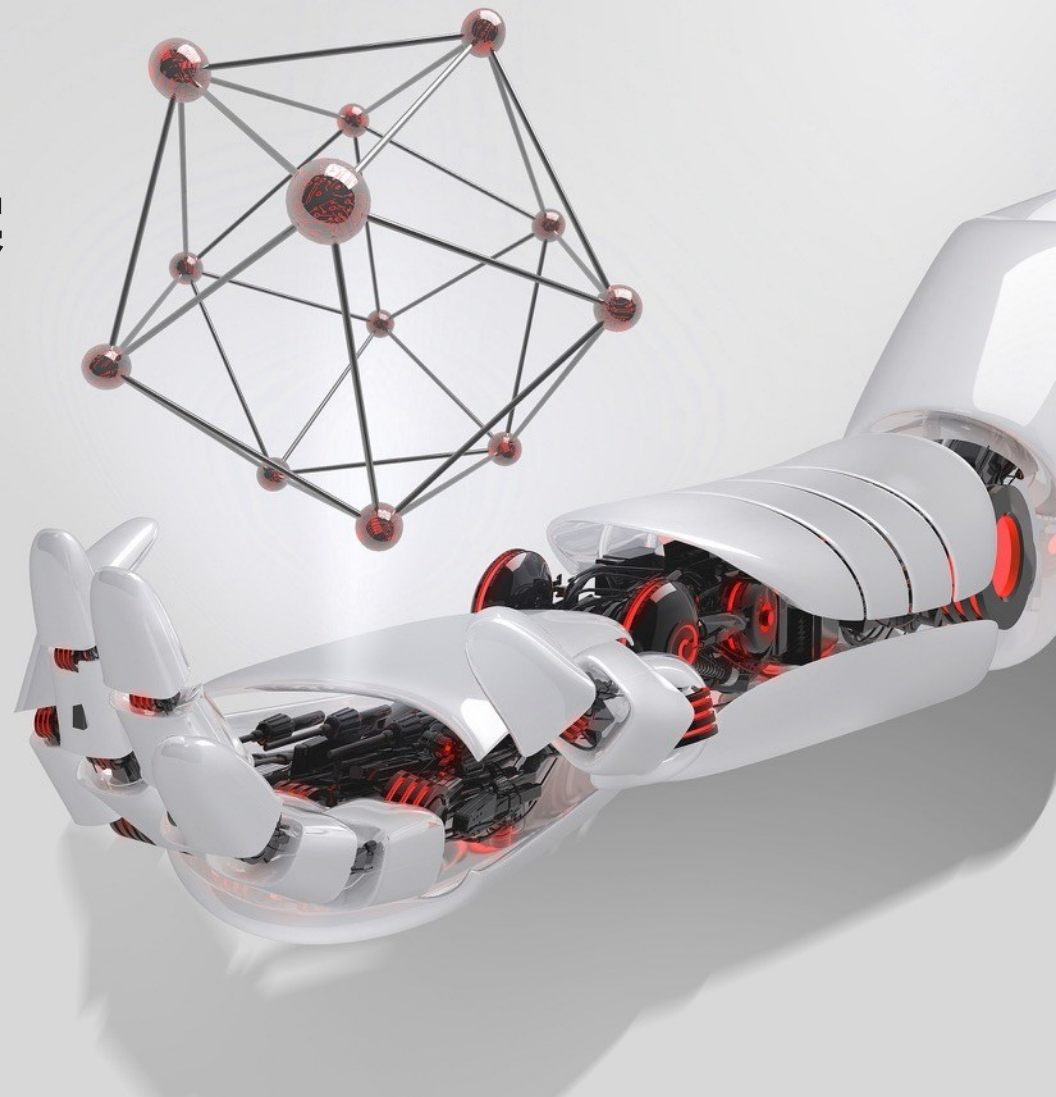


# 全国大学生嵌入式暨智能互联大赛 海思赛道赋能课件

## Hi3516DV300 SVP介绍

主讲人：陈 伟



**每一位开发者  
都是海思要汇聚的星星之火**

# SVP课程说明及入门要求

## SVP说明:

SVP全称是Smart Vision Platform智能视觉平台，是芯片内部智能各模块的统一名称，专门用于处理智能相关的任务。

智能的开发分为训练与推理两部分，Hi3516DV300是仅用于推理的端侧芯片。

## 课程入门要求:

学习本课程前需要先学习以下知识点:

