在MCU上应用机器学习实现人工智能

ROCKY SONG 宋岩

MCU系统工程部

SECURE CONNECTIONS
FOR A SMARTER WORLD

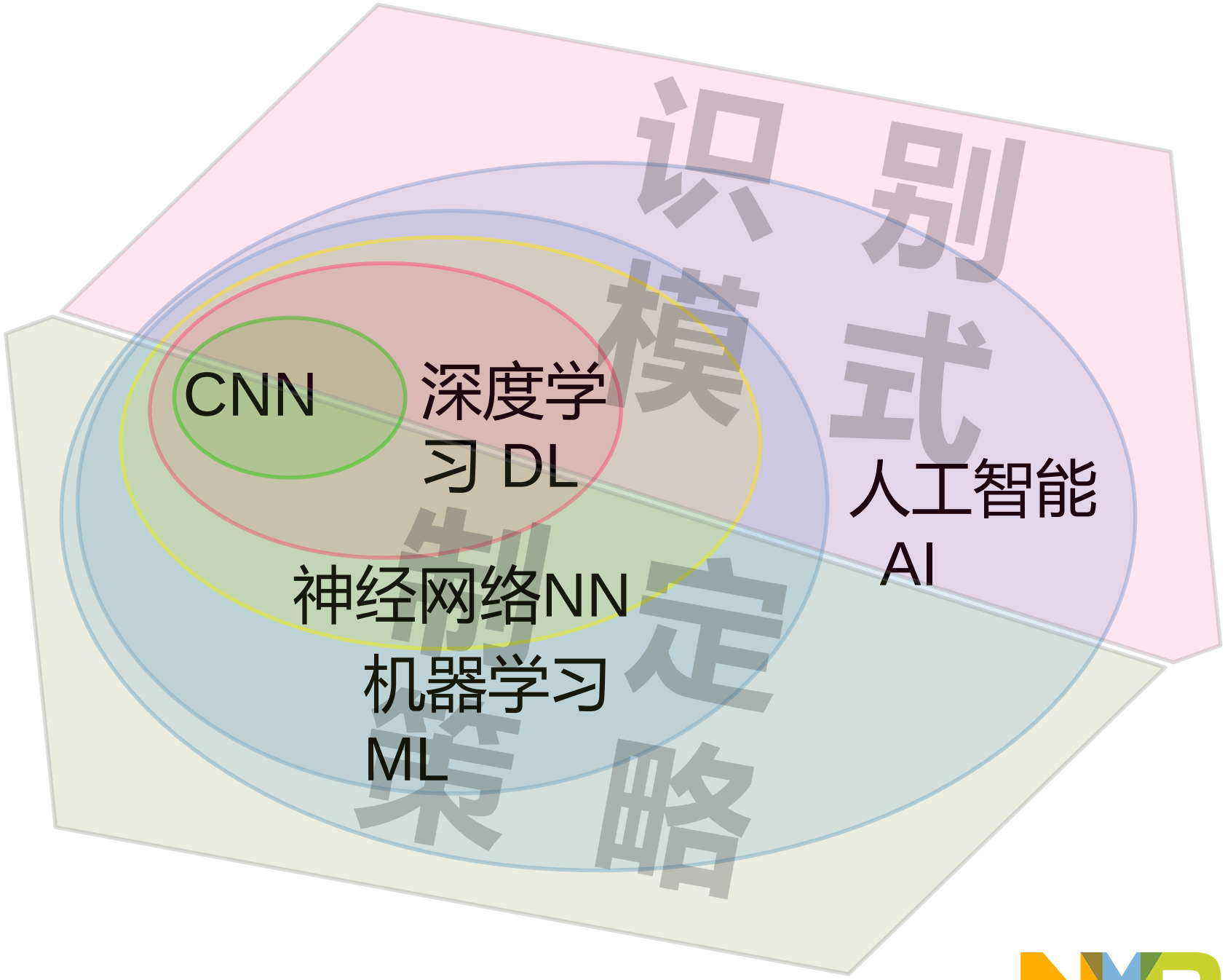
内容提要

- 前半场
  - AI与MCU
  - 工具与工作流程
- 后半场
  - 实例与应用
  - 使用神经网络建模

AI, MCU

人工智能（AI）

- 预测数值
- 辨别种类
- 监控状态
- 分析结构



实现人工智能的演进

xx 从经验(数据）中学习的组件

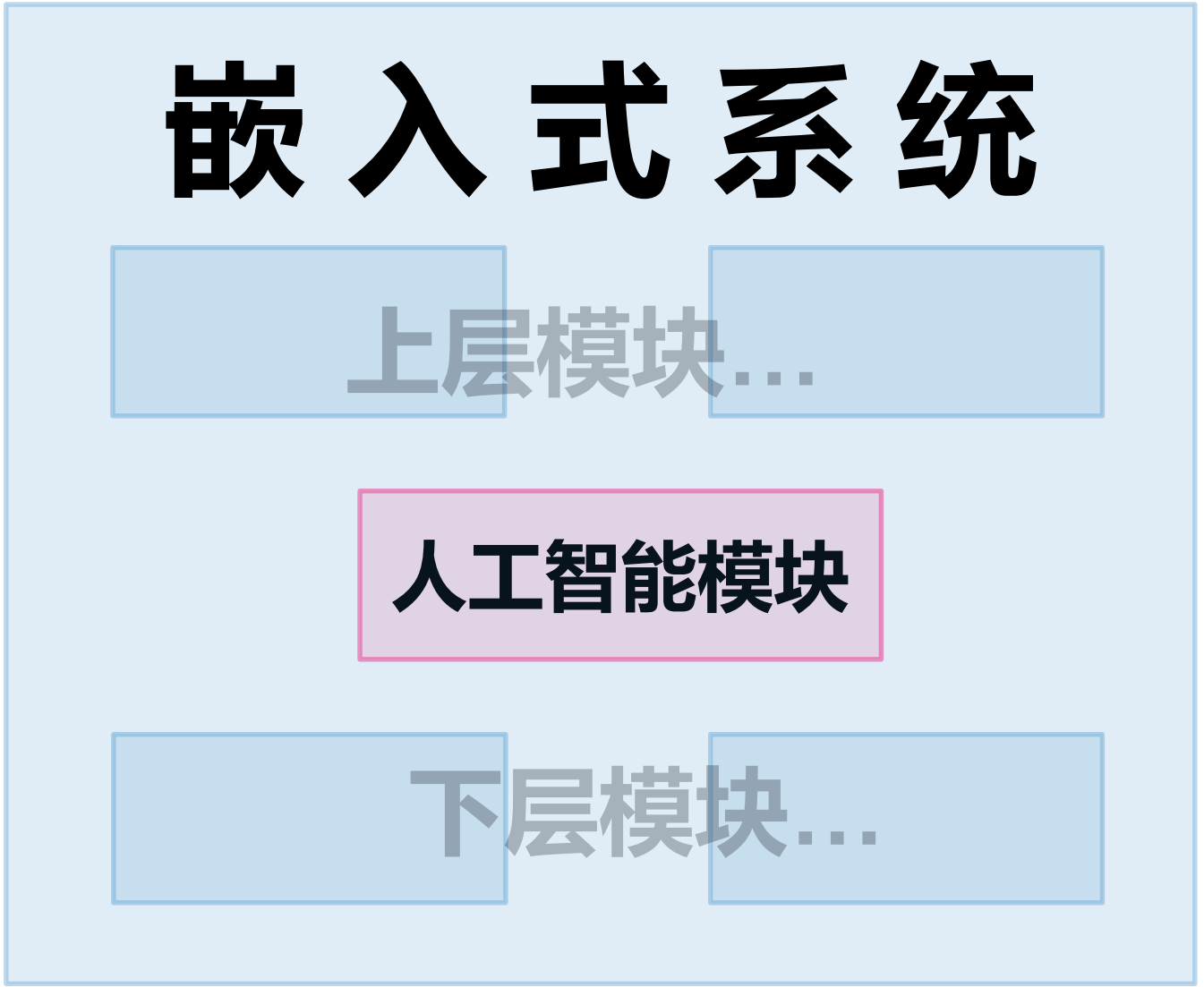
DL常用基于NN的方法，也可沿用传统ML

DL模型**自动**找出特征，多达成百上千上万!

DL模型基于多层**神经网络**

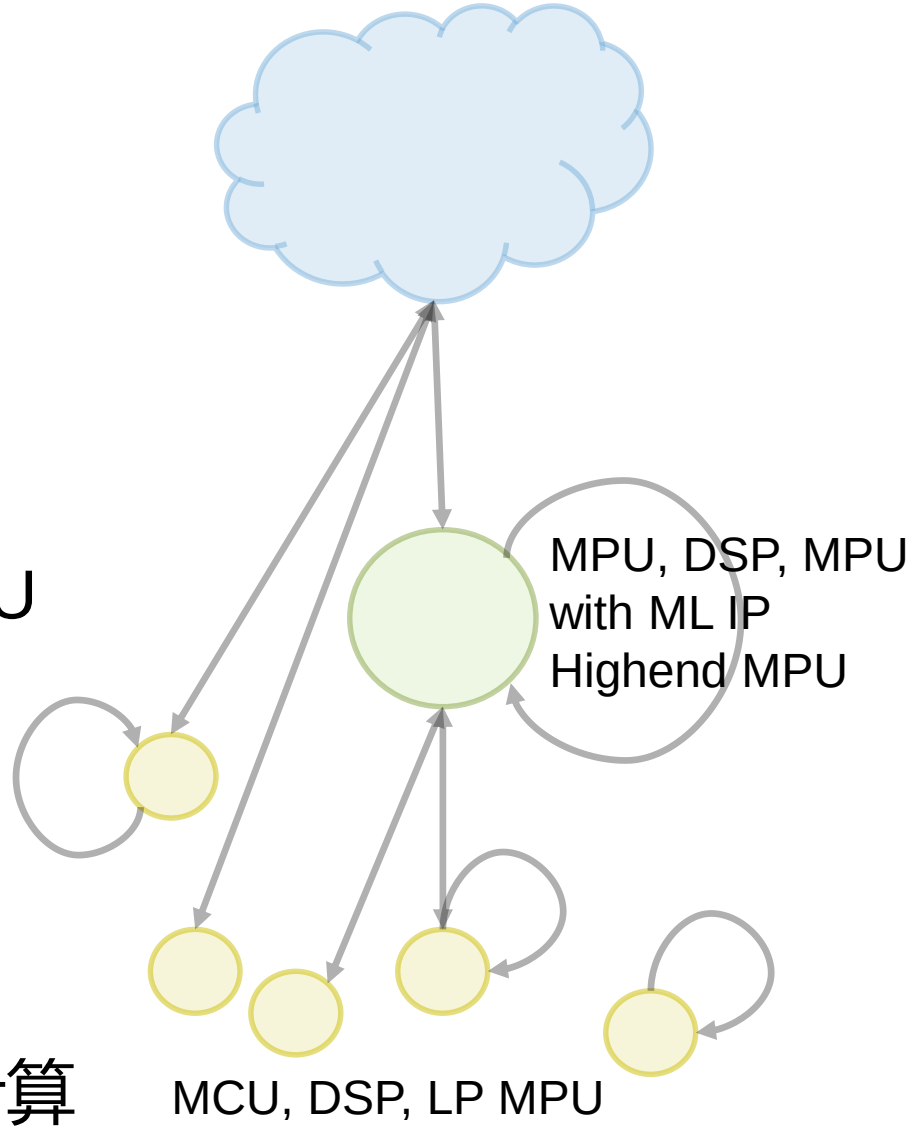
人工智能与嵌入式系统

- 系统是主体
- 人工智能是“装备”
- 强大的“属性加成”
- 以模块来呈现
- 提供新功能
- 改进现有功能



智能应用在IoT边缘的分布

- **云端服务**
  - –运行最复杂的模型并提供IoT服务
- **富功能节点与边缘门户**
  - –跨界处理器，MPU，以及带机器学习IP的MCU
  - –运行更加大型的模型，并提供IoT网关
- **IoT节点与离线节点**
  - –MCU, DSP, 入门级跨界处理器
  - –运行**轻型智能**模型，可永远在线，完成初步计算



“轻型智能”，轻在何处?

- 规模小：模型尺寸和算力要求低
- 自包含：可独立运行AI模块，无需云端连接
- 应用专：对重点应用量身定制
- 优化高：针对计算平台高度优化
- 功耗低：实现IoT设备的长期工作
- 响应稳：多有实时性要求，快速而确定

适合在MCU风格的平台上使用

MCU上AI应用的特点

AI/ML 在MCU级嵌入式系统上的应用领域

大领域	有利条件	挑战
生物识别 (指纹，静脉，声音，面部)	容易想到和接受，量大 正在更新换代(高性能CPU -> DSA)	玩家太多，红海杀价 社会问题
（低实时）现场监控 货架/库存/抄表/过程/畜牧/植被/野外大型设备/气象)	实时要求低得多，可以在秒-分钟的级别响应	模型尺寸大，算力要求高
可穿戴设备 手势, 姿态, 运动，语音口令	模型简单，算力要求低	超低能耗
自学习/自改进设备 随着使用越久，表现越好	改造现成的应用，领域广	传感器/HMI的附加成本 需要在设备上训练
异常检测 & 事故预测 状态异常，设备老化/损坏 事故预测，凶险急症预测	保障功能安全的要求	高度可靠而准确的模型

AI/ML 在嵌入式系统上的应用领域 – 续

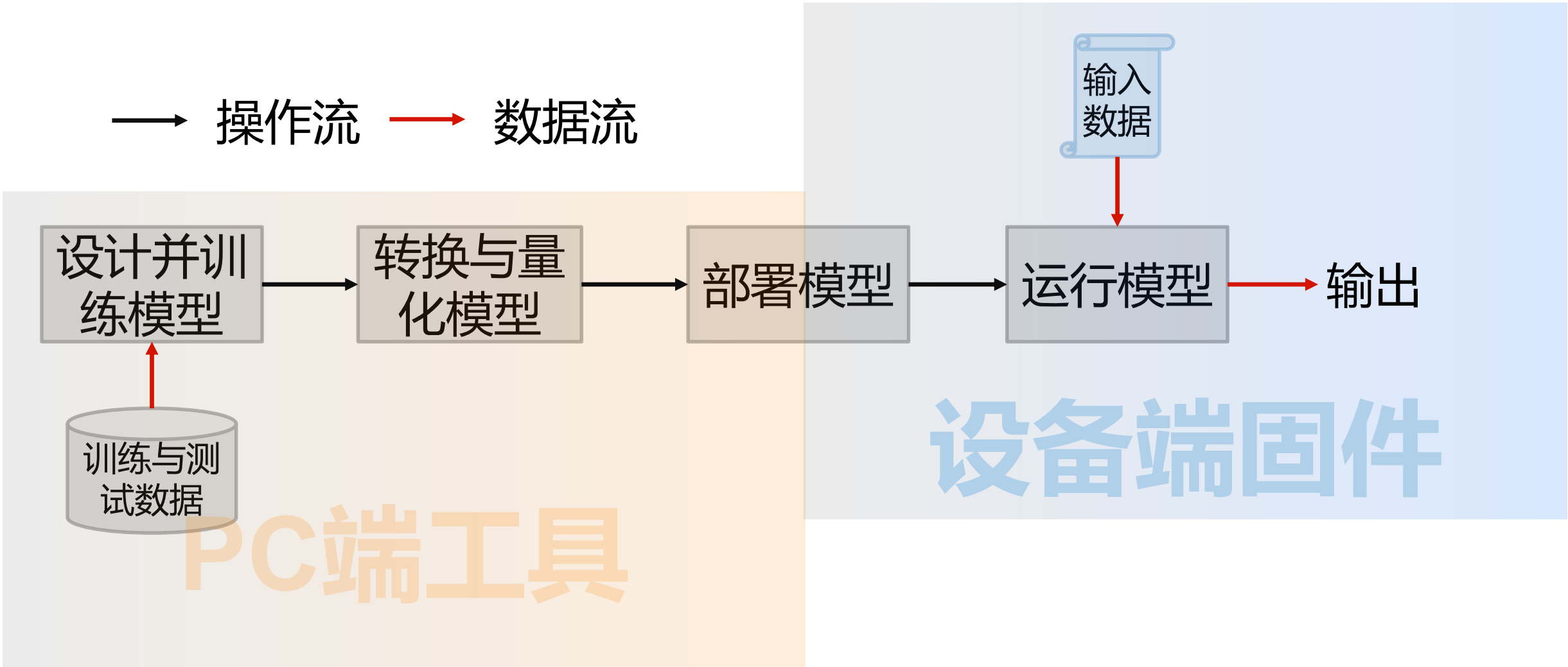
大领域	有利条件	挑战
AI 教育, 竞赛, 娱乐 智能车/船/机器人/无人机/ 可编程硬件	可实验的领域广，落地限制少	需要创意
智能控制模块 给水, 追光, 温控, 报警,喷淋, 自动闸控 水/电/气/刀……	只需在现有执行设备上加装模块	模块安装方式杂 产业碎
算法 转 机器学习 规则与代码驱动 -> 数据驱动	现成的应用 改进性能或精度 更新换代以创造市场机会	改动现有代码 领域有关的数据集 理解应用
产线监控/ 产品/零件质量检查	刚需，实时要求适中	领域有关数据集，产线改造

