

Ascend Full-Stack AI

Software/Hardware Platform

AI Engineer

Kubilay Tuna

FAE

Serkan Celik

FAE

Alper Balmumcu



Contents



- **1. Huawei Da Vinci Architecture & Atlas Products**
- **2. CANN: Compute Architecture for Neural Networks**
- **3. ACL: Ascend Compute Language**
- **4. MindStudio: FullPline Development Toolchain**
- **5. MindX: Ascend Computing Application Enablement Kit**
- **6. Mindspore: Opensource AI Framework**

Towards an AI Era



AI: The New Trend of the Industry

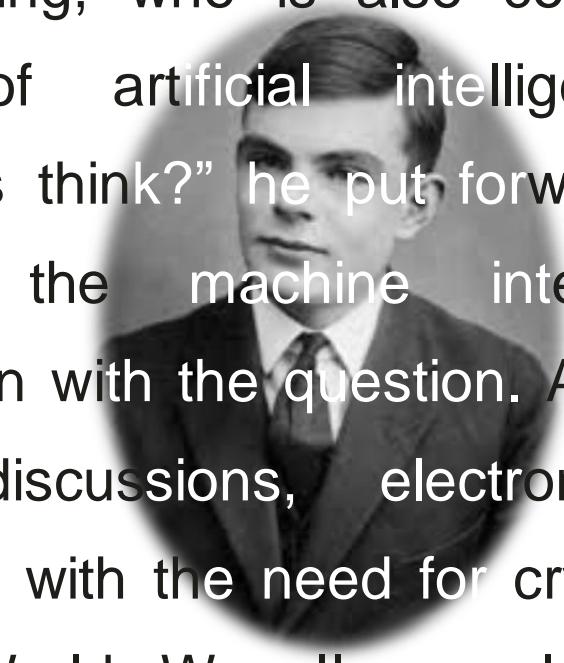
What is AI?

The AI is the invention that enables a computer or a robot connected to a computer system to perform certain actions similar to living things. AI aims to provide machines with skills that are human features such as knowledge, reasoning, perception, learning and managing textures .