- 1.卷积神经网络的概念，常用层的实现原理等
- 2.卷积神经网络的训练与部署流程
- 3.常见卷积神经网络的算法原理等

# 目录

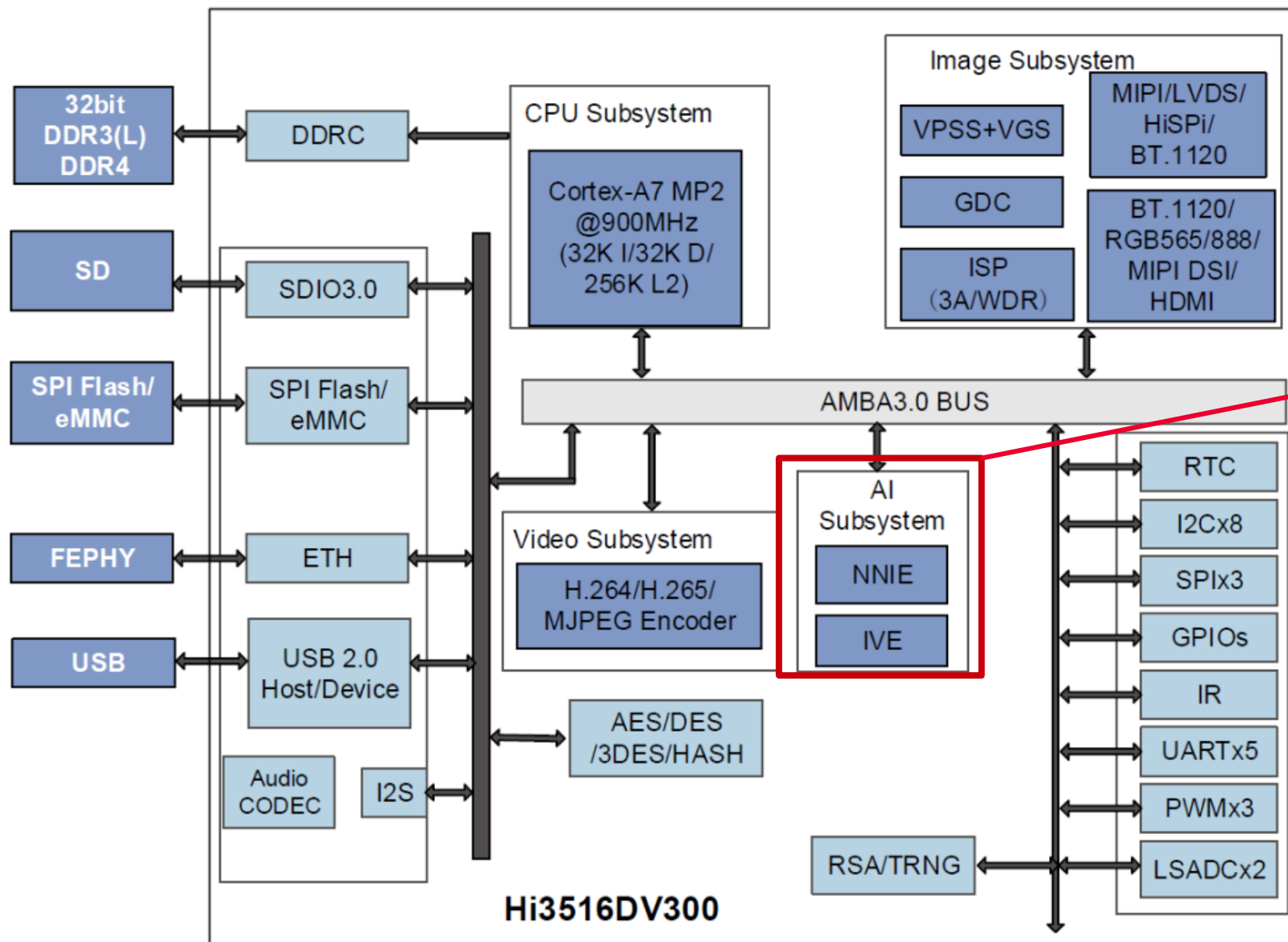
---

**1. 总体介绍**

**2. NNIE**

**3. IVE**

# Hi3516DV300 芯片架构

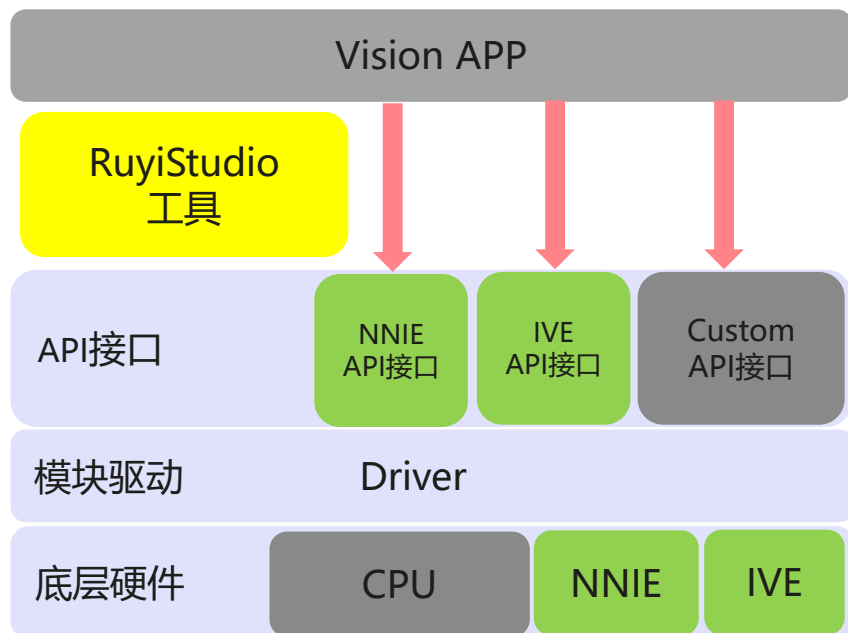


AI推理芯片

NNIE: 1Tops

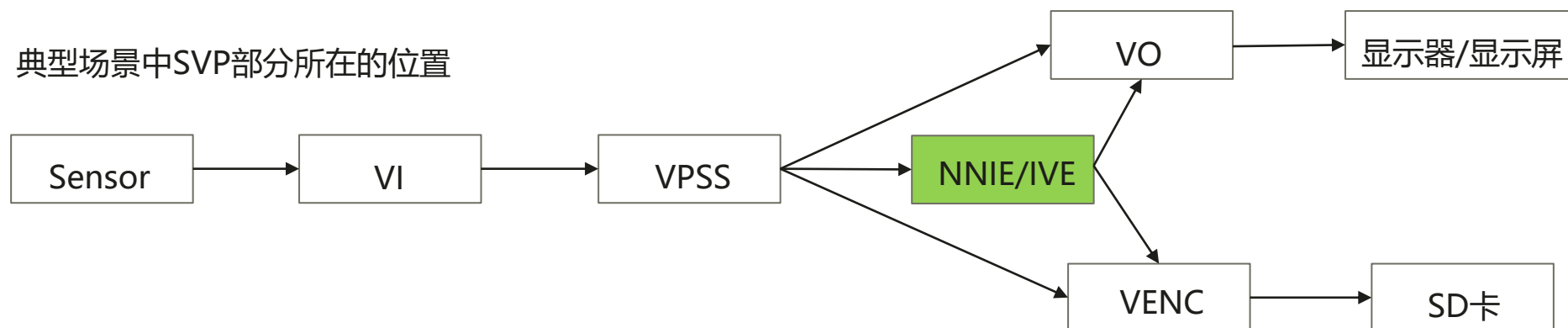
IVE: HOG,KCF,PSP