MCU上应用AI的难点与应对策略

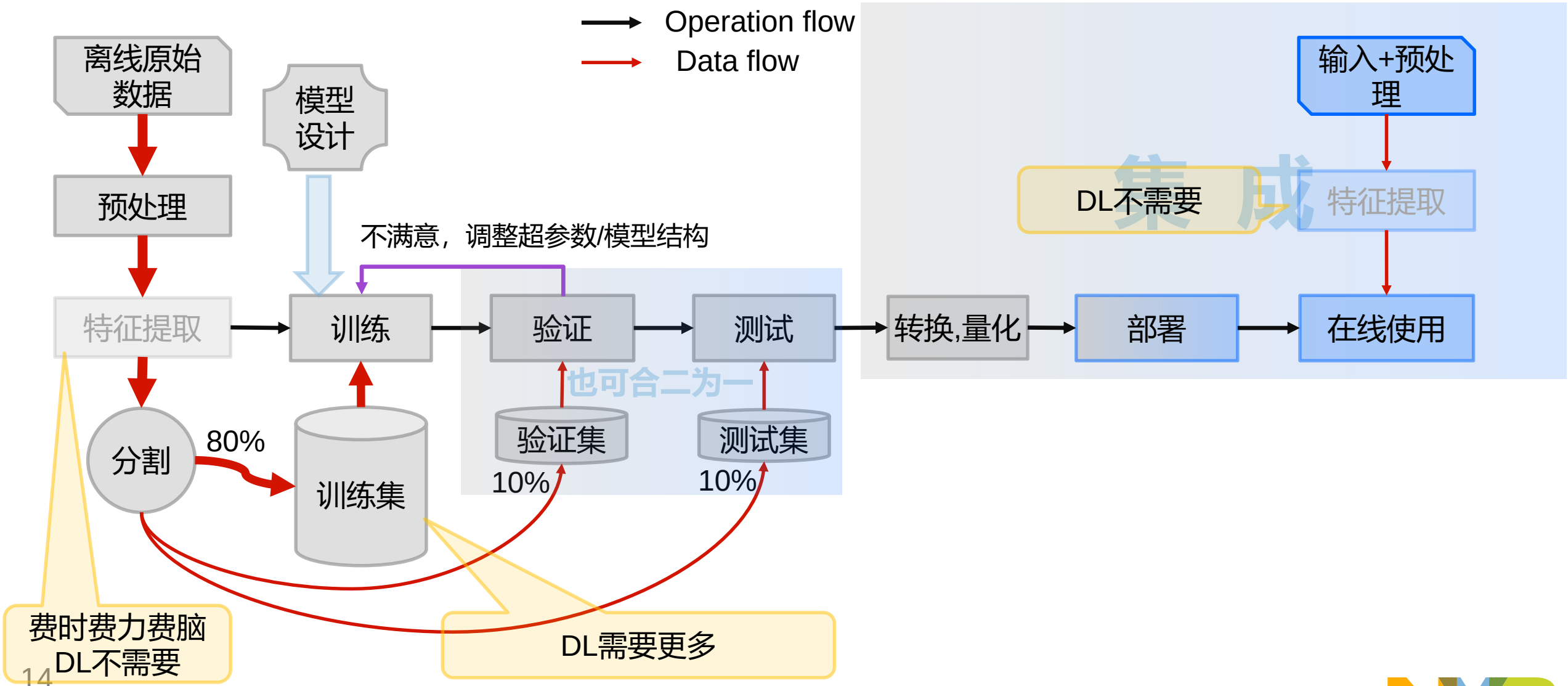
难点	应对策略
算力弱 即使是目前性能最高的MCU，i.MX RT1050/60，int16算力也只有1.2G MAC/s	• 使用较少位数量化模型 • 合理精简模型规模 • 高度优化底层代码 • 充分利用异构多计算单元
缺少建模与训练工具	借助PC/Server来建模与训练
缺少集成工具	NXP提供eIQ (边缘智能)工具

工具与工作流程
EIQ软件包

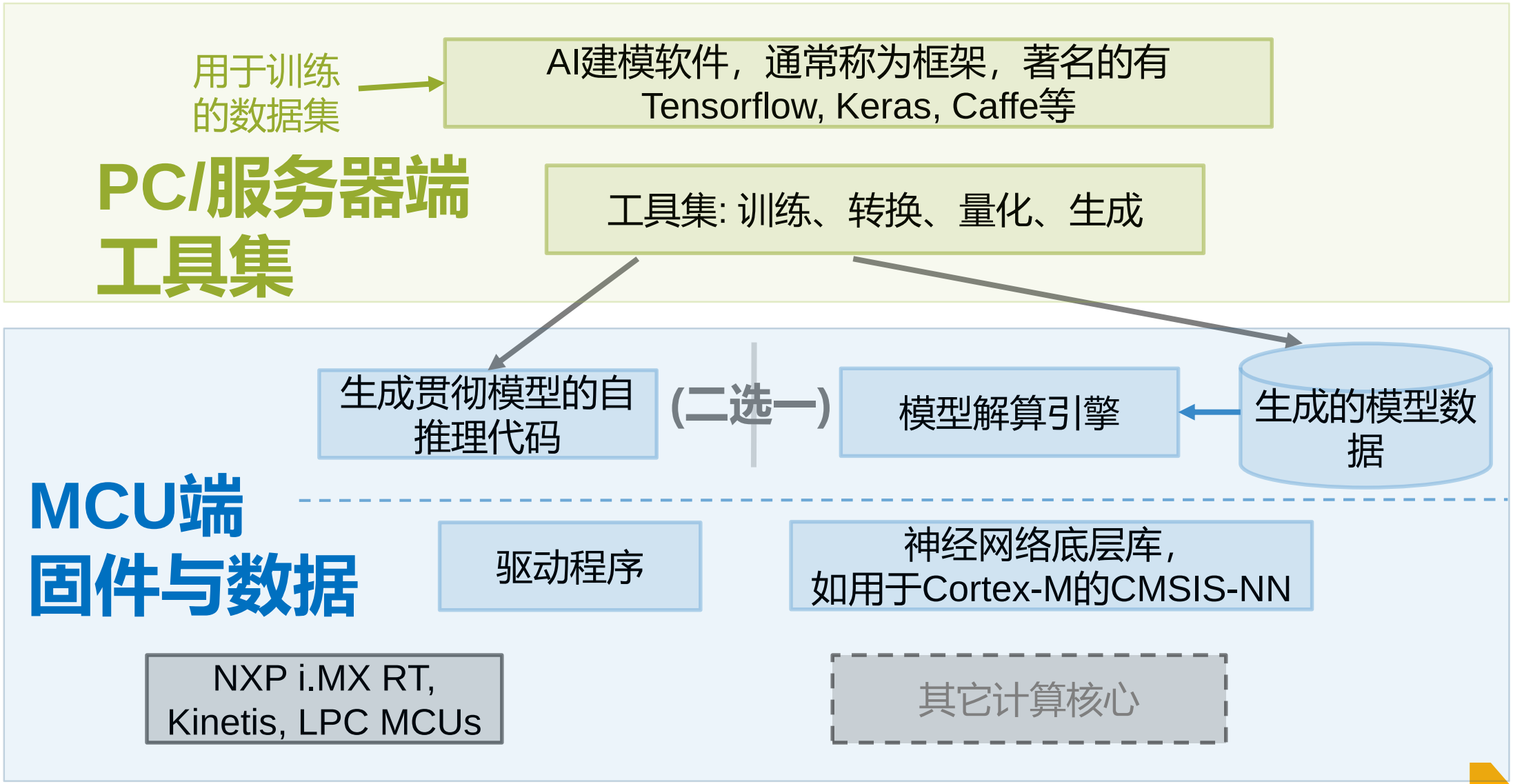
在MCU上集成AI的整体流程



工作流程细节


Operation flow
Data flow
离线原始数据
模型设计
输入+预处理
预处理
DL不需要
特征提取
特征提取
不满意，调整超参数/模型结构
训练
验证
测试
转换,量化
部署
在线使用
也可合二为一
分割
80%
训练集
验证集
测试集
10%
10%
费时费力费脑
DL不需要
DL需要更多

MCU上部署AI的配套工具



自推理代码一瞥

```
extern const int8_t cg_CONV1bias[]; // 128
extern const int8_t cg_CONV1weit[]; // 512 - Co,H,W,Ci: (128, 2, 2, 1)
extern const int8_t cg_FC2bias[]; // 128
extern const int8_t cg_FC2weit[]; // 720896 - Co, Di: (128, 5632)
extern const int8_t cg_FC3bias[]; // 128
extern const int8_t cg_FC3weit[]; // 16384 - Co, Di: (128, 128)
extern const int8_t cg_FC4bias[]; // 6
extern const int8_t cg_FC4weit[]; // 768 - Co, Di: (6, 128)
```

权重声明

```
int32_t img_buffer0[(5632 + 3) / 4];
int32_t img_buffer1[(22784 + 3) / 4];
int32_t col_buf[(11264 + 3) / 4]; // [2, 5632]
int8_t out_buf[6]; // FC4_OC
```

缓冲区(纯串行模型)

```
int32_t img_buffer0[(131072 + 3) / 4];
int32_t img_buffer1[(131072 + 3) / 4];
int32_t img_buffer2[(32768 + 3) / 4];
int32_t col_buf[(32768 + 3) / 4]; // [
int8_t out_buf[128]; // CONV50_OC
```

缓冲区(包含残差连接的串行模型)

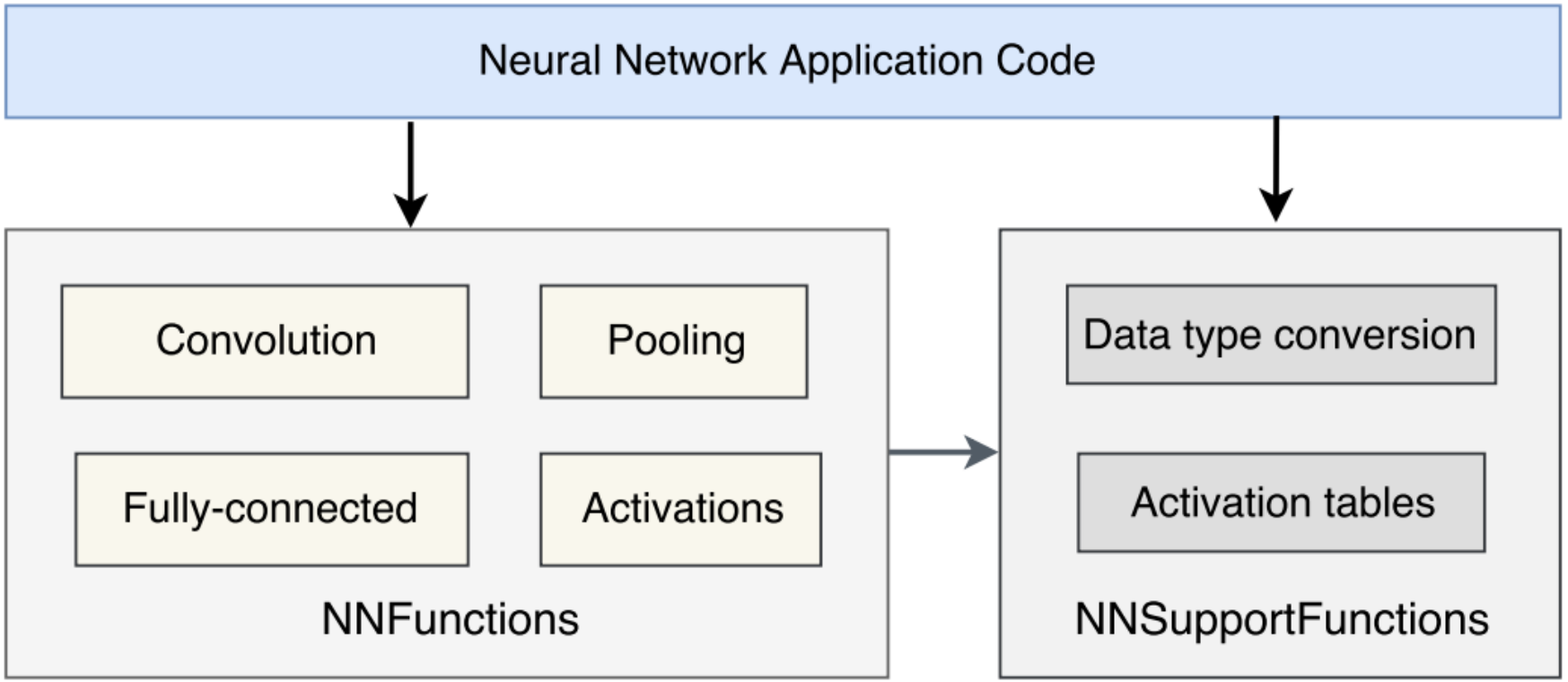
```
// generated RunModel(), returns the output buffer of
void* RunModel(const void *in_buf) {
    memcpy(img_buffer0, in_buf, 270);
    // Block 1: Conv2D - conv2d_1
    arm_convolve_HWC_q7_basic_nonsquare((q7_t*)img_bu
    /*1*/
        , cg_CONV1weit/*weit*/, CONV1_OC/*128*/, CONV
        /*0*/, CONV1_SX/*1*/
        , CONV1_SY/*1*/, cg_CONV1bias/*bias*/, CONV1_
        CONV1_OX/*2*/
        , CONV1_OY/*89*/, (int16_t *) col_buf, NULL);

    // Block 1: Conv2D - conv2d_1, auxact relu
    arm_relu_q7((q7_t*)img_buffer1/*1*/, (uint32_t)(C

    // Block 1: MaxPooling2D - max_pooling2d_1
    arm_maxpool_q7_HWC_nonsquare((q7_t*)img_buffer1/*
     MAXP1_KX/*2*/
        , MAXP1_KY/*2*/, 0, 0, MAXP1_SX/*2*/, MAXP1_S
        col_buf, (q7_t*)img_buffer0/*0*/);

    // Block 2: Dense - dense_1
    arm_fully_connected_q7((q7_t*)img_buffer0/*0*/, c
    FC2_SB/*5*/
        , FC2_SO/*8*/, cg_FC2bias/*bias*/, (q7_t*)img

    // Block 2: Dense - dense_1, auxact relu
    arm_relu_q7((q7_t*)img_buffer1/*1*/, (uint32_t)(F

    // Block 3: Dense - dense_2
    arm_fully_connected_q7((q7_t*)img_buffer1/*1*/, c
    FC3_SB/*3*/
        , FC3_SO/*8*/, cg_FC3bias/*bias*/, (q7_t*)img
```