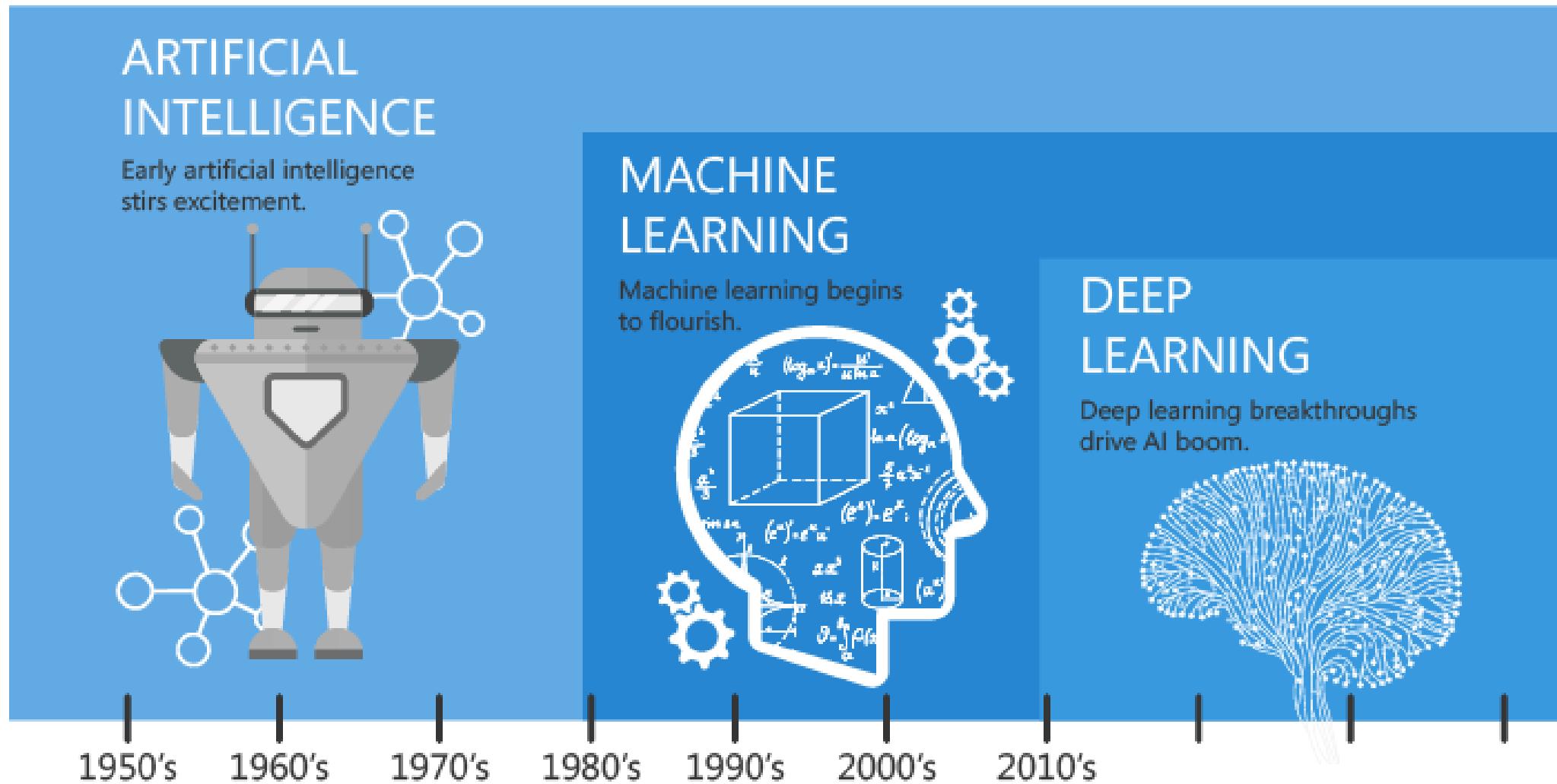


Can AI think?

Alan Turing, who is also considered the father of artificial intelligence; “Can machines think?” he put forward in 1943. opened the machine intelligence to discussion with the question. As a result of these discussions, electronic devices produced with the need for crypto analysis during World War II revealed computer science and artificial intelligence technology.