# SVP 智能开发框架

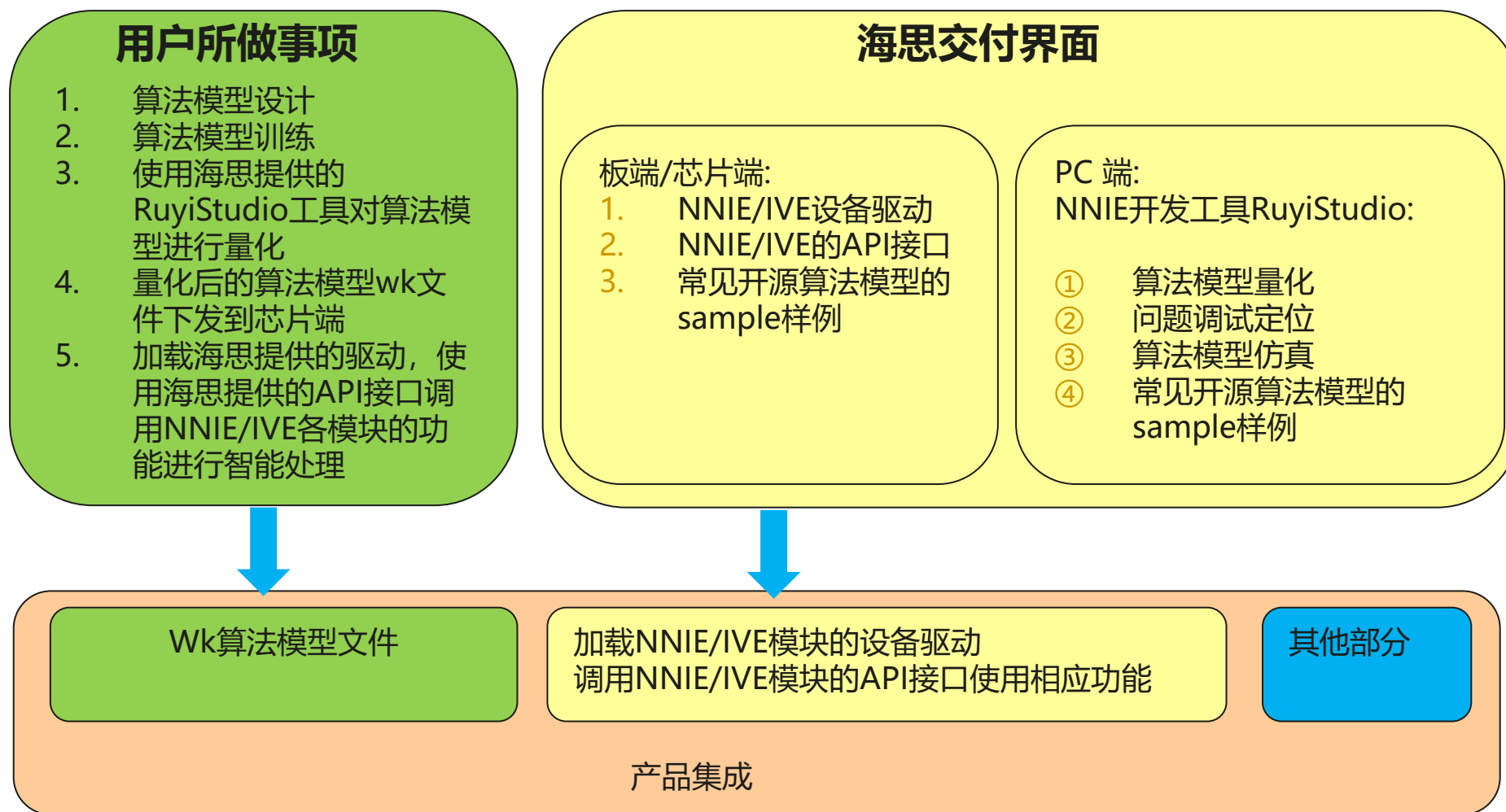


- ◆ NNIE全称Neural Network Inference Engine，用于进行卷积神经网络运算的加速引擎，例如物体的检测，识别，分割等。
- ◆ IVE全称Intelligent Video Engine，用于传统计算机视觉算法处理的加速引擎，主要用于卷积神经网络算法的预处理及后处理上，比如人脸识别算法前的PSP人脸对齐处理，以提升人脸识别算法准确率。
- ◆ RuyiStudio是NNIE的PC端开发工具，用于算法模型的量化/仿真等。