贯彻执行模型的函数

Cortex-M: CMSIS-NN鸟瞰

- 整数运算
- 常用NN算子
- **比纯C能有4.6倍性能提升和4.9倍能效提升**
- **不能单独使用，需配合上层工具集**

- 为Cortex-M DSP扩展优化的**NN基础库**
- 亦提供标准C参考实现



Cortex MCU运行神经网络的基础软件

CMSIS-NN

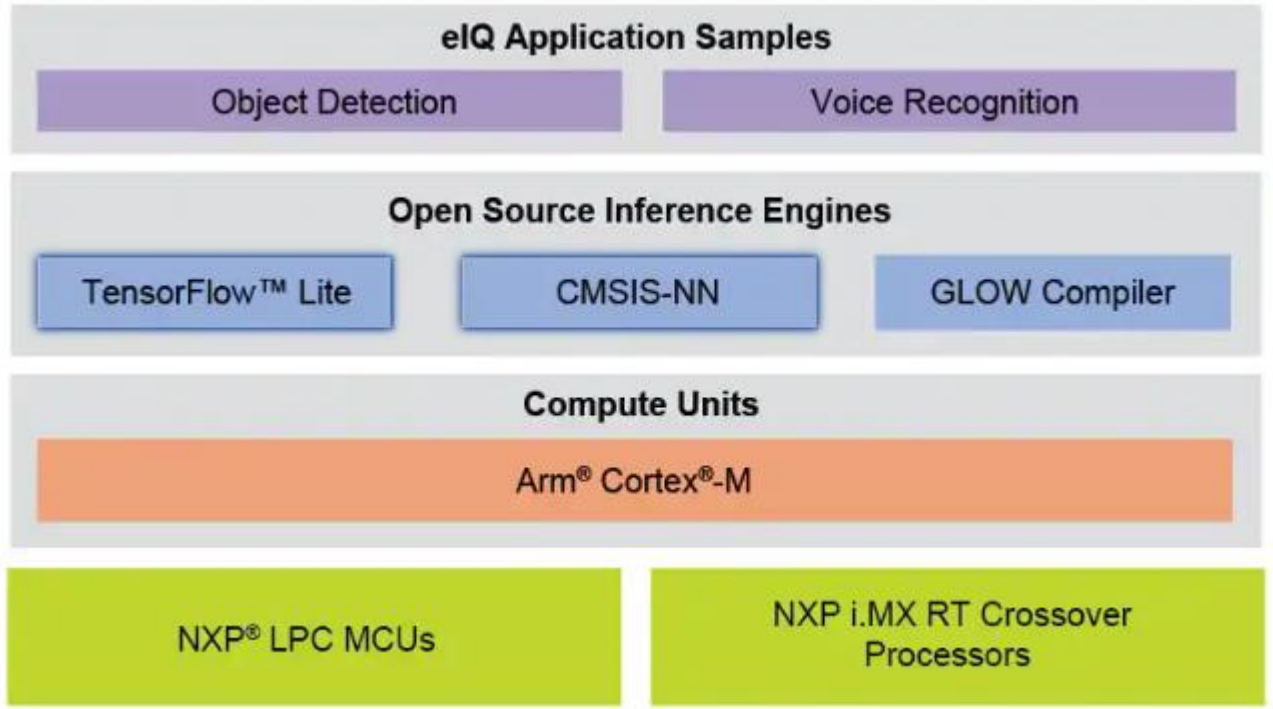
- 仅是底层NN库，需另行生成上层代码或提供执行引擎
- 针对Cortex-M高度优化
- 对算子的支持稍有薄弱
  - 我们补全了一些算子
- 更加高效的量化机制

TensorFlow Lite

- 自带执行引擎和底层NN库
- 性能远不如CMSIS-NN
  - ~20% for int8
- 支持丰富的神经网络构建块与搭建方式
- 支持多种嵌入式平台
- 更加完备的量化机制

NXP MCU + AI 工具计划

- “eIQ”软件包
- CMSIS-NN配套工具
- GLOW模型编译器
- 性能优化的Tensorflow-Lite
- 更多模型格式
- 更多NXP器件

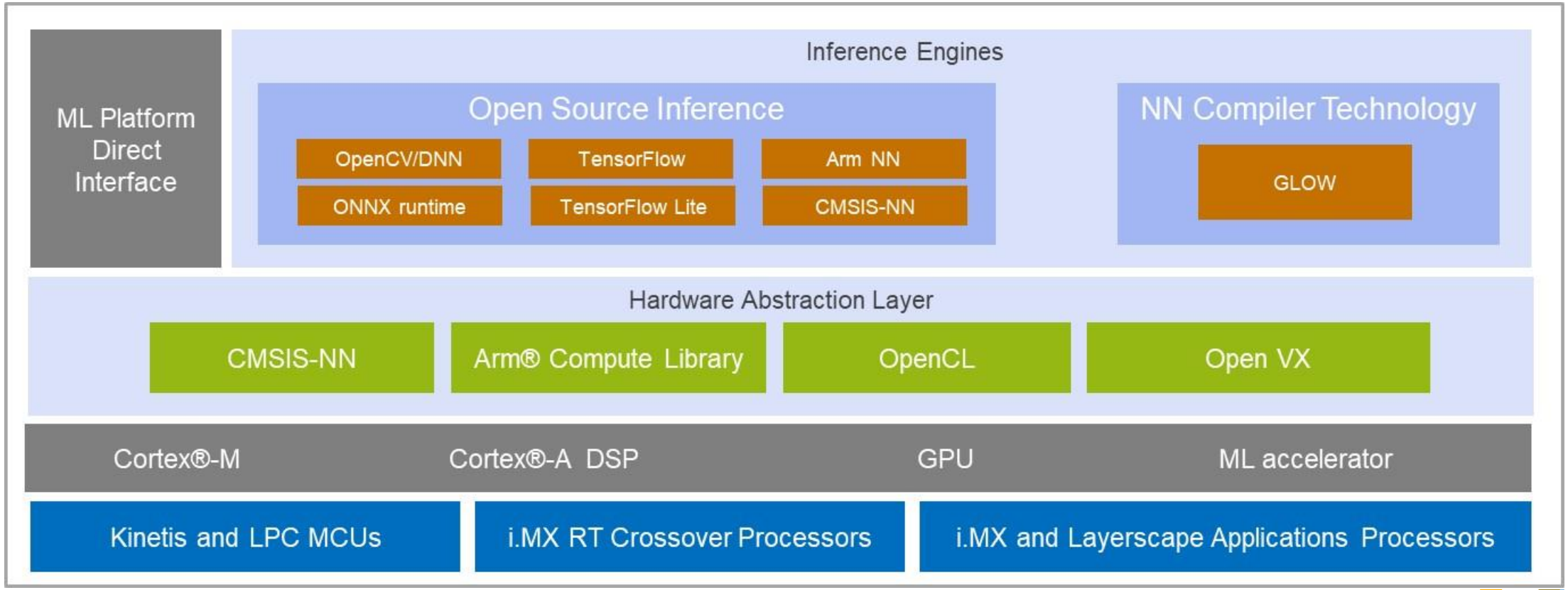


eIQ初探

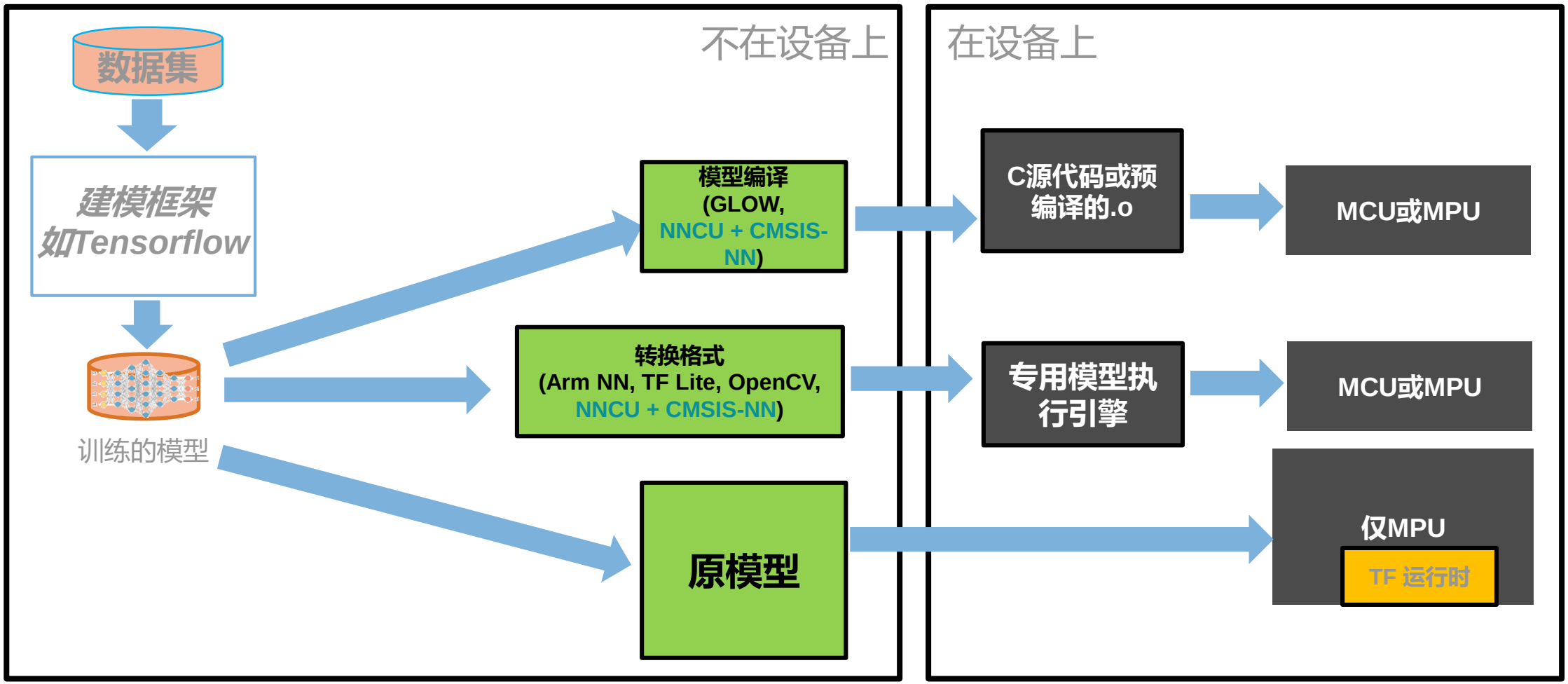
- 是在NXP MCU和MPU上运行AI模型所需的软件集合
- 开发中的工具:
  - 推理引擎: Arm NN, OpenCV, Arm CMSIS-NN, TensorFlow Lite, 等等
  - 在线实时示例，演示典型使用场景
  - 正在支持前沿的NN编译技术，如GLOW
  - 为传统机器学习开发独立的工具 (SVM, 随机森林等)
  - 为MCU和MPU各开发1套工具
- eIQ最终将以中间件纳入Yocto和MCUXPresso SDK的发布包

eIQ整体框架图

- eIQ是一套软件集合，用于在NXP MCU和MPU上运行ML模型
- 大量使用开源软件

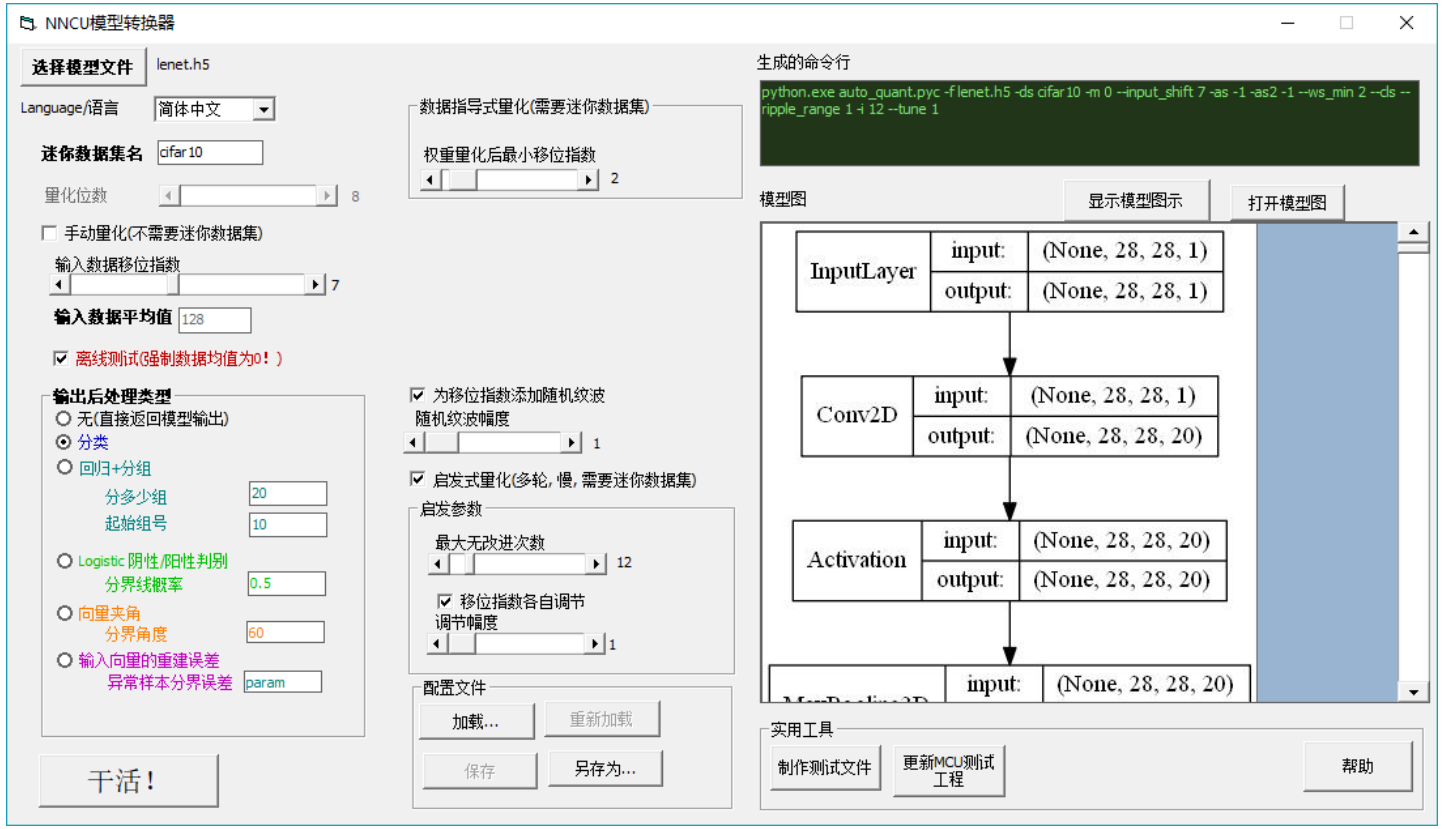


eIQ下的三种工作流程



为CMSIS-NN定制的工具集 –NNCU (NN toolkit for MCU)

- **配合CMSIS-NN库使用**
- 模型量化与转换工具
- 测试向量制作工具
- GUI界面
- 附带示例与迷你数据集
- 量化质量测试工具
- 转换后模型的解释器
- MCU测试工程与更新工具
- 详细的用户指南