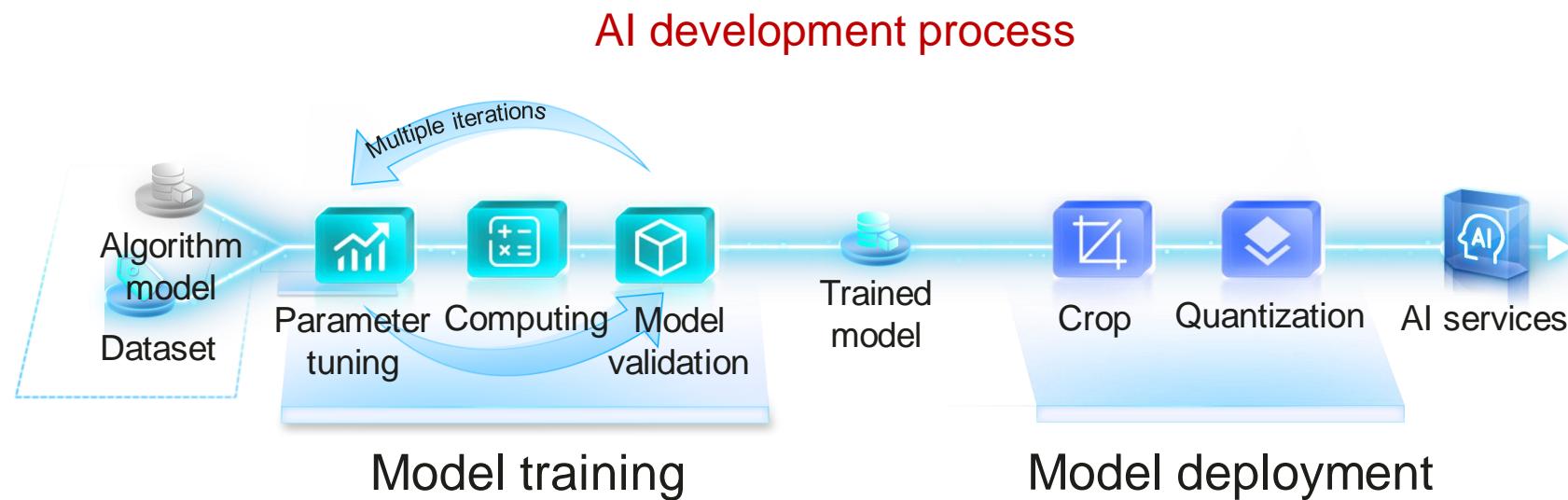
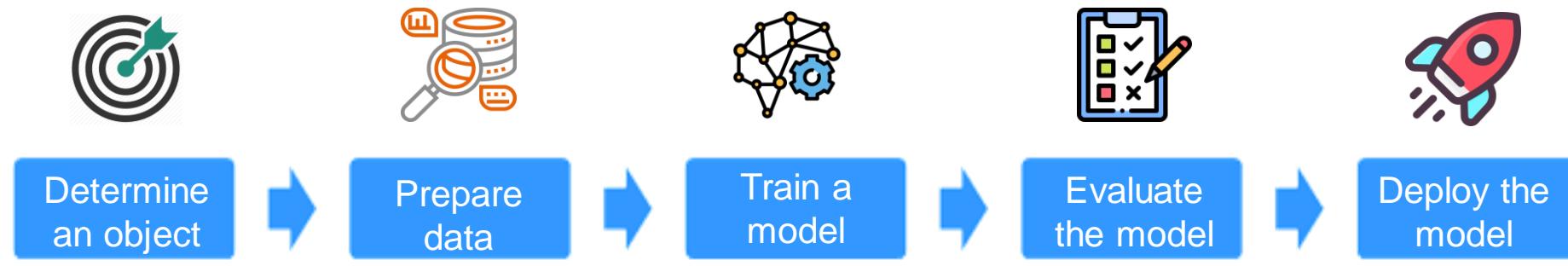


Relationship between AI, ML and DL



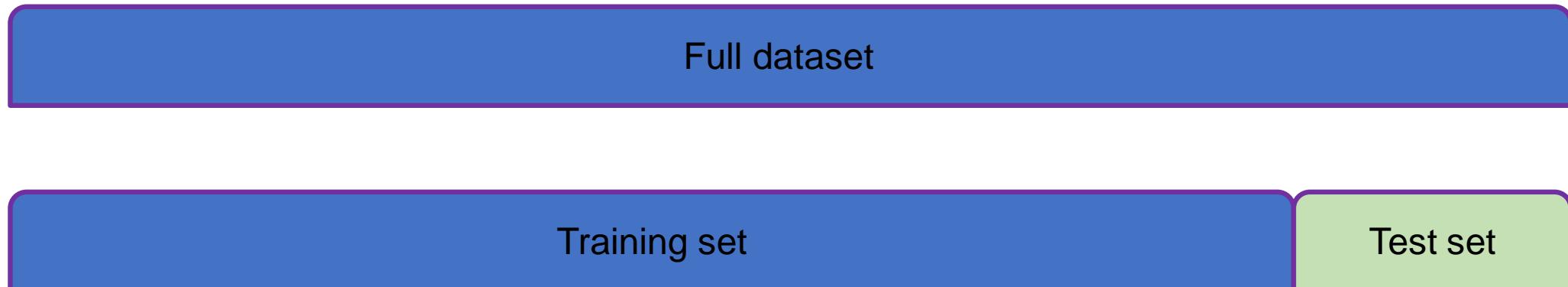
Basic Process of AI Development

The **AI development process** is as follows: determining an objective, preparing data, and training, evaluating, and deploying a model.