典型场景中SVP部分所在的位置



# 海思SVP交付界面



非NNIE模块的“其他部分”：用户直接调用海思提供的API接口就可以使用各模块的相应功能

# 目录

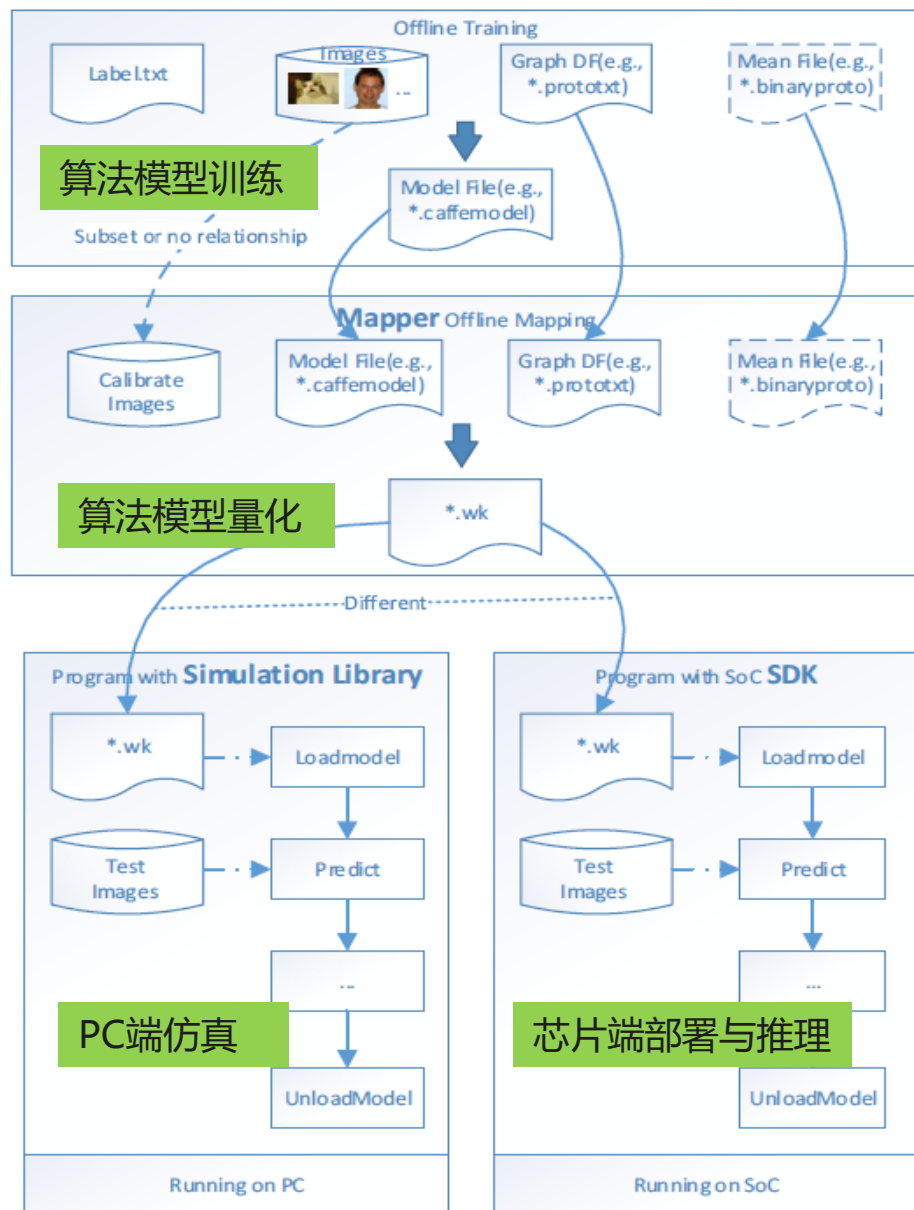
---

1. 总体介绍

2. NNIE

3. IVE

# NNIE 开发流程图



NNIE只支持Caffe 1.0框架，只支持int8或int16精度模式

如果使用非Caffe框架，需要在训练结束后先转为Caffe框架，再参考这个开发流程

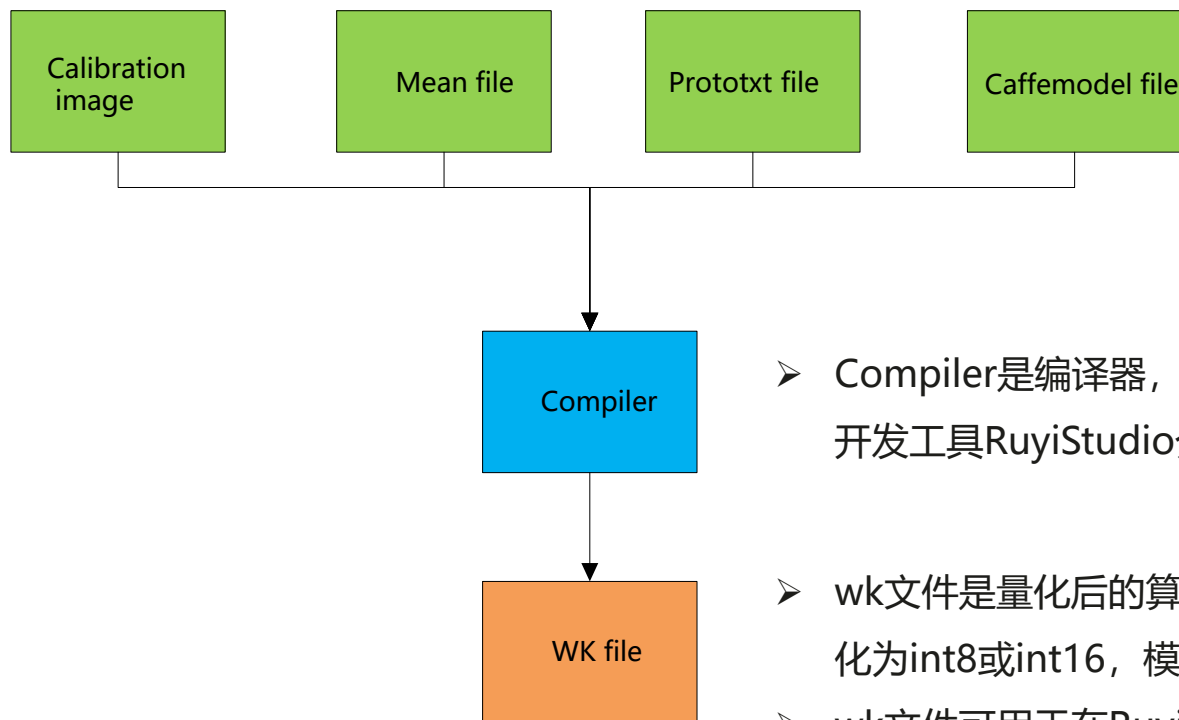
NNIE只用于智能的推理，训练相关部分请参考《训练和部署方法介绍》课程

量化为int8/int16在精度损失可控 (<1%) 的前提下，有以下优势：

1. 节省内存
2. 节省DDR带宽
3. 减小计算量，提升性能，降低功耗，满足嵌入式设备需求

# NNIE 模型的量化

量化过程中的配置及操作流程，请参考《NNIE开发工具RuyiStudio介绍》课程及《HiSVP开发指南》文档第3章节的内容。



➤ 算法模型量化时，需要依赖算法模型文件(prototxt file)，算法参数文件 (caffemodel file)，均值文件(mean file)，参考图像 (Calibration image)这四项；其中参考图像建议使用产品使用中的典型场景图像，建议30-50张左右。

➤ Compiler是编译器，放在配套的RuyiStudio工具里面，请参考《NNIE开发工具RuyiStudio介绍》课程。

➤ wk文件是量化后的算法模型文件，在算法训练完成后将算法模型的数据和参数从float32量化为int8或int16，模型可以减小为原来的1/4 (int8) 左右或1/2(int16)左右。

➤ wk文件可用于在RuyiStudio工具里面仿真或者是部署在芯片上进行实际的智能推理操作。

# NNIE的高低精度特性

8bit/16bit

NNIE支持全8bit或全16bit单一精度模式的算法模型量化。

使用场景：用户也可以折中选择两者的混合精度模式的算法模型量化，解决全低精度8bit模式下量化损失大的问题。

使用方法：需要用户在量化前先修改模型文件，然后再进行量化和部署及推理来实现，prototxt修改示例如下：

```
layer {  
  name: "conv5_hp"  
  type: "Convolution"  
  bottom: "conv4"  
  top: "conv5"  
  convolution_param {  
    num_output: 256  
    kernel_size: 3  
    pad: 1  
    stride: 1  
  }  
}
```