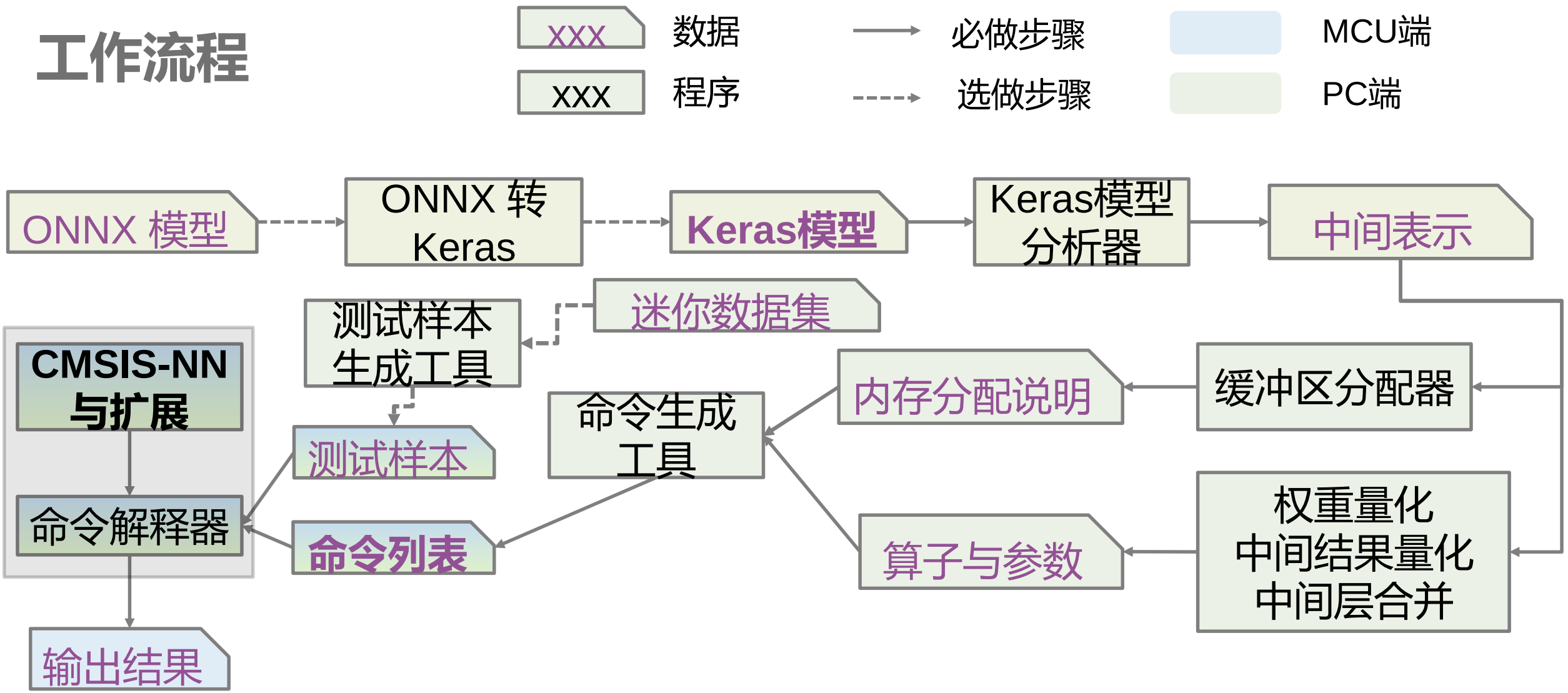


随demo逐步开发，**提供抢鲜体验**

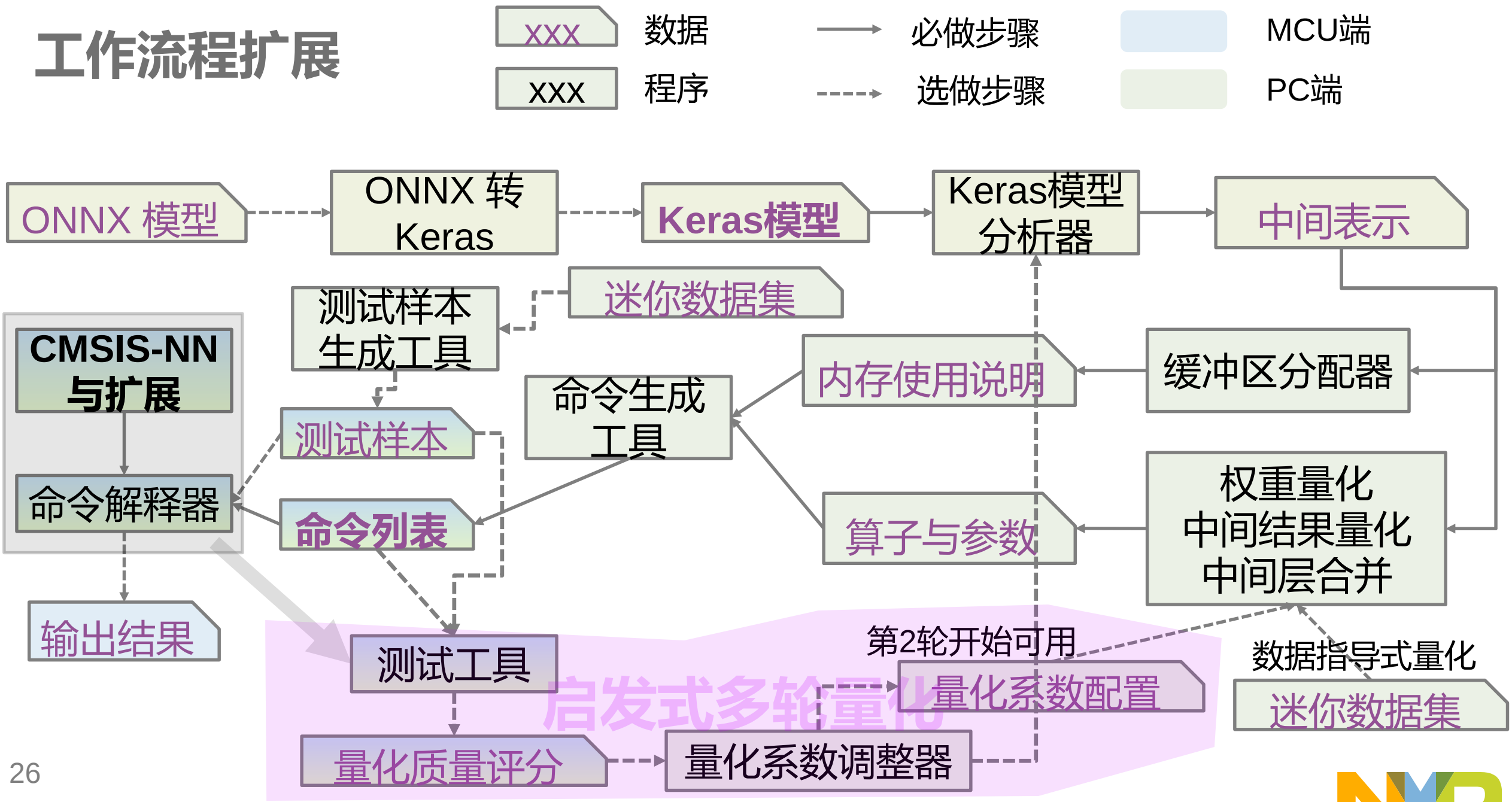
https://pan.baidu.com/s/1eQADEHg8UVhTsZWT_lRRgw

微信交流：“NNCU体验交流群”

工作流程


xxx
数据
必做步骤
MCU端
xxx
程序
选做步骤
PC端
ONNX 模型
ONNX 转 Keras
Keras模型
Keras模型分析器
中间表示
迷你数据集
测试样本生成工具
CMSIS-NN与扩展
内存分配说明
缓冲区分配器
测试样本
命令生成工具
命令解释器
命令列表
算子与参数
权重量化
中间结果量化
中间层合并
输出结果

工作流程扩展

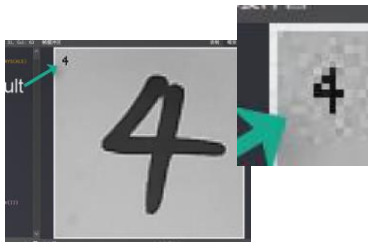

xxx
数据
xxx
程序
必做步骤
选做步骤
MCU端
PC端
ONNX 模型
ONNX 转 Keras
Keras模型
Keras模型分析器
中间表示
迷你数据集
测试样本生成工具
CMSIS-NN与扩展
内存使用说明
缓冲区分配器
命令生成工具
测试样本
命令解释器
命令列表
算子与参数
权重量化
中间结果量化
中间层合并
输出结果
测试工具
第2轮开始可用
数据指导式量化
量化系数配置
迷你数据集
启发式多轮量化
量化质量评分
量化系数调整器

实例，应用与支持

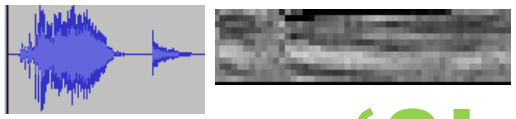
MCU AI/MV 演示一览

10/100 分类

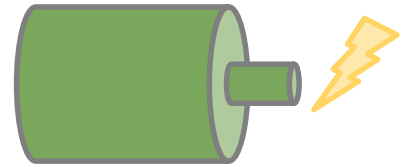
人脸检测与识别



手写数字识别



语音口令识别 ‘8’

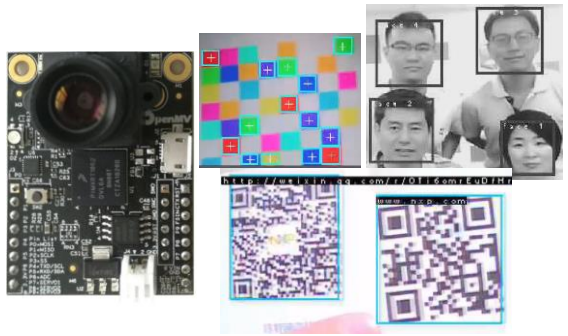


电机控制异常检测
运动/震动异常检测

红细胞感染疟疾检测

基于面部的性别识别

石头剪刀布



OpenMV-RT 多项演示

模型的类型

分类

- 识别10类物体，手写数字，语音口令

回归

- 性能识别，年龄估算

阴性/阳性 判别

- 疟疾感染，性别检测

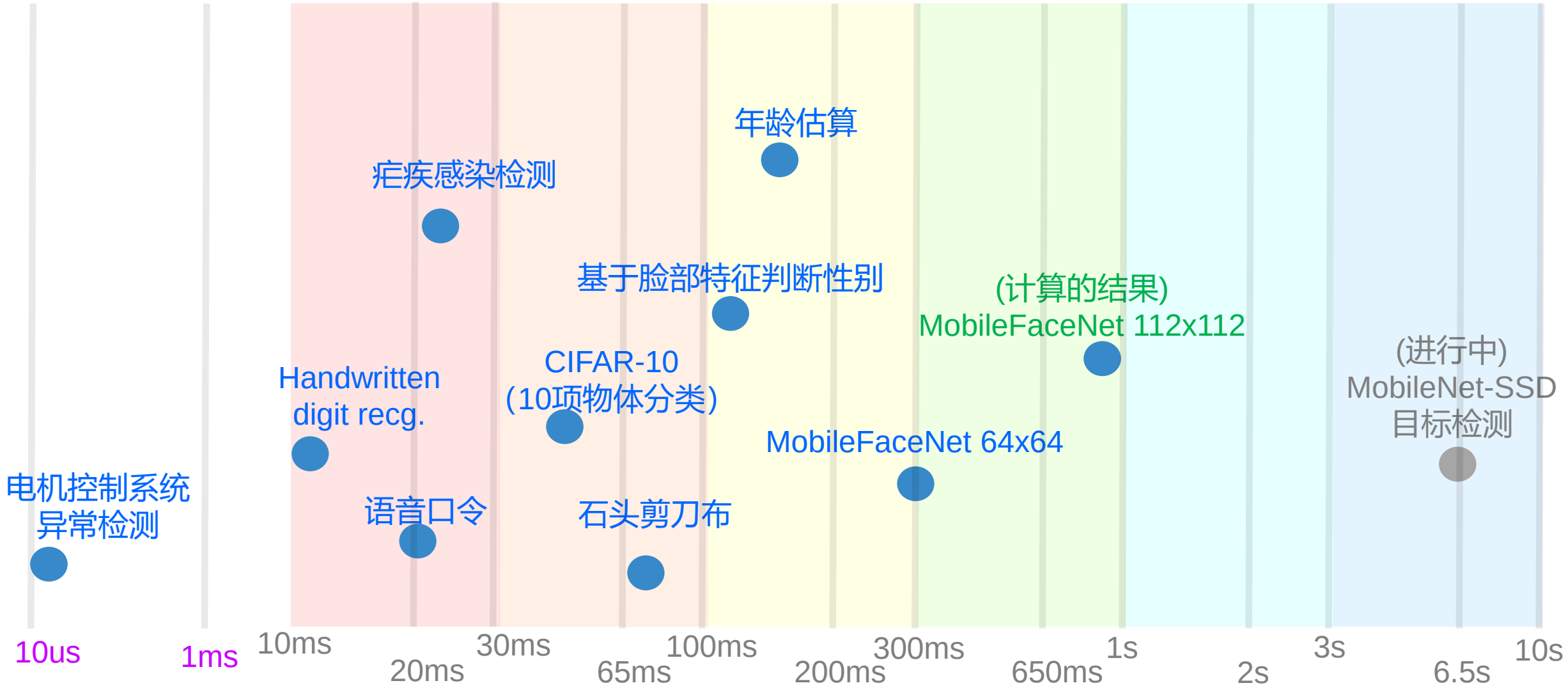
一对向量相似度比较

- 人脸识别

重建输入向量

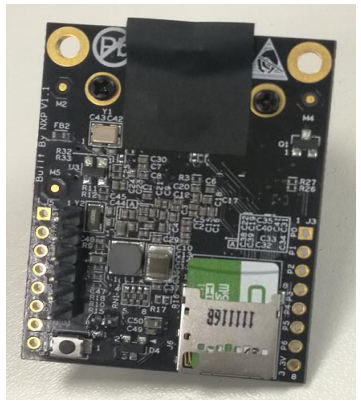
- 电机控制异常检测，震动异常检测

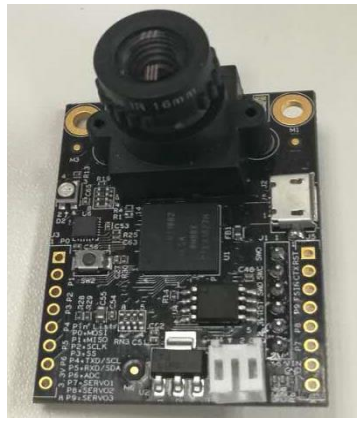
i.MX RT1050上demo模型耗时

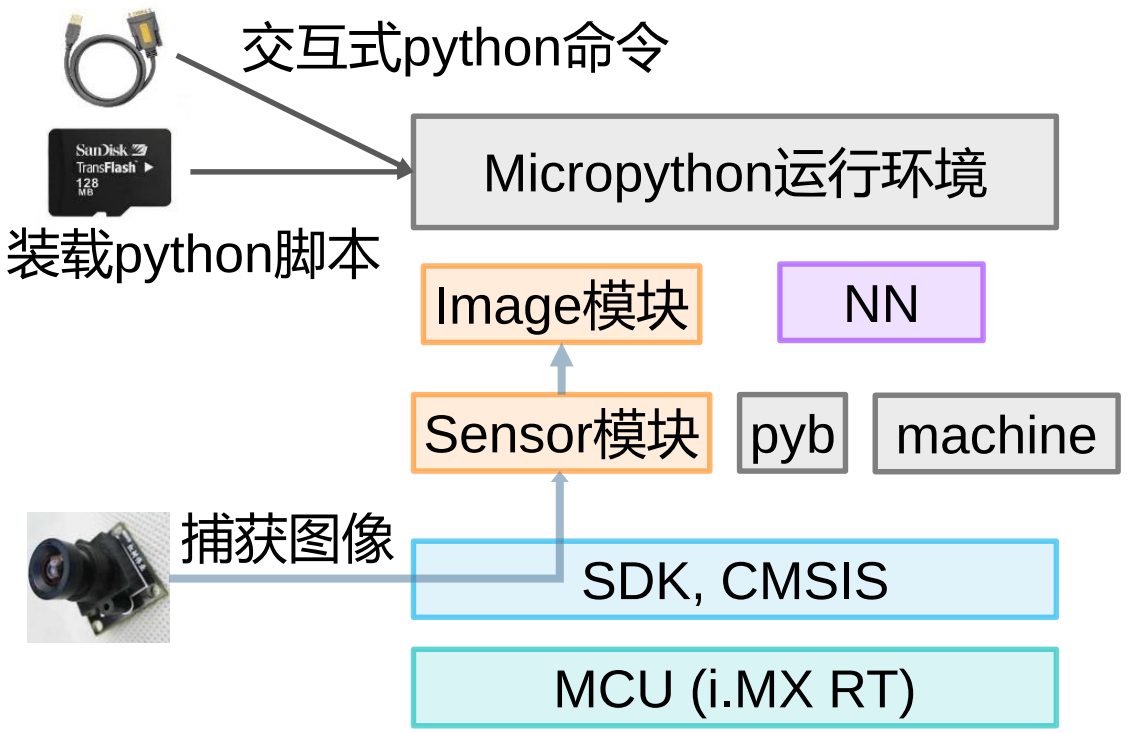


可编程视觉模块

- 基于NXP i.MX RT1062
- OV7725 + M12系列镜头
- 高度兼容OpenMV Cam
- 使用Micropython开发
- 一键部署脚本到RAM
- 支持多种机器视觉算法
- 支持神经网络的运算
- 板载9轴运动传感器
- 可通过SPI, I2C, UART连接外设或底板



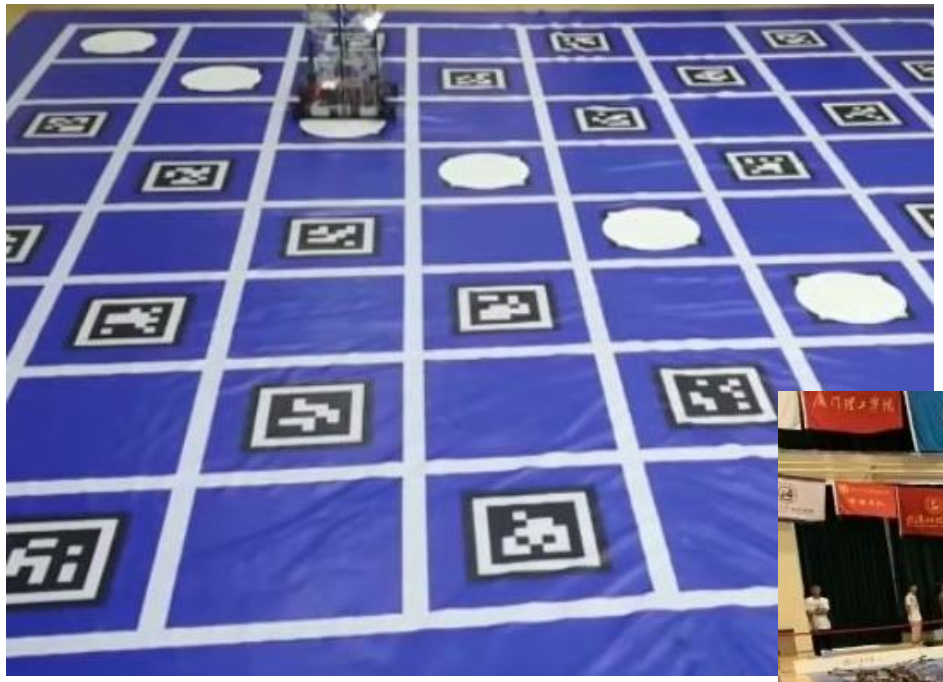




项目地址:
https://github.com/RockySong/micropython-rocky

视觉模块在2019年全国大学生智能车竞赛中的应用

- April tag
  - 标签识别与定位
- 色块
  - 信标灯识别
- 线条与形状
  - 赛道识别
- 图像预处理
  - 有助于后续处理
- 通用MCU功能