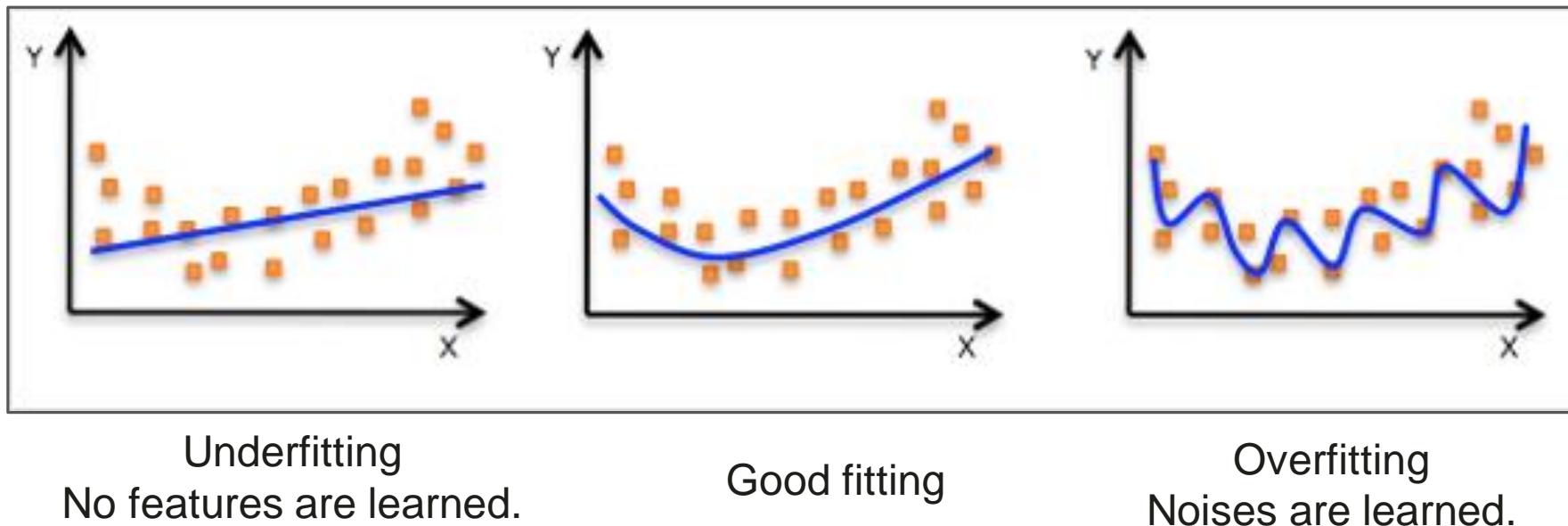
Training Set and Test Set

- **Training set:** dataset used in the training process, where each piece of training data is referred to as a training sample. Learning (or training) is the process of creating a model from data.
- **Test set:** dataset used for the testing process, where each sample is called a test sample. Testing refers to the process, during which the created model is used for prediction.



Underfitting and Overfitting

- **Underfitting:** The model is too simple. As a result, there are large training error and generalization error.
- **Overfitting:** The training error of a model obtained through training is very small, but the generalization capability is weak; that is, the generalization error is large.



Huawei Ascend AI Solutions

— AI Unlocks Boundless Possibilities —



Huawei Da Vinci Architecture Unlocks the Ultimate AI Computing Power

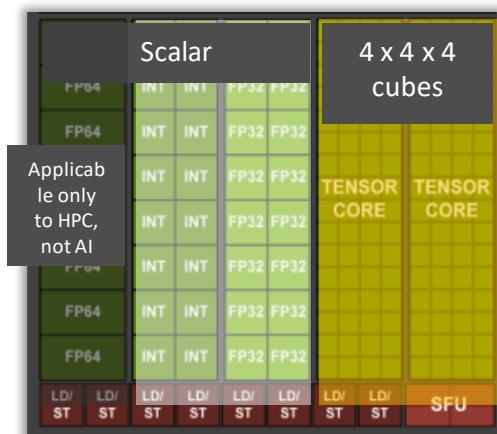
AI computing characteristics

~90% are matrix multiply operation

- AI computing is mainly based on the convolutional neural network (CNN) model.
- About 90% of the CNN model is based on matrix multiply operations.
- The **cube computing unit** is the most suitable.