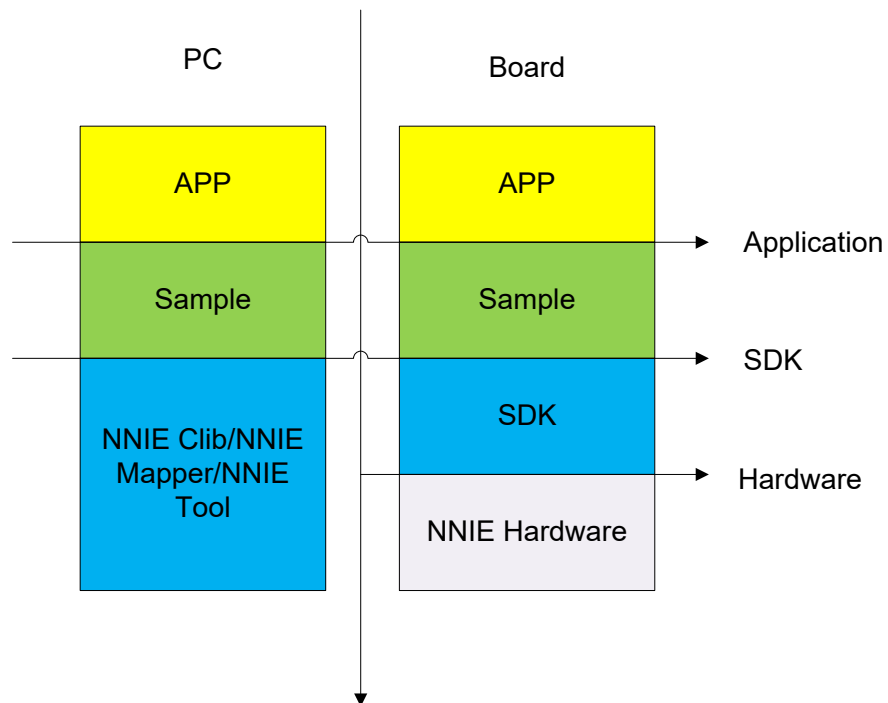
# NNIE中间层结果上报功能

使用场景：NNIE支持在推理的过程中将中间层结果上报到DDR内存里面，主要用于开发阶段的调试。

使用方法：需要用户在量化前先修改模型文件，然后再进行量化和部署及推理来实现，prototxt修改示例如下：

```
layer {
  name: "conv5 "
  type: "Convolution"
  bottom: "conv4"
  top: "conv5_report"
  convolution_param {
    num_output: 256
    kernel_size: 3
    pad: 1
    stride: 1
  }
}
```

