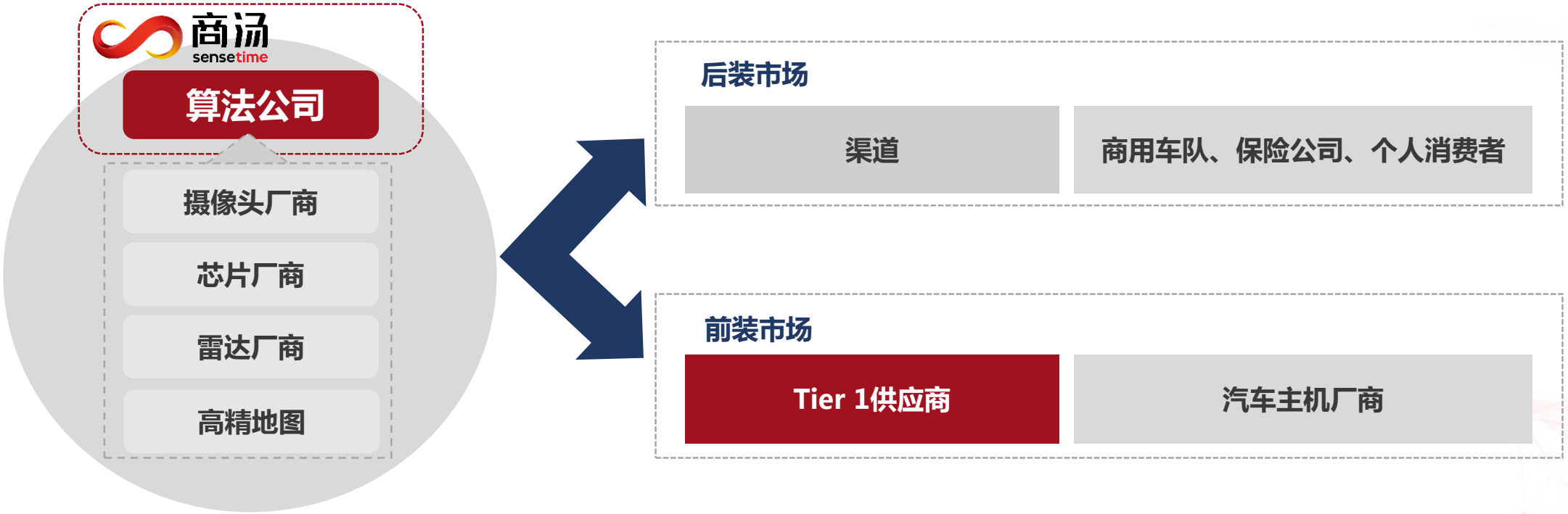
heterogeneous platform computing (ARM+GPU/FPGA etc.)

Deep learning for ADAS：Field test



Deep learning for ADAS：Field test

智能驾驶 | 商汤在ADAS产业链中的位置：Tier2+算法决策



- 商汤**以深度学习和算法为核心、立足于道路测试和数据积累**，完成了高性能的汽车视觉软件算法
- 采用具备成本优势的**单目摄像头**，同样实现了高效的FCW、LDW、PCW等ADAS 功能
- 未来**依靠算法的进步能实现的成本下降**的幅度可能完全不输于硬件所带来的成本下降

智能驾驶 | 商汤 ADAS产品&解决方案

- 基本功能：：LDW/FCW/PDW
- 可选： SLI/HMW/UFCW等
- 前装车规质量
- 预警功能方案包，可选可定制

- 推荐硬件方案，客户生产.
- 产品镜像+SDK（检测 or 预警）
- Application可灵活定制

智能驾驶 | ADAS前装软硬一体化解决方案

产品形态

- 摄像头和运算平台一体式
- 质量标准：符合ISO16949

硬件规格

- 处理器：FPGA车规级芯片
- 内存：1GB DDR3
- 摄像头：1MP 、FOV 52°
- IO接口：CAN总线

功能

- 车道偏离预警/车道保持辅助
- 前向碰撞预警/行人碰撞预警
- 可行驶区域
- 前车追尾AEB/行人碰撞AEB
- 交通信号灯识别/交通标志识别

智能驾驶 | ADAS SDK功能

车道线检测

车辆行人检测

可行驶区域检测

交通标志识别

智能驾驶 | 算法参数

01 车道线识别

检测数量	4根（左左车道、左车道、右车道、右右车道）
车道线属性	黄/白、虚/实、单/双、路边沿
检测类型	直道、弯道、部分遮挡车道（预测）
检测距离	50米

02 目标探测

机动车	小汽车、客运车、货车、卡车、常见异型车等
非机动车	自行车、三轮车、摩托车等
行人	正面、背面、侧面、站/蹲、半遮挡
检测距离	行人/非机动车50米，机动车100米
测距精度	近距离绝对误差小于0.5m，远处误差小于5%

03 交通标志识别

限速标志	限速5、限速15、限速20、限速30、限速40、限速50、限速60
禁止标志	禁止左转、禁止右转、禁止转弯、禁止掉头、禁止车辆临时或长时停放、停车让行、禁止驶入

04 交通信号灯识别

颜色	红、绿、黄
形状	圆形、箭头

05 可行驶区域

检测距离	50米

智能驾驶 | 商汤ADAS优势

预警准确率高

基于商汤科技自主设计的深度学习网络结构，视觉感知算法更精准，预警准确率更高

报警及时

基于商汤科技自主设计的深度学习计算硬件加速引擎，实时并行处理多种算法，预警更及时

鲁棒性高

全天候全时段数据采集与训练，适用各种光照、天气、道路场景，预警更通用

自标定快

在算法设计的时候充分考虑了内外参的兼容性，并在不同车型的测试中验证，简化了安装步骤

车规级

车规级标准，性能更稳定

Adaptable.
Intelligent.