展望：机器学习用于智能车和AGV自动驾驶

- 控制算法 -> AI模型
- 第1阶段：离线训练，在线使用
  - 收集数据：场景+命令
  - 离线训练模仿模型
  - 模仿但难以超越人类
- 第2阶段：离线+在线强化学习
  - 离线训练赛道感知模型
  - 在线训练驾驶策略模型
  - 可以超越人类

NXP器件的使用

i.MX 8m

- 图像导引
- 雷达导引

i.MX RT

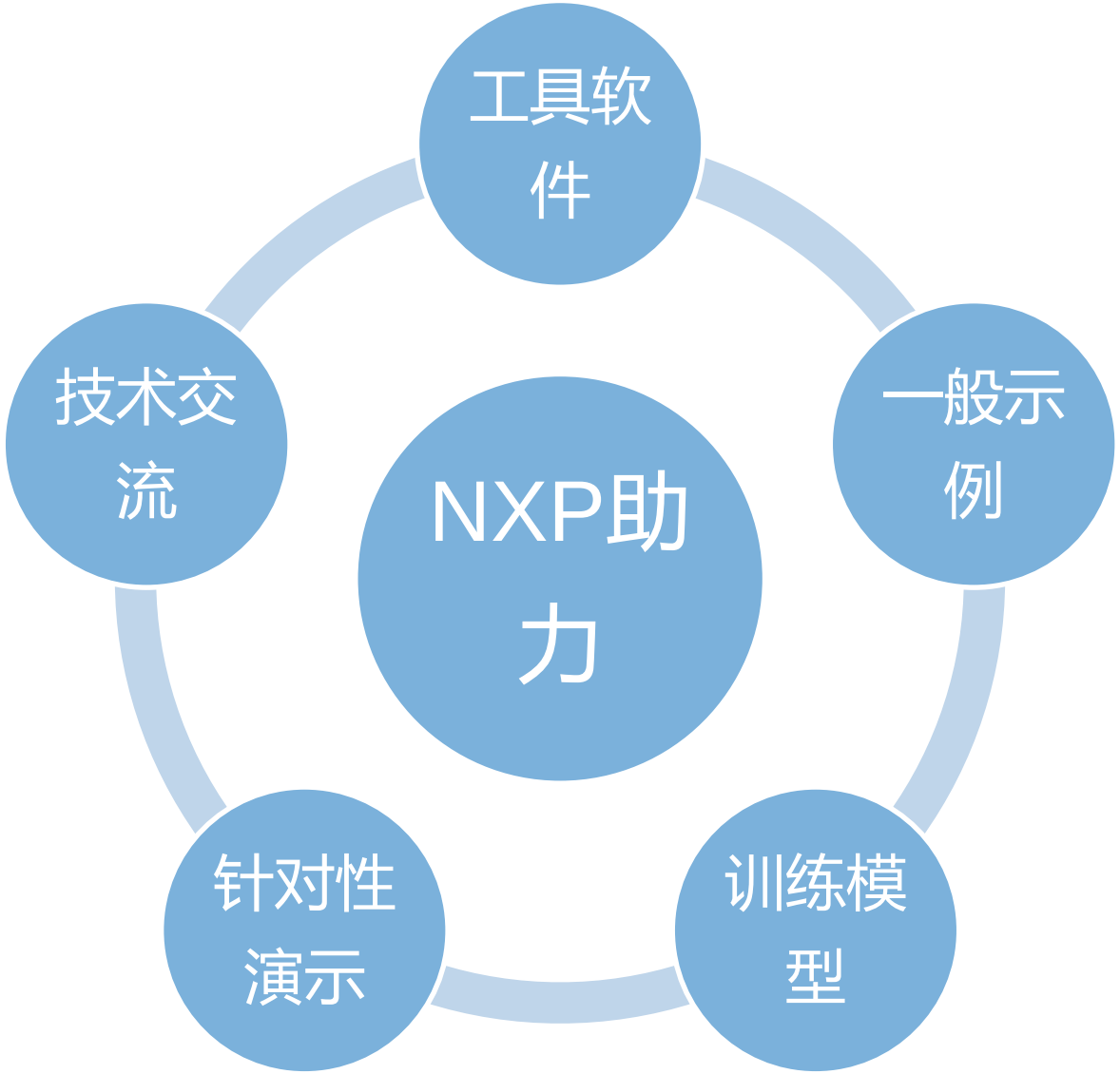
- 电磁导引
- 传感器、信标辅助

智能车自动驾驶的推荐平台：i.MX RT与i.MX 8M

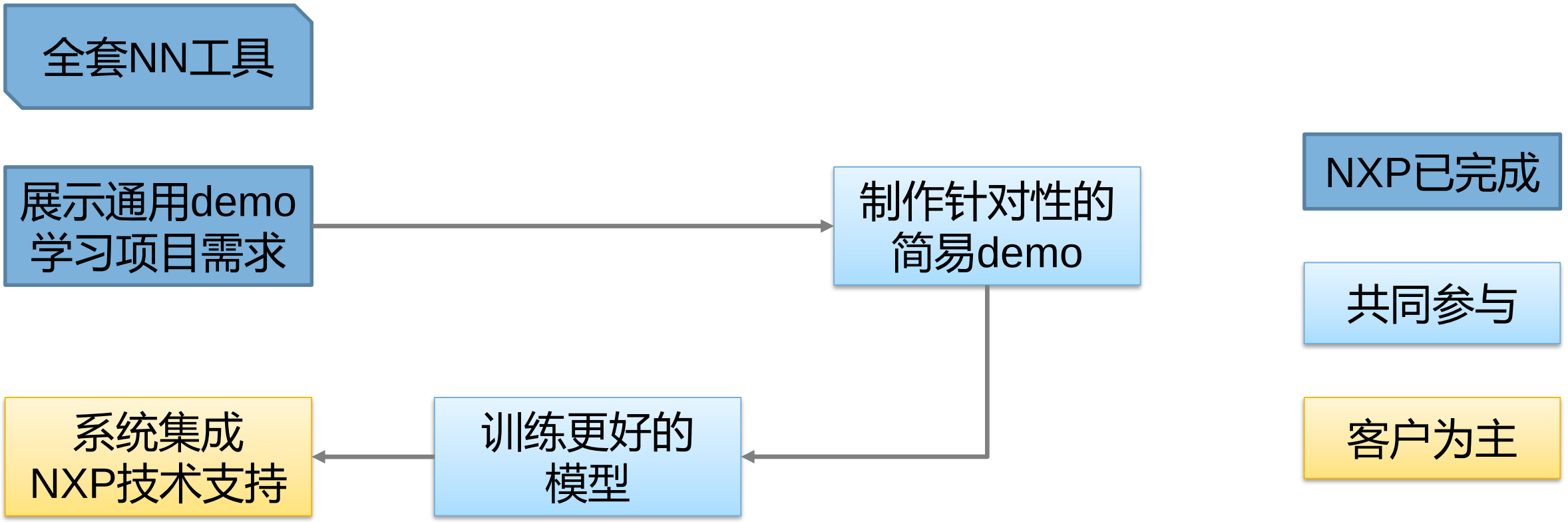
i.MX RT (MCU)	i.MX 8M (MPU)
M7,M7+M4 @500M-1G	4xA53 @1.8G，GPU
精确的实时控制	强大的综合算力
C/C++或Micropython开发	Linux下开发
中小型AI模型，视觉模块	大中型AI模型
专用AI/MV工具	通用或专用AI/MV工具

受NXP eIQ环境支持

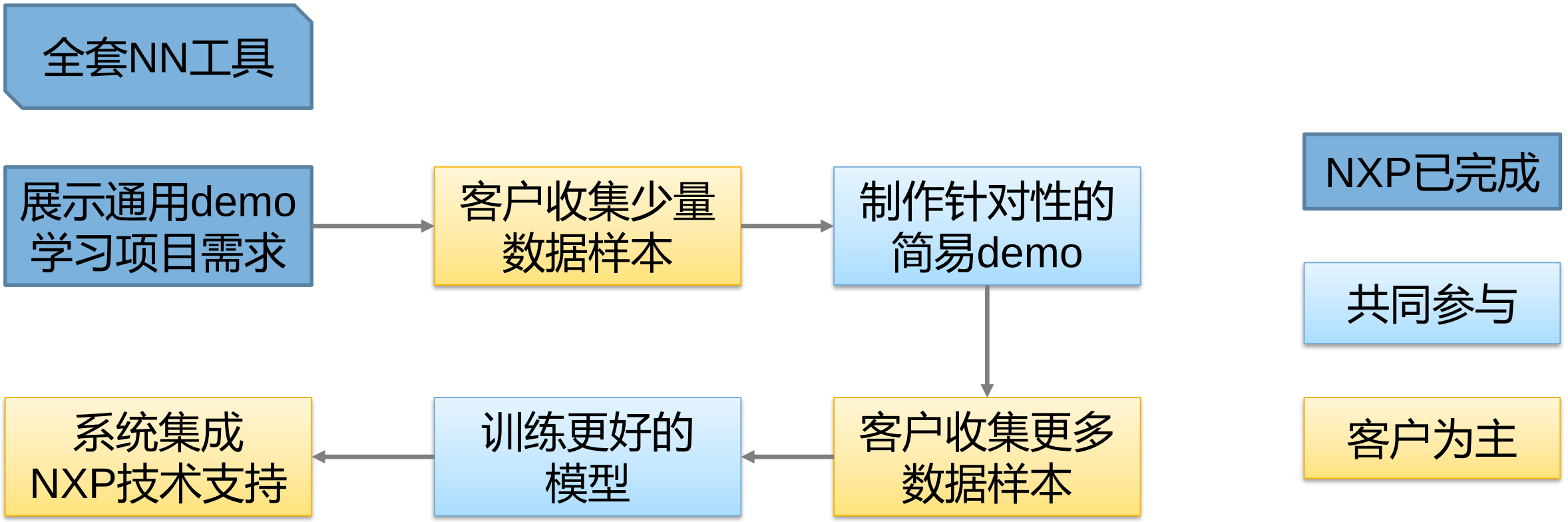
携手前进，共创辉煌



我们如何支持客户的MCU+AI项目（通用数据集）

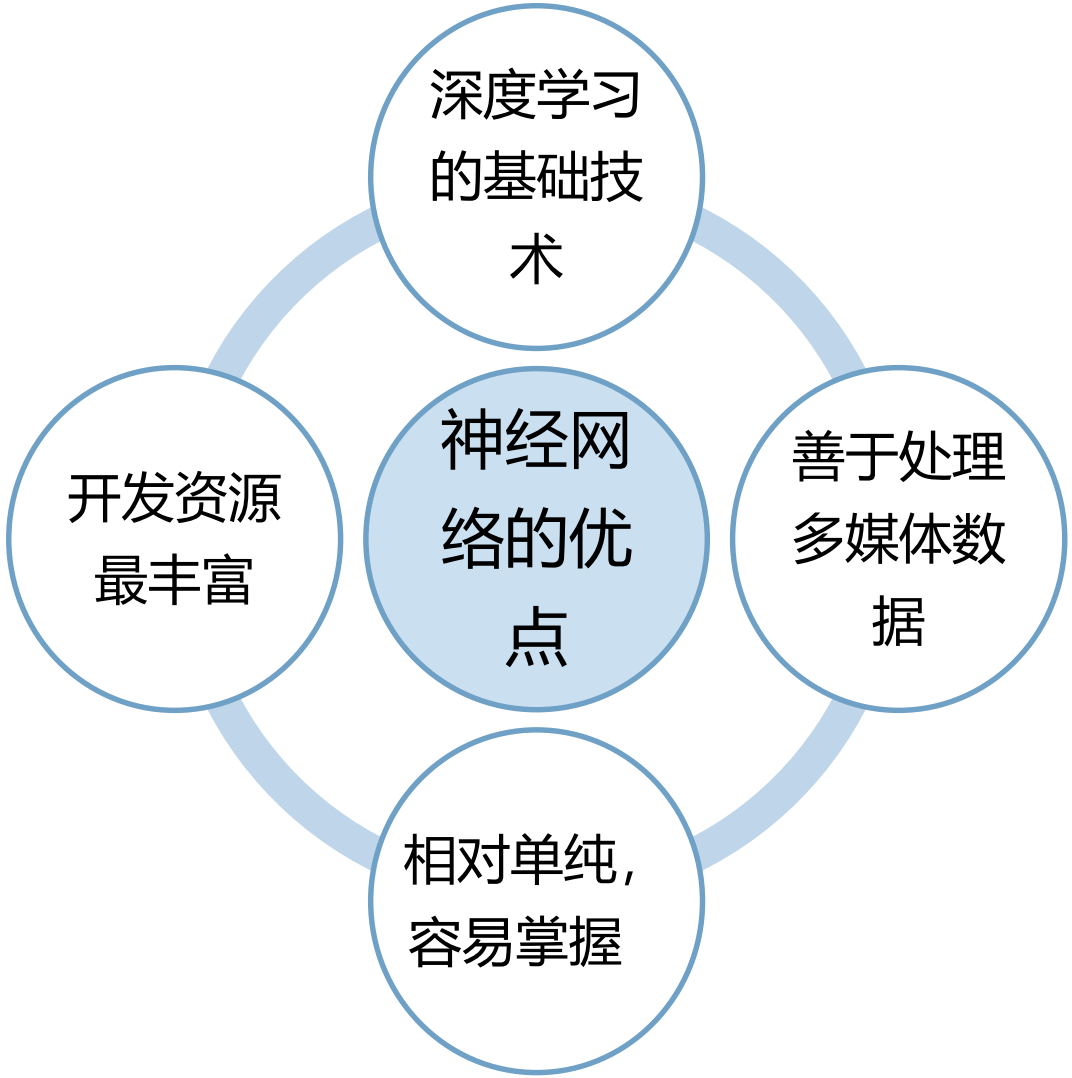


我们如何支持客户的MCU+AI项目（领域专用数据集)