# NNIE 仿真



仿真类型：功能仿真和指令仿真

- 功能仿真用于算法开发的早期或是没有硬件环境的情况下，先进行算法的开发及效果初步评估
- 指令仿真用于在PC端评估算法模型在芯片端的精度，性能，DDR带宽消耗

仿真需要使用NNIE的PC端工具RuyiStudio环境来实现

仿真过程中的配置及操作流程，请参考《NNIE开发工具RuyiStudio介绍》课程及《HiSVP开发指南》文档第5章节的内容。

# NNIE不支持层的处理

## ➤ 量化前：

修改算法模型prototxt文件，将不支持的层修改为custom层或proposal层

1.输出为矩形信息的不支持层需要修改为proposal层

2.输出为特征信息的不支持层需要修改为custom层

不支持层的修改方法请参考《HiSVP开发指南》文档3.1.5章节的详细说明

## ➤ 量化中：

生成wk算法模型文件

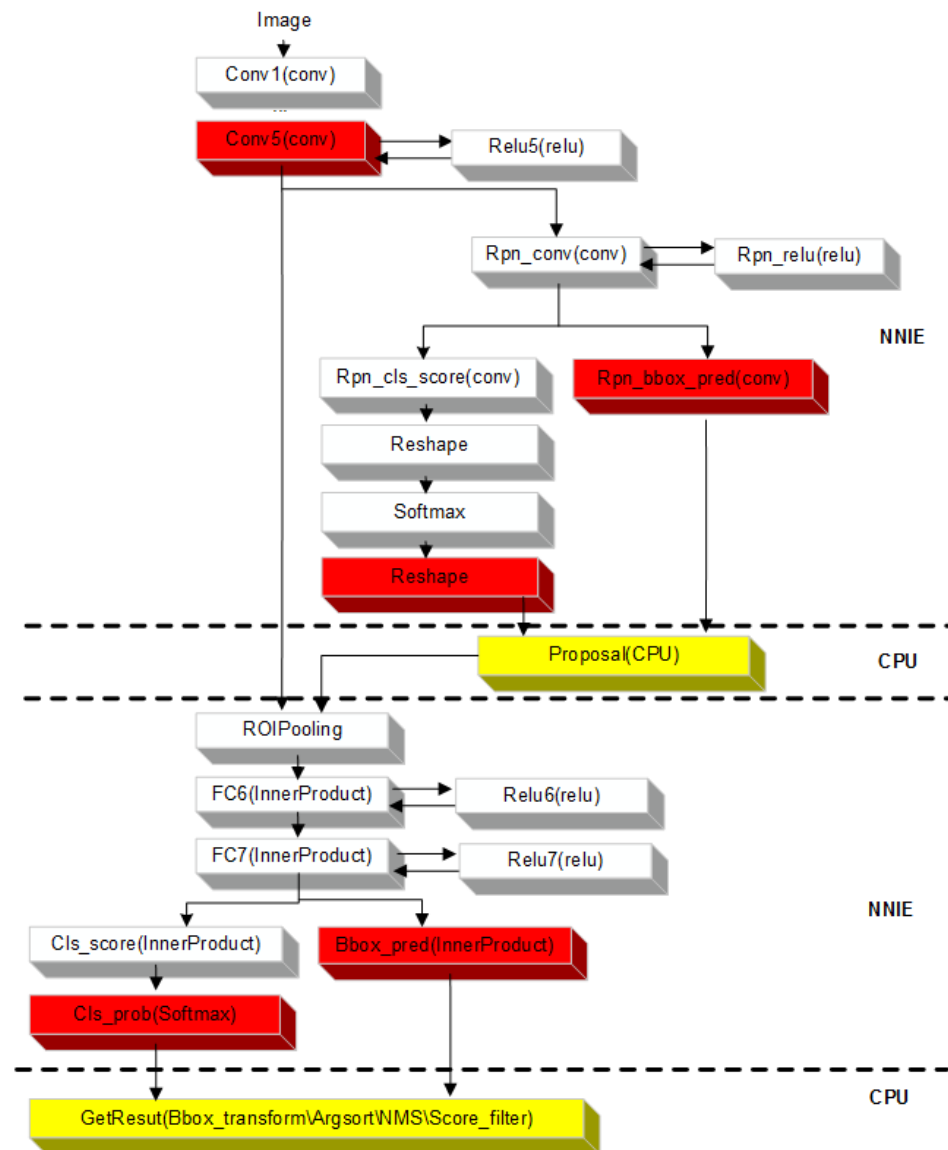
## ➤ 量化后，使用wk文件开始仿真及芯片端部署：

不支持的层将运行在CPU上，根据算法模型分段信息，需要用户做以下事项

1.确认算法模型分为几段

2.NNIE运行的每段最后一层结果的获取

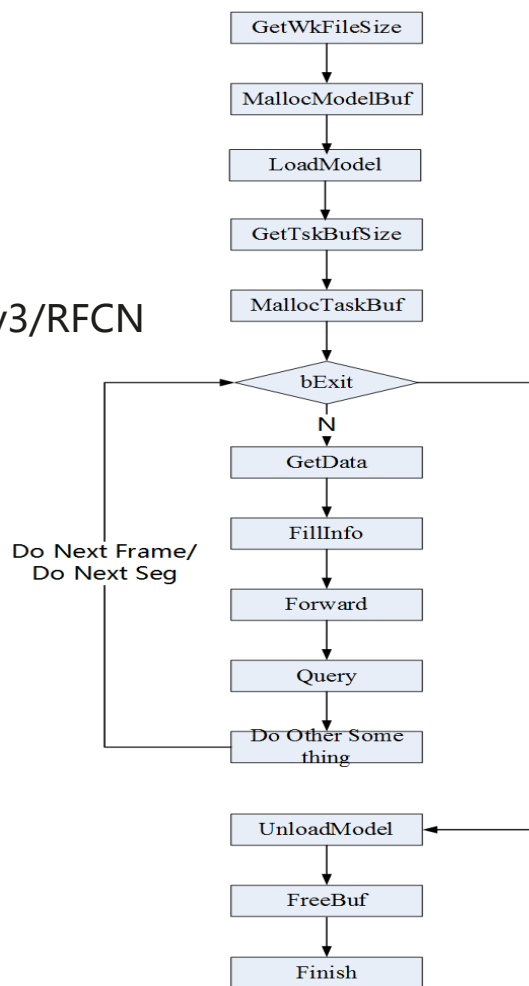
3.CPU运行部分代码的实现，比如参考芯片端FasterRcnn sample样例



# NNIE Sample用例

目前芯片端已经提供以下开源算法的Sample样例：

- 分类网： CNN
- 检测网： FasterRcnn/SSD/Yolov1/Yolov2/Yolov3/RFCN
- 分割网： SegNet
- 循环网络： LSTM



Sample的具体实现请参考发布包里面mpp/sample/svp/nnie目录下的源码

```
SAMPLE_SVP_PRINTF_INFO("Cnn init!\n");
s32Ret = SAMPLE_SVP_NNIE_Cnn_Init(&stCnnCfg,&s_stCnnNniePara,&s_stCnnSoftwarePara);
SAMPLE_SVP_CHECK_EXPR_GOTO(HI_SUCCESS != s32Ret, CNN_FAIL_0, SAMPLE_SVP_ERR_LEVEL_ERROR,
    "Error, SAMPLE_SVP_NNIE_Cnn_Init failed!\n");

/*NNIE fill src data*/
SAMPLE_SVP_PRINTF_INFO("Cnn start!\n");
stInputDataIdx.u32SegIdx = 0;
stInputDataIdx.u32NodeIdx = 0;
s32Ret = SAMPLE_SVP_NNIE_FillSrcData(&stCnnCfg,&s_stCnnNniePara,&stInputDataIdx);
SAMPLE_SVP_CHECK_EXPR_GOTO(HI_SUCCESS != s32Ret, CNN_FAIL_1, SAMPLE_SVP_ERR_LEVEL_ERROR,
    "Error, SAMPLE_SVP_NNIE_FillSrcData failed!\n");

/*NNIE process(process the 0-th segment)*/
stProcSegIdx.u32SegIdx = 0;
s32Ret = SAMPLE_SVP_NNIE_Forward(&s_stCnnNniePara,&stInputDataIdx,&stProcSegIdx,HI_TRUE);
SAMPLE_SVP_CHECK_EXPR_GOTO(HI_SUCCESS != s32Ret, CNN_FAIL_1, SAMPLE_SVP_ERR_LEVEL_ERROR,
    "Error, SAMPLE_SVP_NNIE_Forward failed!\n");

/*software process*/
/*if user has changed net struct, please make sure SAMPLE_SVP_NNIE_Cnn_GetTopN
function's input datas are correct*/
s32Ret = SAMPLE_SVP_NNIE_Cnn_GetTopN(&s_stCnnNniePara,&s_stCnnSoftwarePara);
SAMPLE_SVP_CHECK_EXPR_GOTO(HI_SUCCESS != s32Ret, CNN_FAIL_1, SAMPLE_SVP_ERR_LEVEL_ERROR,
    "Error, SAMPLE_SVP_NNIE_Cnn_GetTopN failed!\n");

/*print result*/
SAMPLE_SVP_PRINTF_INFO("Cnn result:\n");
s32Ret = SAMPLE_SVP_NNIE_Cnn_PrintResult(&(s_stCnnSoftwarePara.stGetTopN),
    s_stCnnSoftwarePara.u32TopN);
SAMPLE_SVP_CHECK_EXPR_GOTO(HI_SUCCESS != s32Ret, CNN_FAIL_1, SAMPLE_SVP_ERR_LEVEL_ERROR,
    "Error, SAMPLE_SVP_NNIE_Cnn_PrintResult failed!\n");

NN_FAIL_1:
SAMPLE_SVP_NNIE_Cnn_Deinit(&s_stCnnNniePara,&s_stCnnSoftwarePara);
```