10% Vector operation
Pooling
Relu

88.7% Matrix multiply operation
Convolution
Full-mesh



2x area-to-efficiency ratio

On the same area, the Da Vinci architecture delivers 2x computing power.

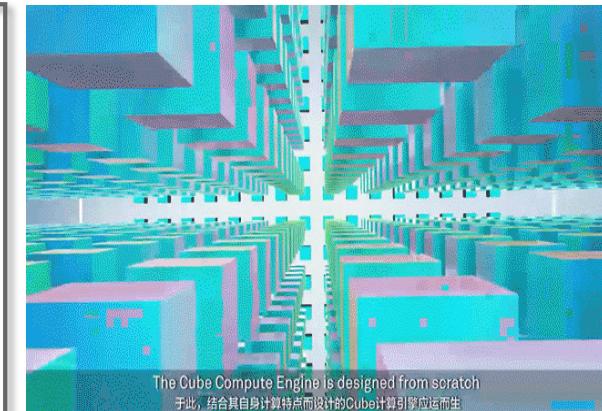
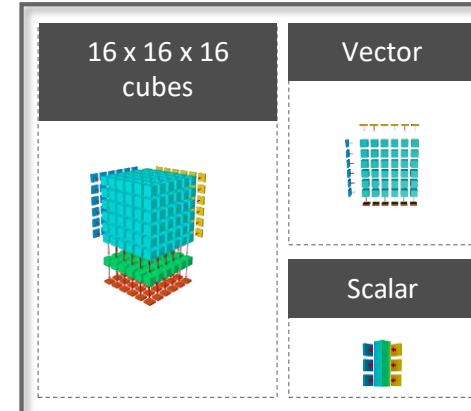
Da Vinci architecture: best-fit for AI computing

GPU

- The GPU is not designed for AI computing. Therefore, it is inefficient in matrix multiply computing.

NPU

- The Da Vinci architecture is specially designed for AI computing.
- Provides cube, vector, and scalar computing units for AI computing
- The Da Vinci architecture has a large proportion of cubes, enabling **high matrix multiply computing efficiency, and optimal area-to-efficiency ratio**.



	Other AI Chip Architectures	Da Vinci Architecture
Chip area (12nm)	5.x mm ²	13.x mm ²
Computing power	1.7 TOPS FP16	8 TOPS FP16
Area-to-efficiency ratio	~0.3	~0.6

Ascend AI Processors Empower Superior Compute Intelligence



AI SoC with ultimate perf./watt

- Ascend-Mini
Architecture: Da Vinci
- Half precision (FP16): 11 TFLOPS**
- Integer precision (INT8): 22 TOPS**
- 16-channel full-HD video decoder: H.264/265**
- Single-channel full-HD video encoder: H.264/265**
- Max. power consumption: 8 W**
- 12 nm