使用神经网络建模

MCU上基于神经网络(NN)建模的优点

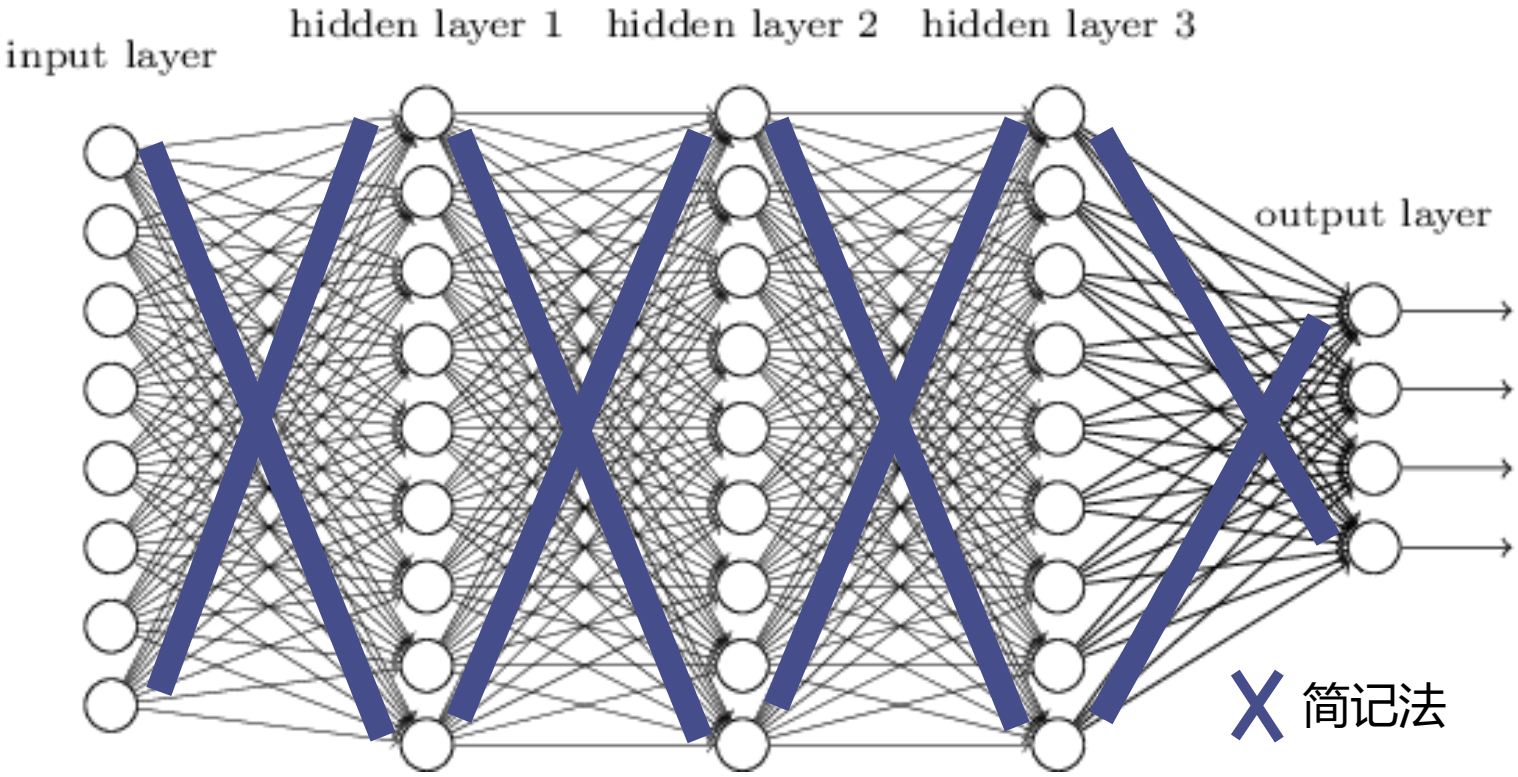


(深度) 人工神经网络

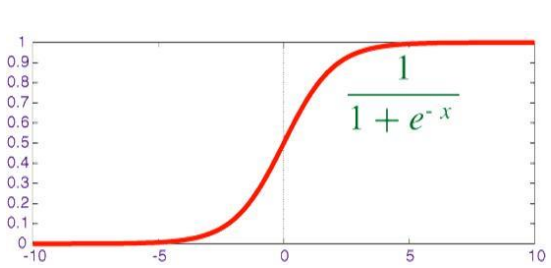
- 简称“神经网络”，通过向量内积和层次连接模拟大脑工作
- 解决现代ML/DL问题的最热建模手段，“连接派”的法宝
- 人工神经元按“层”组织，层有先后关系，之间的神经元互相带权连接
- 对于N维输入向量，输入层有N个神经元
  - 例1, 对于处理20笔3相电流数据的模型，输入层有60个神经元
  - 例2, 对于处理64x64 RGB图片的模型, 输入层有多达64*64*3=12288个神经元!
- 传统NN的参数按平方增长，不方便处理高维非结构化数据

全连接/稠密(FC/DNN) 神经网络

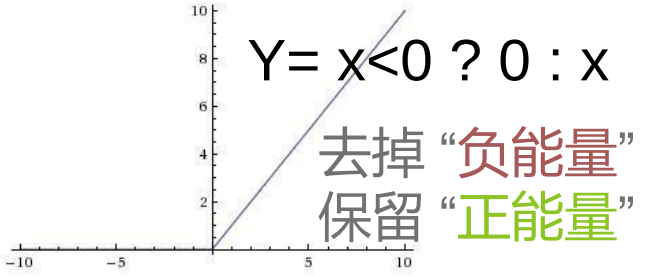
- 又称为”内积(IP)层”，“密集(Dense)层”
- 各个神经元连接到上层的全部神经元
- 各个连接的重要性（权重）可以不同
- 神经元函数： f(**WX** + b)
  - **W**,**X** 都是向量; b是标量
    - **W** 是权重向量
    - **X** 是输入向量，来自上层
    - b 是偏置标量
  - f 是 “激活函数”，简称“激活”，常用于添加表达非线性的能力.
  - 流行的激活有”S”型函数sigmoid, tanh；和”ReLU”。它们都非常容易求导数——训练的关键操作。
- $O(n^2)$ 复杂度和参数量!



X 简记法



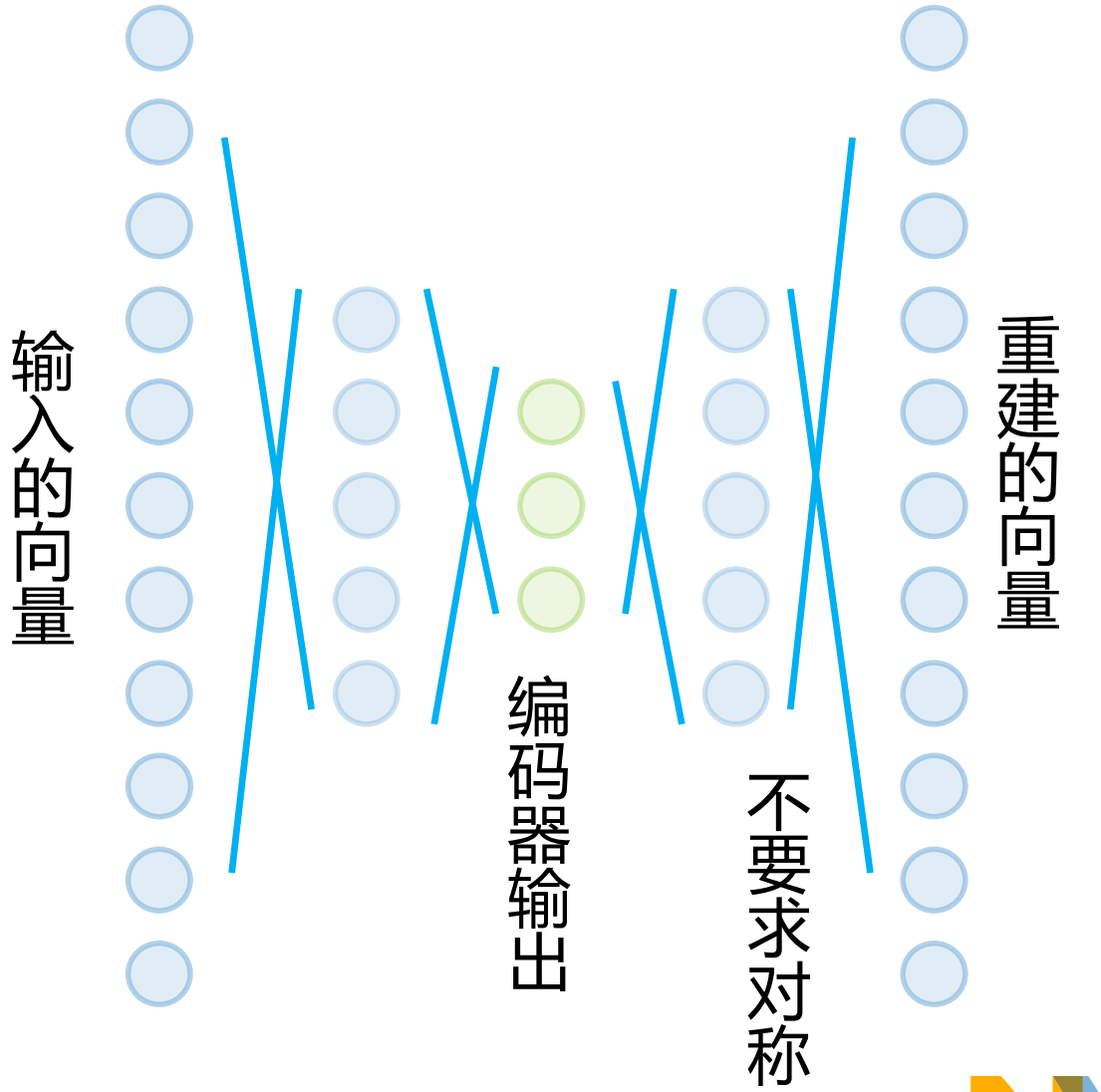
Sigmoid激活函数



ReLU 激活函数

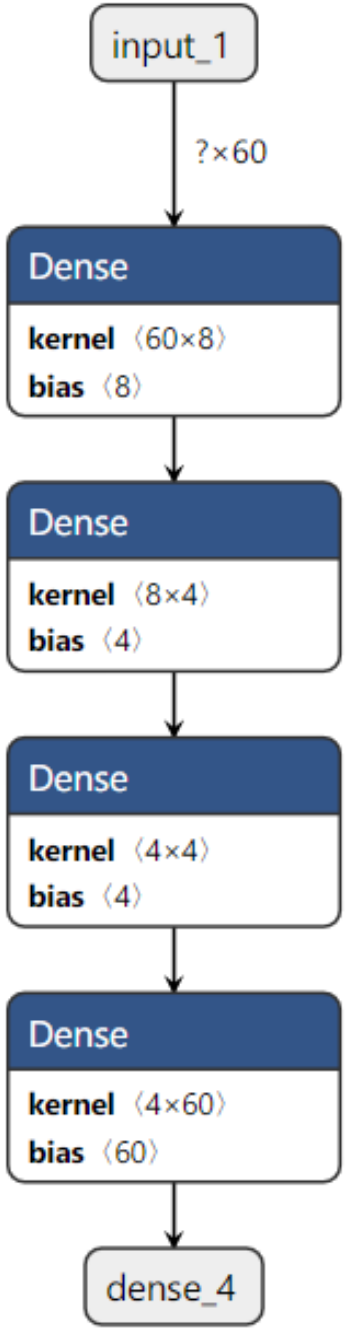
基于全连接/稠密神经网络的异常检测 – Autoencoder

- 形似“平躺的沙漏”：不要求轴对称
- 输入与输出**同维度**
  - 输入层：实测的数据
  - 输出层：重建的数据
- 训练目标：减小**重建误差**
- 只用**正常数据训练**
- 重建误差大的就当作异常数据



电机控制系统异常检测

- 基于autoencoder
- 数据8位友好化预算理
- 检测缺相等多种异常
- 缺相检测精度可达100%
- 模型不足2KB (60->8->4->4->60)
- i.MX RT1010上运行耗时15us以内

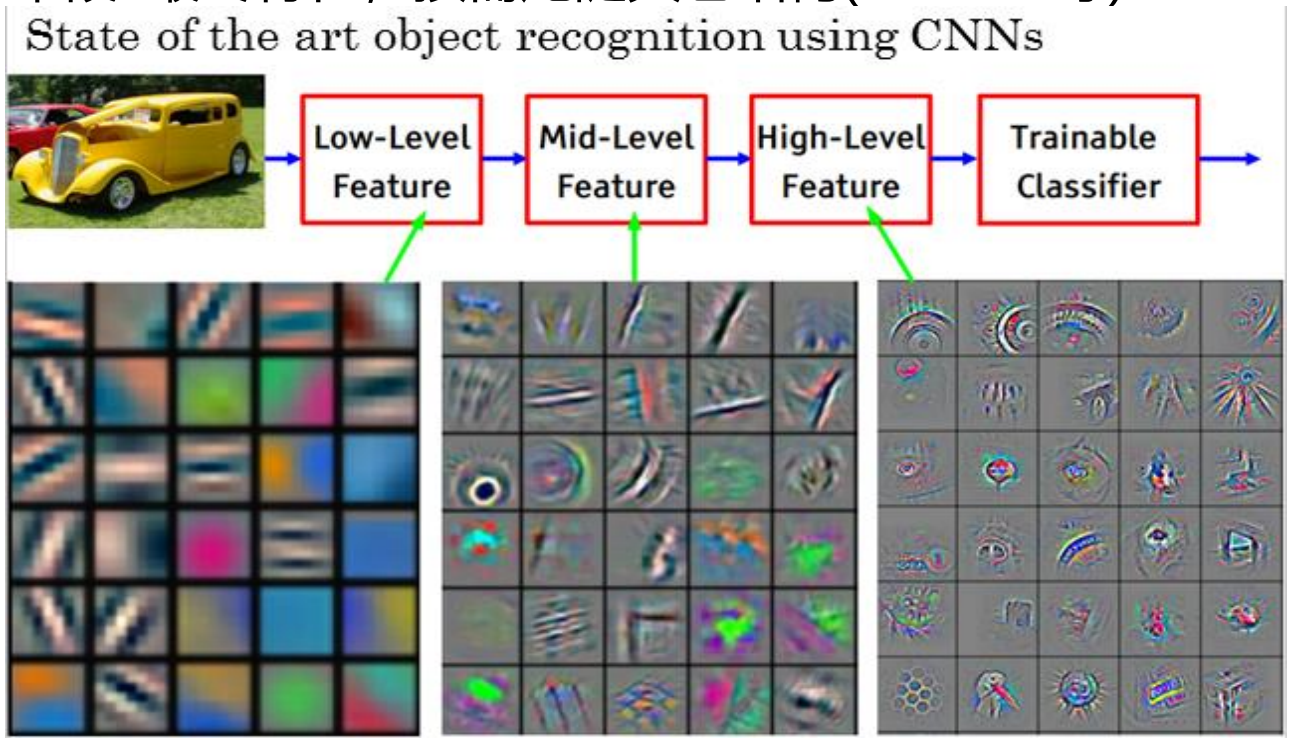


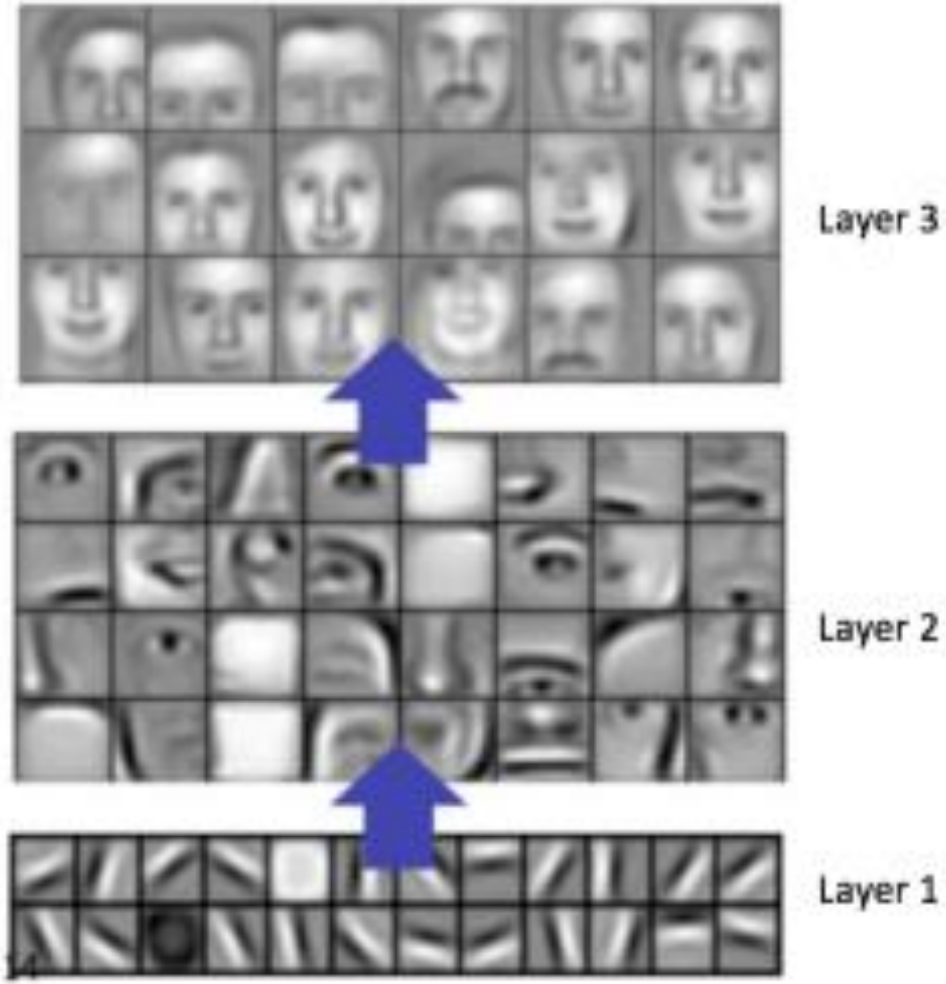
卷积神经网络(CNN)

- **必“考”**：几乎可用于所有（复杂）模式识别
- 特征**“精炼厂”**：提炼特征我第一
- **混搭**王：与各种其它网络配合
- 易**加速**：可化为矩阵相乘
- CNN的技术要点
  - **专心致志，心无旁物**：（只感知上层很小的邻近位置，常见1x1, 3x3大小)
  - **去粗取精，大浪淘沙**：一般与向下采样(池化层)配合以提炼特征
  - **由表及里，环环相扣**：多个CNN层相连逐步精炼