用户需要将RuyiStudio工具量化后生成的wk模型文件放入芯片端的Flash或DDR里面，然后就可以参考芯片端的sample样例进行部署和推理

# NNIE MPI接口

```
/*加载模型*/
HI_S32 HI_MPI_SVP_NNIE_LoadModel(const SVP_SRC_MEM_INFO_S *pstModelBuf, SVP_NNIE_MODEL_S *pstModel);

/*卸载模型*/
HI_S32 HI_MPI_SVP_NNIE_UnloadModel(SVP_NNIE_MODEL_S *pstModel);

/*解析模型*/
HI_S32 HI_MPI_SVP_NNIE_GetTskBufSize(HI_U32 u32MaxInputNum, HI_U32 u32MaxBboxNum, const SVP_NNIE_MODEL_S *pstModel, HI_U32
au32TskBufSize[], HI_U32 u32NetSegNum);

/*模型推理*/
HI_S32 HI_MPI_SVP_NNIE_Forward(SVP_NNIE_HANDLE *phSvpNnieHandle, const SVP_SRC_BLOB_S astSrc[], const SVP_NNIE_MODEL_S *pstModel,
const SVP_DST_BLOB_S astDst[], const SVP_NNIE_FORWARD_CTRL_S *pstForwardCtrl, HI_BOOL bInstant);

HI_S32 HI_MPI_SVP_NNIE_ForwardWithBbox(SVP_NNIE_HANDLE *phSvpNnieHandle, const SVP_SRC_BLOB_S astSrc[], const SVP_SRC_BLOB_S
astBbox[], const SVP_NNIE_MODEL_S *pstModel, const SVP_DST_BLOB_S astDst[], const SVP_NNIE_FORWARD_WITHBBOX_CTRL_S *pstForwardCtrl,
HI_BOOL bInstant);

/*推理任务状态查询*/
HI_S32 HI_MPI_SVP_NNIE_Query(SVP_NNIE_ID_E enNnieId, SVP_NNIE_HANDLE svpNnieHandle, HI_BOOL *pbFinish, HI_BOOL bBlock);

/*模型任务节点的分配与回收*/
HI_S32 HI_MPI_SVP_NNIE_AddTskBuf(const SVP_MEM_INFO_S* pstTskBuf);
HI_S32 HI_MPI_SVP_NNIE_RemoveTskBuf(const SVP_MEM_INFO_S* pstTskBuf);
```

接口的具体使用请参考《HiSVP API 参考》文档第2章节

# NNIE 重要概念

## ➤ 网络分段

对于NNIE 不支持的某些网络层节点，编译器支持用户对网络分段，不执行的部分编译器不去编译，由用户自己用CPU 去实现。  
强烈建议用户尽量使用NNIE 支持的层去实现网络模型；NNIE 不支持的段数越多，网络切分越碎，软硬件交互越频繁，效率越低。

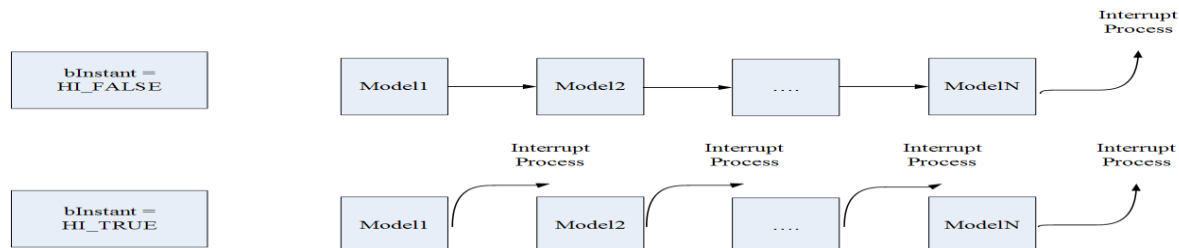
## ➤ 句柄handle

用户在调用NNIE API创建任务时，系统会为每个任务分配一个handle，用于标识不同的任务。

## ➤ 及时返回结果标志bInstant

用户在创建某个任务后，希望及时得到该任务完成的信息，则需要在创建该任务时，将bInstant设置为HI\_TRUE。

否则，如果用户不关心该任务是否完成，建议将bInstant设置为HI\_FALSE，这样可以与后续任务组链执行，减少中断次数，提升性能。



# NNIE 重要概念

## ➤ 查询query

用户根据系统返回的handle，调用HI\_MPI\_SVP\_NNIE\_Query可以查询对应算子任务是否完成。

## ➤ 及时刷 Cache

NNIE 硬件只能从DDR中获取数据。如果用户在调用NNIE 任务时，访问空间可cache而且CPU曾经访问，为了保证NNIE 输入输出数据不被CPU cache干扰，此时用户需要调用HI\_MPI\_SYS\_MmzFlushCache接口刷cache（详细信息请参见《HiMPP V4.0 媒体处理软件开发参考》），将数据从cache 刷到DDR，以供NNIE使用。

# NNIE Proc信息

```
[NNIE] Version: [Hi3516CV500_MPP_V2.0.2.0 B030 Release], Build Time[Sep 10 2019, 19:48:47]
```

```
-----NNIE MODULE PARAM-----
nnie_save_power  nnie_max_tskbuf_num
|               |
|               | 1                32
|               |
|               |

-----NNIE QUEUE INFO-----
CoreId      Wait      Busy WaitCurId WaitEndId BusyCurId BusyEndId
|           |         |         |         |         |         |
|           | 0       1         0         0         1         0         1
|           |
|           |

-----NNIE TASK INFO-----
CoreId      Hnd      TaskFsh      LastId      TaskId      HndWrap      FshWrap      FreeTskBufNum      UseTskBufNum
|           |         |         |         |         |         |         |         |
|           | 0       310896      310894         0         0         0         0         32         0
|           |
|           |

-----NNIE RUN-TIME INFO-----
CoreId      LastInst      CntPerSec      MaxCntPerSec      TotalIntCntLastSec      TotalIntCnt      QTCnt      STCnt      CfgErrCnt
|           |         |         |         |         |         |         |         |
|           | 0         1         90         121         310841         310894         22         0         0
|           |
|           |
CostTm      MCostTm      CostTmPerSec      MCostTmPerSec      TotalIntCostTm      RunTm
|           |         |         |         |         |         |
|           | 16        199        2105        4794        6774595        6066
|           |
|           |

-----NNIE INVOKE INFO-----
CoreId      Forward      ForwardWithBbox
|           |         |         |
|           | 0       310896         0
|           |
|           |
```

cat /proc/umap/nnie

proc信息各字段的具体含义请参考《HiSVP API 参考》文档第2章节的2.6小节

# NNIE 主要规格

- $N * N$  (1~255) Convolution / Deconvolution
- Pooling (Max, Average)
- Stride (1~255)
- Pad (0~255)
- Activation Function: Relu, Sigmoid, TanH and so on
- LRN
- BN(Batch Normalization)
- Inner Product
- Concat
- Eltwise
- 8bit data and parameter mode

功能的具体介绍请参考《HiSVP 开发指南》文档第3章节

# NNIE 主要规格

- 数据和参数的位宽可配置为int8或int16模式
- 支持灰度图, RGB图, FeatureMap等输入
- NNIE内部支持YUV SP420/SP422转RGB功能
- 支持输入图像预处理
- 支持输入图像批处理功能
- 支持网络中间层上报功能