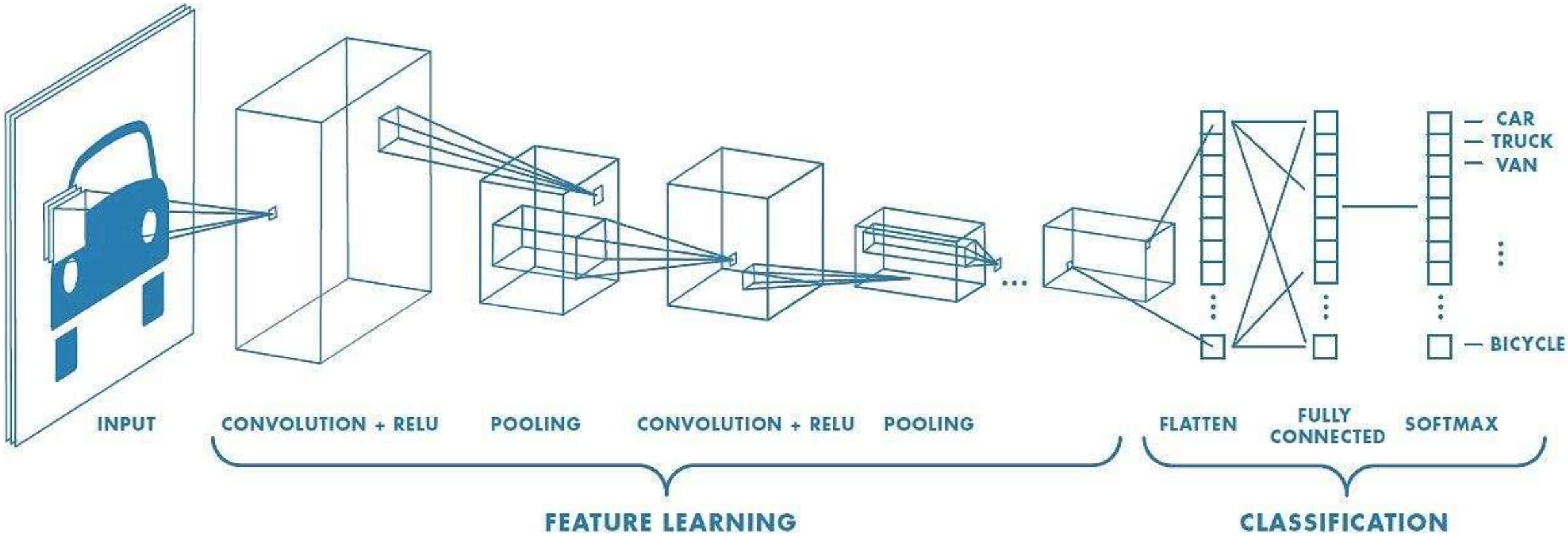
CNN精练特征的形象演示

- 精练的过程是层层相扣，逐步完成的.
  - 前段: 初级特征，例如边缘，色块
  - 中段: 中级特征，例如 线条、局部形状等
  - 后段: 高级特征，例如关键区域，甚至整体描述
  - 末段: 最终特征，按需尾随其它结构(FC/SSD等)





卷积神经网络用于分类示意图



卷积层

全连接层

适合在MCU端使用的NN基本构建块

主运算部

- 普通卷积层 (CNN)
- CNN按空间与通道分解 (DS-CNN)
- 全连接层 (Dense/FC/IP)

基本构建块

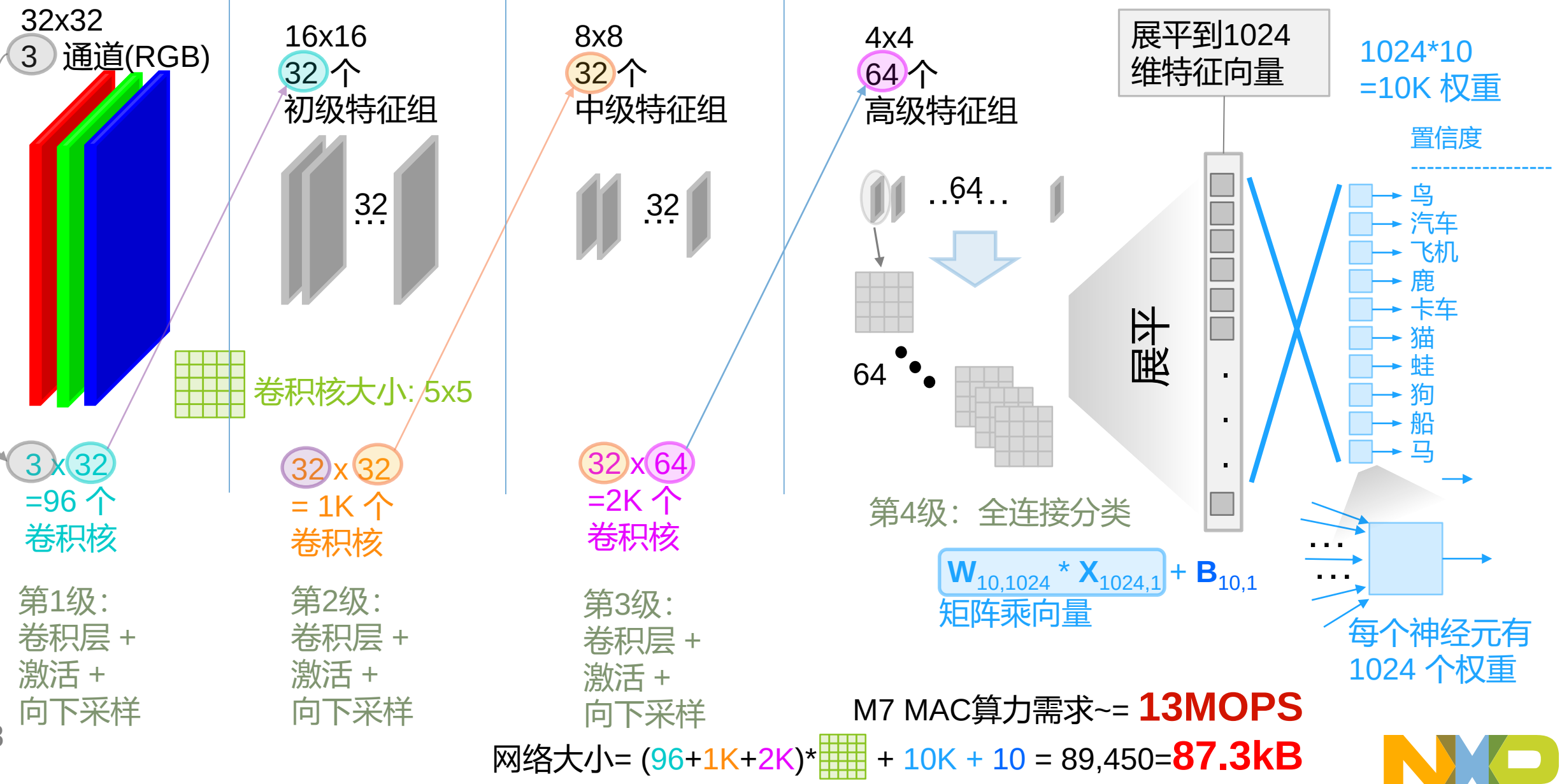
先做主运算

常常再做后加工

(多数情况)后加工

- 激活 - 表达非线性关系
- 下采样/池化 - 精炼特征
- Softmax - 份量转成概率

一个微型CNN分类网络实例的结构



构建块的常用搭建方式

串联/直筒式：一层一层往上迭

- 最简单的结构，如同“糖葫芦”。微型/小型网络的首选

直筒 + 跳跃：某层的输出又加到后续几层后

- 胜任较复杂的问题，如人脸识别。如MobileNet V2

化整为零: 大型构建块做并联/串联分解

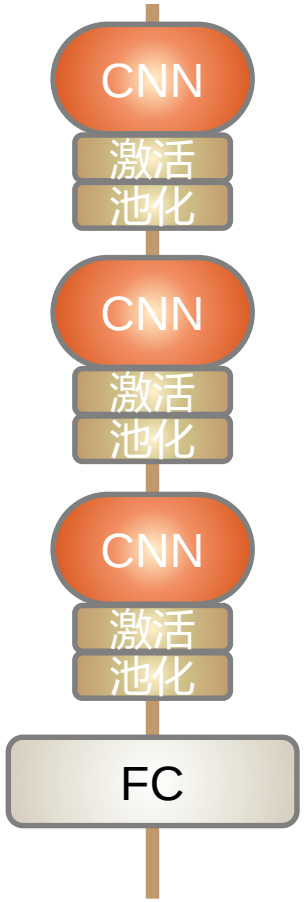
- 处理大型问题，在MCU上耗时较长。如各代inception

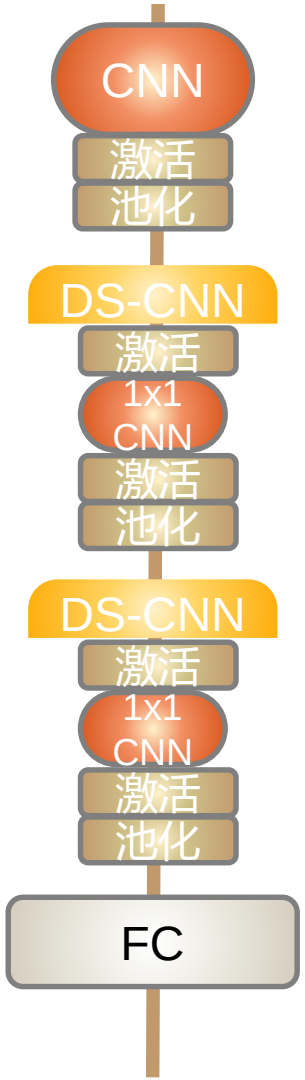
预制复合结构

- 若干层构建块组成一个复合单元，后者再串在一起: 简化设计，灵活多用

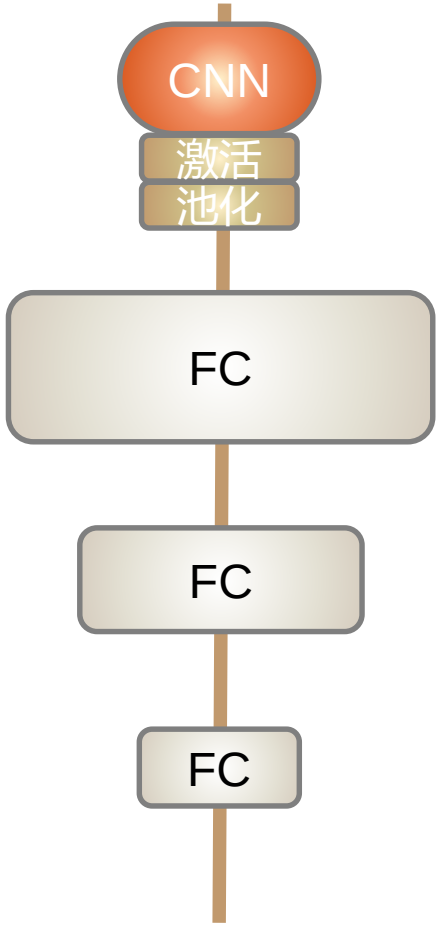
适合MCU的小型神经网络常见结构

典型CNN

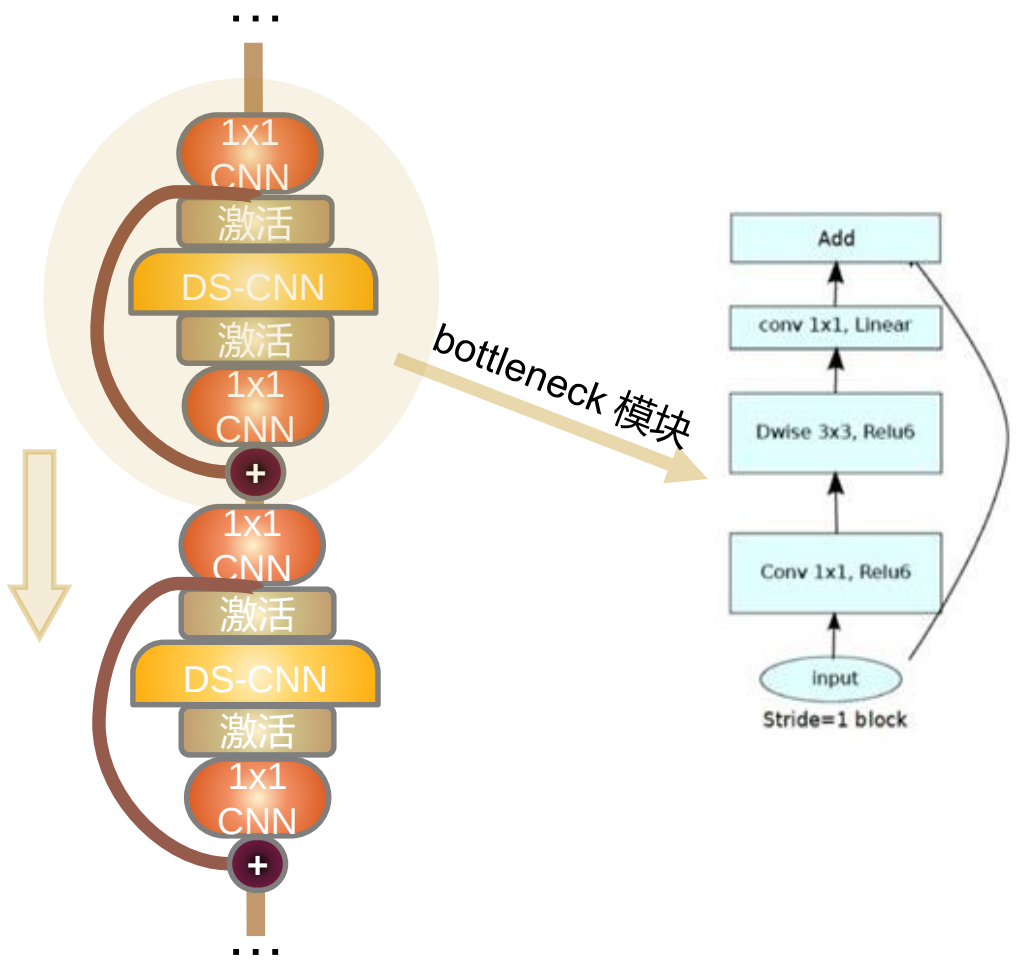




空间要求低



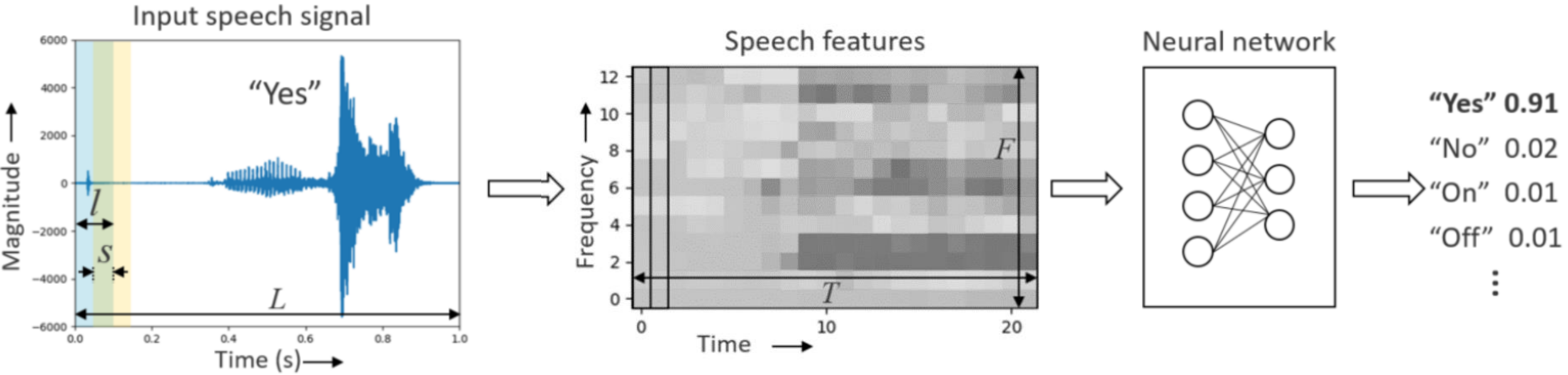
算力要求低



更深的网络

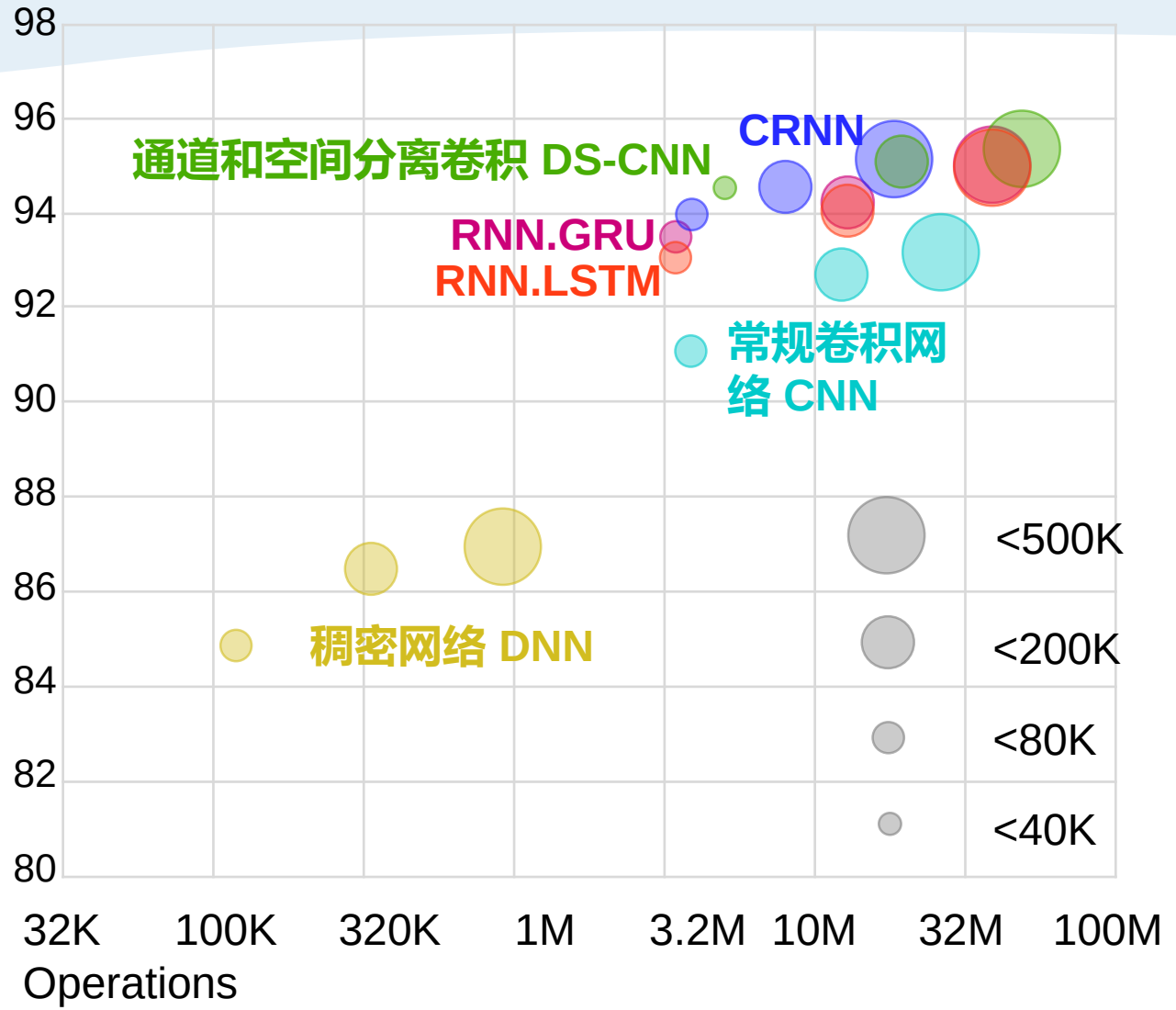
使用DL实现话音口令检测 (KWS)

- 接收短口令，并映射到预定义的命令
- 广泛用于语音控制的系统中，作为永久在线的低功耗唤醒模块
- 把时域信号分割变换成多段频谱，形成灰度图，再应用NN技术



关键词检测模型的资源需求

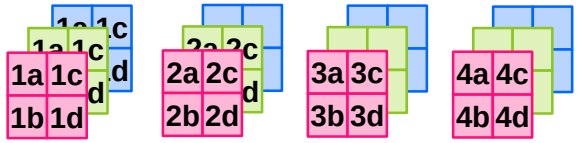
- 显著的边际效应递减
- 全连接神经网络(DNN) 精度最低，但计算量也最低.
- 常规CNN性能稍有不足
- **DS-CNN**有效减少了模型尺寸和提高性能，只是对算力要求有所提高.
- RNN比常规CNN效果好, CRNN融合了CNN与RNN的优点.



HWC格式下某Conv层权重示例：3输入通道，4输出通道

无论W,H取何值，都不影响卷积核的参数数量
数量 = Cin * Kx * Ky * Cout = 3*2*2*4=48

2x2卷积核阵列
对于同一输出通道：
每个输入通道有一个
共Cin * Cout个核



真实情况下输入通道可达16-1280
在设计网络时，尽量避免使一层的参数总量明显超过Dcache容量
例2：Cin=64, Cout=128, 3x3大小：64*3*3*128=72k 个参数，性能无保障

输出通道号 1 2 3 4

1 2 3 4 每个输出通道有一个偏置项(bias)

1号输出通道	2号输出通道	3号输出通道	4号输出通道
1a 1a 1a 1c 1c 1c 1b 1b 1b 1d 1d 1d	2a 2a 2a 2c 2c 2c 2b 2b 2b 2d 2d 2d	3a 3a 3a 3c 3c 3c 3b 3b 3b 3d 3d 3d	4a 4a 4a 4c 4c 4c 4b 4b 4b 4d 4d 4d

HWC格式下，卷积核参数在内存中的布局（地址从低 -> 高）

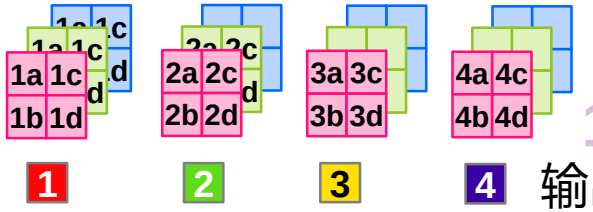
1号输出通道	2号输出通道	3号输出通道	4号输出通道
1a 1c 1b 1d 1a 1c 1b 1d 1a 1c 1b 1d	2a 2c 2b 2d 2a 2c 2b 2d 2a 2c 2b 2d	3a 3c 3b 3d 3a 3c 3b 3d 3a 3c 3b 3d	4a 4c 4b 4d 4a 4c 4b 4d 4a 4c 4b 4d

CHW格式下，卷积核参数在内存中的布局（地址从低 -> 高）

HWC格式下的2D卷积运算

2x2卷积核阵列
对于同一输出通道:
每个输入通道有一个
共Cin * Cout个核

卷积核(权重)



输入通道 1 2 3

1 2 3 4 输出通道的偏置

输出通道 1 2 3 4

1号输出通道 1
2号输出通道 2
3号输出通道 3
4号输出通道 4
偏置向量

+

1a	1a	1a	1c	1c	1c	1b	1b	1b	1d	1d	1d
2a	2a	2a	2c	2c	2c	2b	2b	2b	2d	2d	2d
3a	3a	3a	3c	3c	3c	3b	3b	3b	3d	3d	3d
4a	4a	4a	4c	4c	4c	4b	4b	4b	4d	4d	4d

权重矩阵: 4,12
按行分块

X

0	1	2	3	4	5	6	7	8	9	A	B
11	12	13	14	21	22	23	24	31	32	33	34
11	12	13	14	21	22	23	24	31	32	33	34
11	12	13	14	21	22	23	24	31	32	33	34
12	13	14	0	22	23	24	0	32	33	34	0
12	13	14	0	22	23	24	0	32	33	34	0
12	13	14	0	22	23	24	0	32	33	34	0
21	22	23	24	31	32	33	34	0	0	0	0
21	22	23	24	31	32	33	34	0	0	0	0
21	22	23	24	31	32	33	34	0	0	0	0
22	23	24	0	32	33	34	0	0	0	0	0
22	23	24	0	32	33	34	0	0	0	0	0
22	23	24	0	32	33	34	0	0	0	0	0

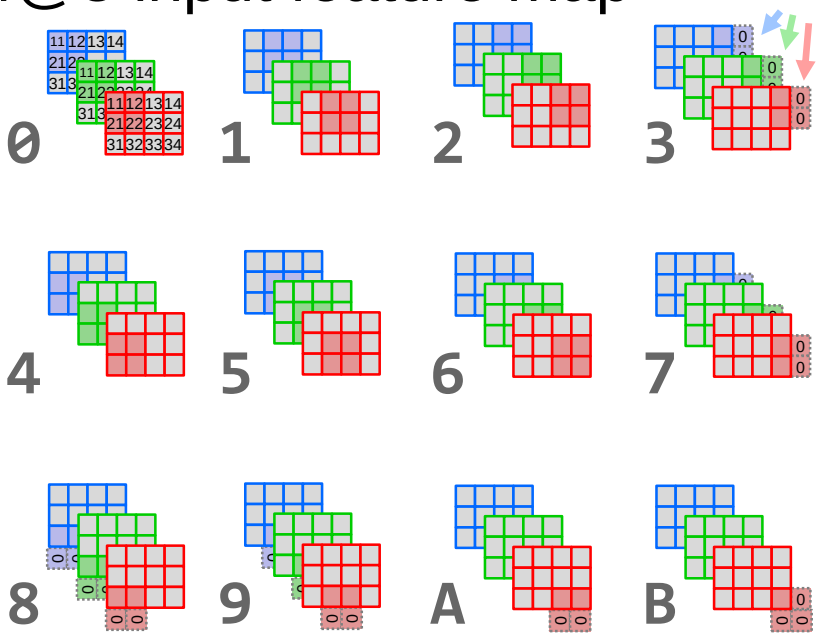
卷积窗矩阵: 按列分块
12(2x2x3), 12(3x4)

=

Output: 3x4@4
Chn=4,
H(rows)=3
W (cols)=4,

3x4@3 input feature map

padding

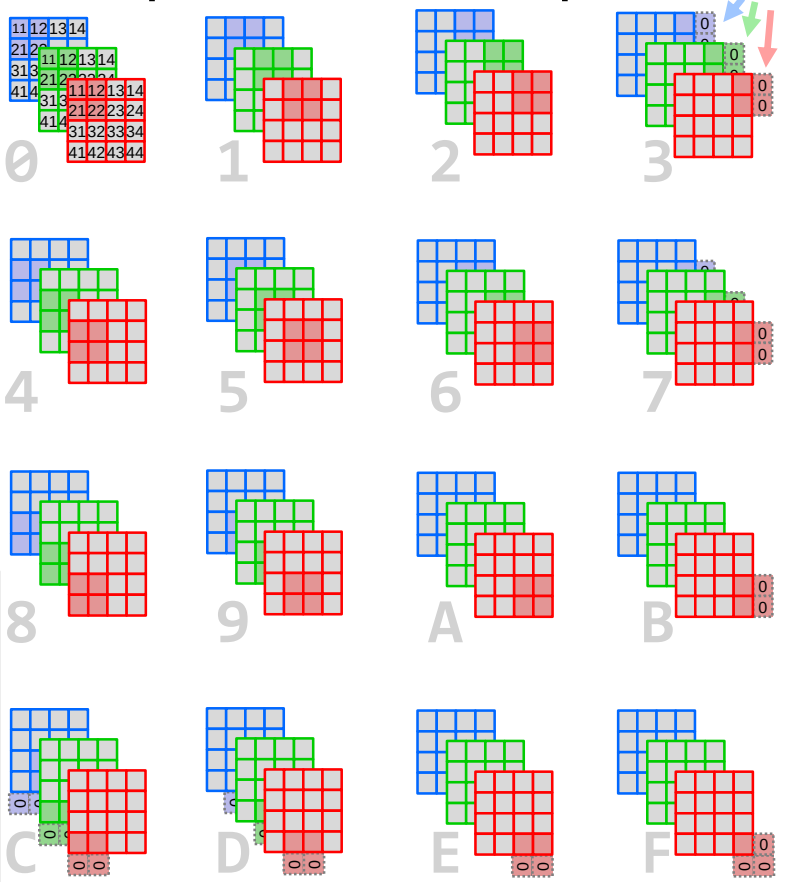


卷积窗口滑动全过程

HWC格式下的2D卷积运算 例2

4x4@3 input feature map

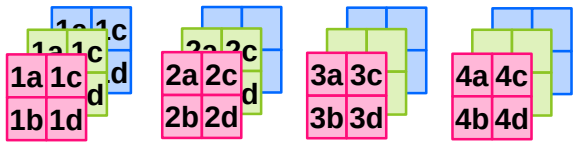
padding



2x2卷积核阵列
对于同一输出通道:
每个输入通道有一个
共Cin * Cout个核

卷积核(权重)

输入通道



1 2 3 4 输出通道的偏置

输出通道 1 2 3 4

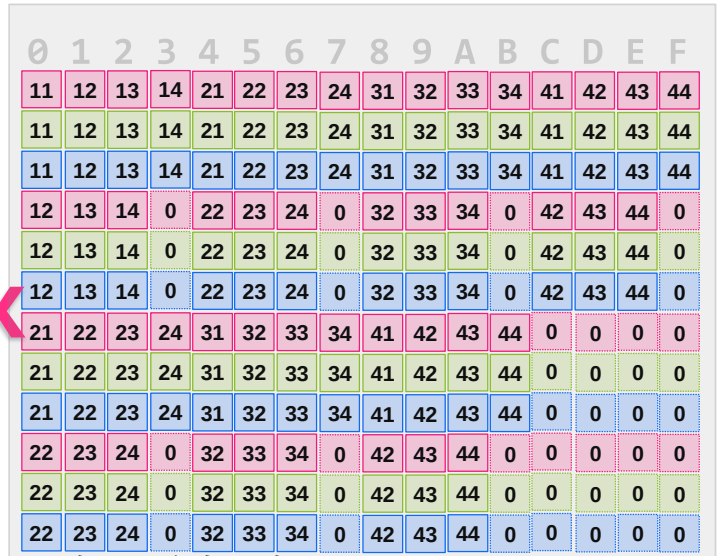
1号输出通道 1
2号输出通道 2
3号输出通道 3
4号输出通道 4

偏置向量

+

权重矩阵: 4,12

X



卷积窗矩阵:
12(2x2x3), 16(4x4)

=

Output
Chn=4, W (cols)=4,
H(rows)=4

Cortex-M DSP扩展 理想情况下2D卷积内循环算力利用率

```
LDR       R4, [R1], #4   ; col
LDR       R6, [R0], #4   ; row0
LDR       R8, [R14],#4   ; row1
SXTB16    R5, R4, ror #8
SXTB16    R7, R6, ror #8
SXTB16    R9, R8, ror #8
SXTB16    R4, R4
SXTB16    R6, R6
SXTB16    R8, R8
SMLAD     R10, R4, R6, R10
SMLAD     R10, R5, R7, R10
SMLAD     R11, R4, R8, R11
SMLAD     R11, R5, R9, R11
```

双发射
仅限于M7

8位量化

M7 MAC: 4 cycle, Misc: 7 cycle — 36%

M4 M33? MAC: 4 cycle, Misc: 12 cycle — 25%

```
LDRD      R4, R5, [R1], #8
LDRD      R6, R7, [R0], #8
LDRD      R8, R9, [R14],#8
SMLAD     R10, R4, R6, R10
SMLAD     R10, R5, R7, R10
SMLAD     R11, R4, R8, R11
SMLAD     R11, R5, R9, R11
（下一轮开始）
LDRD      R4, R5, [R1], #8   ; col
…
```

双发射
仅限于M7

16位量化

M7 MAC: 4 cycle, Misc: 2.25 cycle — ~64%

M4 M33? MAC: 4 cycle, Misc: 9 cycle — ~31%

```
LDR       R4, [R1], #4   ; col
LDRD      R6, R7, [R0], #8      ; row0
LDRD      R8, R9, [R14],#8      ; row1
SXTB16    R5, R4, ror #8
SXTB16    R4, R4
SMLAD     R10, R4, R6, R10
SMLAD     R10, R5, R7, R10
SMLAD     R11, R4, R8, R11
SMLAD     R11, R5, R9, R11
（下一轮开始）
LDRD      R4, R5, [R1], #8   ; col
…
```

双发射
仅限于M7

16位权重
8位数据

M7 MAC: 4 cycle, Misc: 4.25 cycle — ~48%

M4 M33? MAC: 4 cycle, Misc: 10 cycle — ~29%

总结与展望

- MCU + AI 大有可为
- 丰富的示例
- 易用的工具
- 敢想，敢做

SECURE CONNECTIONS
FOR A SMARTER WORLD