功能的具体介绍请参考《HiSVP 开发指南》文档第3章节

# NNIE 支持的神经网络类型

- 分类网: Alexnet, VGG16, Googlenet, Resnet50, Squeezenet, Mobilenetv2
- 检测网: FasterRcnn, SSD, Yolov1, Yolov2, Yolov3, RFCN
- 分割网: SegNet, FCN
- 循环网络: LSTM

其余算法模型请参考《HiSVP开发指南》第3章文档中支持的层来判断是否支持

# NNIE硬件利用率优化建议

- 一般情况下，相同计算量场景，尽量增加单层的计算量，减少网络层数，以减少层之间切换造成的利用率损失，提高端到端的利用率。
- 尽量使用卷积、反卷积、Pooling、FC层，提高硬件资源利用率，减少LRN、MVN、Normalize、Softmax层的使用。
- 建议使用RELU、Sigmod、Tanh、PRELU、RRELU激活函数并使用in-place的配置方式（与前一层共享blob），以便硬件高效完成计算。

# 目录

---

1. 总体介绍

2. NNIE

3. IVE

# IVE算子介绍

DMA  
Filter  
CSC  
FilterAndCSC  
Sobel  
MagAndAng  
Dilate  
Erode  
Thresh  
And  
Sub  
Or  
HOG  
KCF  
PSP

Integ  
Hist  
Thresh\_S16  
Thresh\_U16  
16BitTo8Bit  
OrdStatFilter  
Map  
Add  
Xor  
NCC  
CCL  
GMM  
CANNY  
SAD

各个算子的实现原理可以参考opencv的实现原理，具体使用方法及说明请参考《HiIVE API 参考》文档

# IVE常用算子使用场景

- DMA算子：用于数据的搬移，支持快速拷贝、间隔拷贝、内存填充：可实现数据从一块内存快速拷贝到另一块内存，或者从一块内存有规律的拷贝一些数据到另一块内存，或者对一块内存进行填充操作。
- CSC算子：用于色彩空间的转换，可实现YUV2RGB\YUV2HSV\YUV2LAB\RGB2YUV的色彩空间转换。
- HOG算子：计算给定区域的HOG(Histogram of Oriented Gradient)特征。常与KCF算子结合，用于物体跟踪场景。
- KCF算子族：目标跟踪算法，用于目标物体的跟踪。
- PSP算子：透视变换，用于人脸识别前的人脸对齐预处理，提升人脸识别准确率。
- Resize算子：用于图像的放大或缩小，可以进行NNIE输入图像的预处理。

# IVE 重要概念

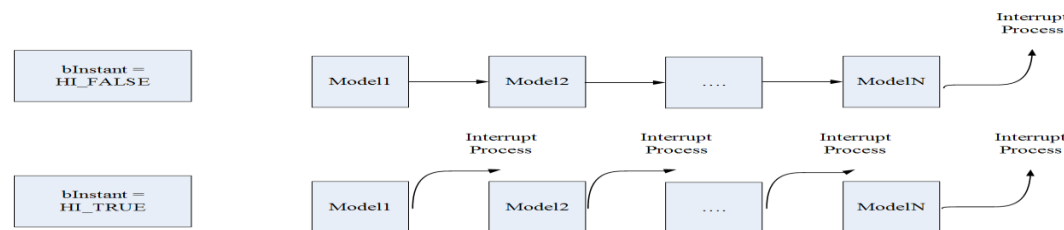
## ➤ 句柄handle

用户在调用算子创建任务时，系统会为每个任务分配一个handle，用于标识不同的任务。

## ➤ 及时返回结果标志bInstant

用户在创建某个任务后，希望及时得到该任务完成的信息，则需要在创建该任务时，将 bInstant 设置为 HI\_TRUE 。

否则，如果用户不关心该任务是否完成，建议将 bInstant 设置为 HI\_FALSE ，这样可以与后续任务组链执行，减少中断次数，提升性能 。



## ➤ query

用户根据系统返回的handle，调用HI\_MPI\_IVE\_Query可以查询对应算子任务是否完成。

## ➤ 及时刷 Cache

IVE硬件只能从DDR中获取数据。如果用户在调用IVE 任务时，访问空间可cache而且CPU曾经访问，为了保证IVE输入输出数据不被CPU cache干扰，此时用户需要调用HI\_MPI\_SYS\_MmzFlushCache接口刷cache（详细信息请参见《HiMPP V4.0 媒体处理软件开发参考》），将数据从cache 刷到DDR，以供IVE使用。

# IVE Proc信息

```
[IVE] Version: [Hi3516CV500_MPP_V2.0.2.F B010 Release], Build Time[May 25 2020, 10:56:30]
```

```
-----MODULE PARAM-----
    save_power      max_node_num
      0             512

-----IVE QUEUE INFO-----
    Wait      Busy      WaitCurId WaitEndId BusyCurId BusyEndId
      0        -1         0         0         0         0

-----IVE TASK INFO-----
    Hnd      TaskFsh      LastId      TaskId      HndWrap      FshWrap
    312      312         0         0         0         0

-----IVE RUN-TIME INFO-----
    LastInst  CntPerSec  MaxCntPerSec  TotalIntCntLastSec  TotalIntCnt  QTCnt  STCnt
      1         1         2         312         312         0         0

    CostTm    MCostTm    CostTmPerSec  MCostTmPerSec  TotalIntCostTm  RunTm
      4        28         4         36         2986      192620042

-----IVE INVOKE INFO-----
    DMA      Filter      CSC      FltCsc      Sobel      MagAng      Dilate      Erode
      7         0         0         0         0         0         0         0

    Thresh      And      Sub      Or      Integ      Hist  ThreshS16  ThreshU16
      0         0         0         0         0         0         0         0

    16to8  OrdStatFlt  BernSen      Map      EqualH      Add      Xor      NCC
      0         0         0         0         0         0         0         0

    CCL      GMM      Canny      LBP  NormGrad      LK  ShiTomasi  GradFg
      0         0         0         0         0         0         0         0

    MatchMod  UpdateMod  Radon      ANN      SVM      AdpThr  LineFltH  NoiseRmH
      0         0         0         0         0         0         0         0

    PlateChar      SAD      GMM2      Resize      CNN  PerspTrans      KCF      HOG
      0         0         0         0         0         0         305         0

    XNN
      0
```

cat /proc/umap/ive

proc信息各字段的具体含义请参考《HiIVE API 参考》文档第5章节



# 筑就智能时代基石

Copyright©2021 Shanghai HiSilicon Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Shanghai HiSilicon may change the information at any time without notice.