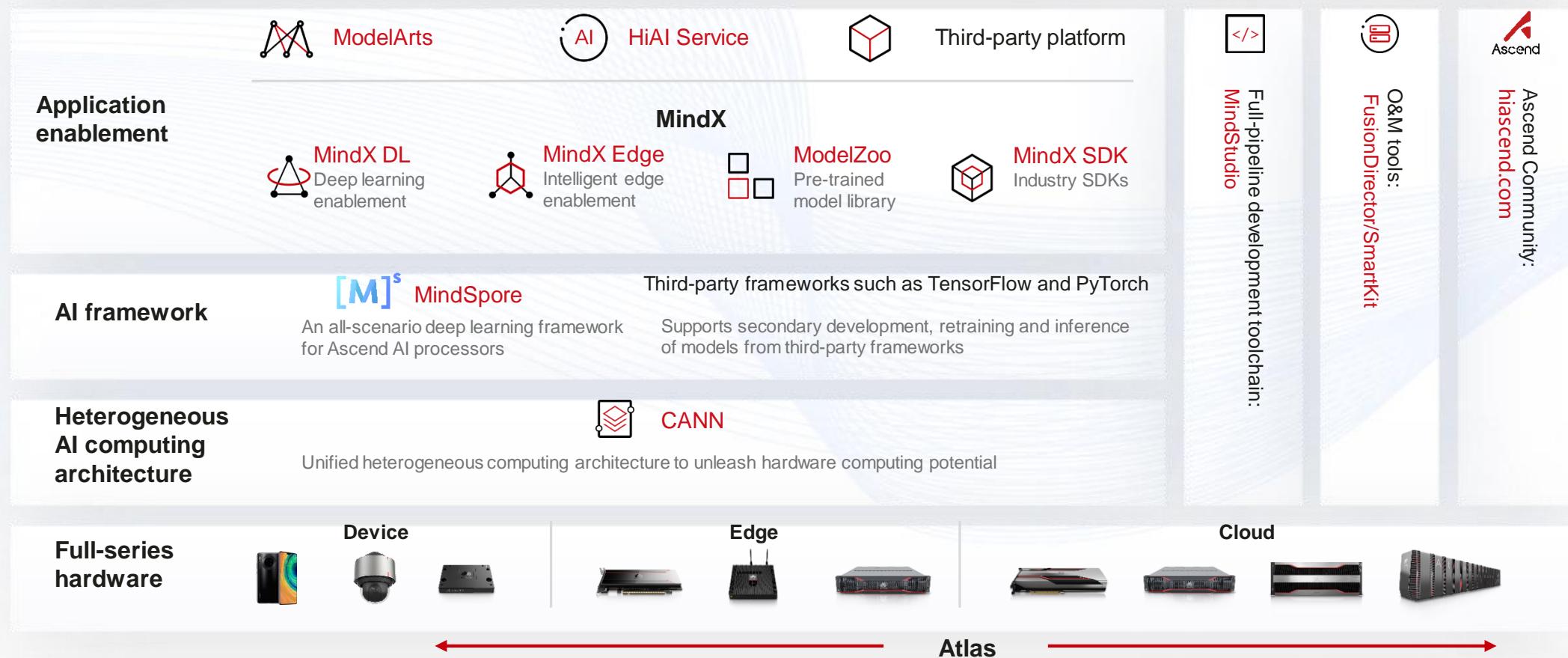
Most powerful AI processor

- Ascend-Max
Architecture: Da Vinci
- Half precision (FP16): 320 TFLOPS**
- Integer precision (INT8): 640 TOPS**
- 128-channel full-HD video decoder: H.264/265**
- Max. power consumption: 310 W**
- 7 nm

Overview of The Huawei Ascend Full-Stack AI Platform

Industry applications

Smart city, manufacturing, energy, transportation, finance, carriers, education, and more



Ascend AI Processors Empower Superior Compute Intelligence

Based on the Ascend AI Processors and mainstream heterogeneous computing components in the industry, the Atlas series hardware products provide various product forms, such as modules, boards (inference and training cards based on application scenarios), edge stations, servers, and clusters, to build a cloud-edge-device AI infrastructure solution for all scenarios.

 **Ultimate computing power**

 **All-scenario deployment**

 **Cloud-edge-device synergy**



Atlas 900 AI cluster



Atlas 800 AI server
Model: 9000/9010



Atlas 800 AI server
Model: 3000/3010



Atlas 300 AI accelerator card
Model: 3000/9000



Atlas 500 AI
edge station



Atlas 200 AI
accelerator module

● ----- **Cloud** ----- ● ----- **Edge** ----- ● ----- **Device** ----- ●



Ascend 310
AI Processor



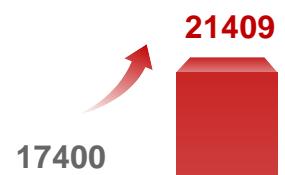
Ascend 910
AI Processor

Software-Hardware Collaboration for Superior Computing in Model Training

Best-in-class training performance of mainstream models

ResNet-50 Network
Performance (images/s)

23% higher



BERT-Large Network
Performance (sentences/s)

23.6% higher



Test benchmark: MindSpore tested on July 15, 2021



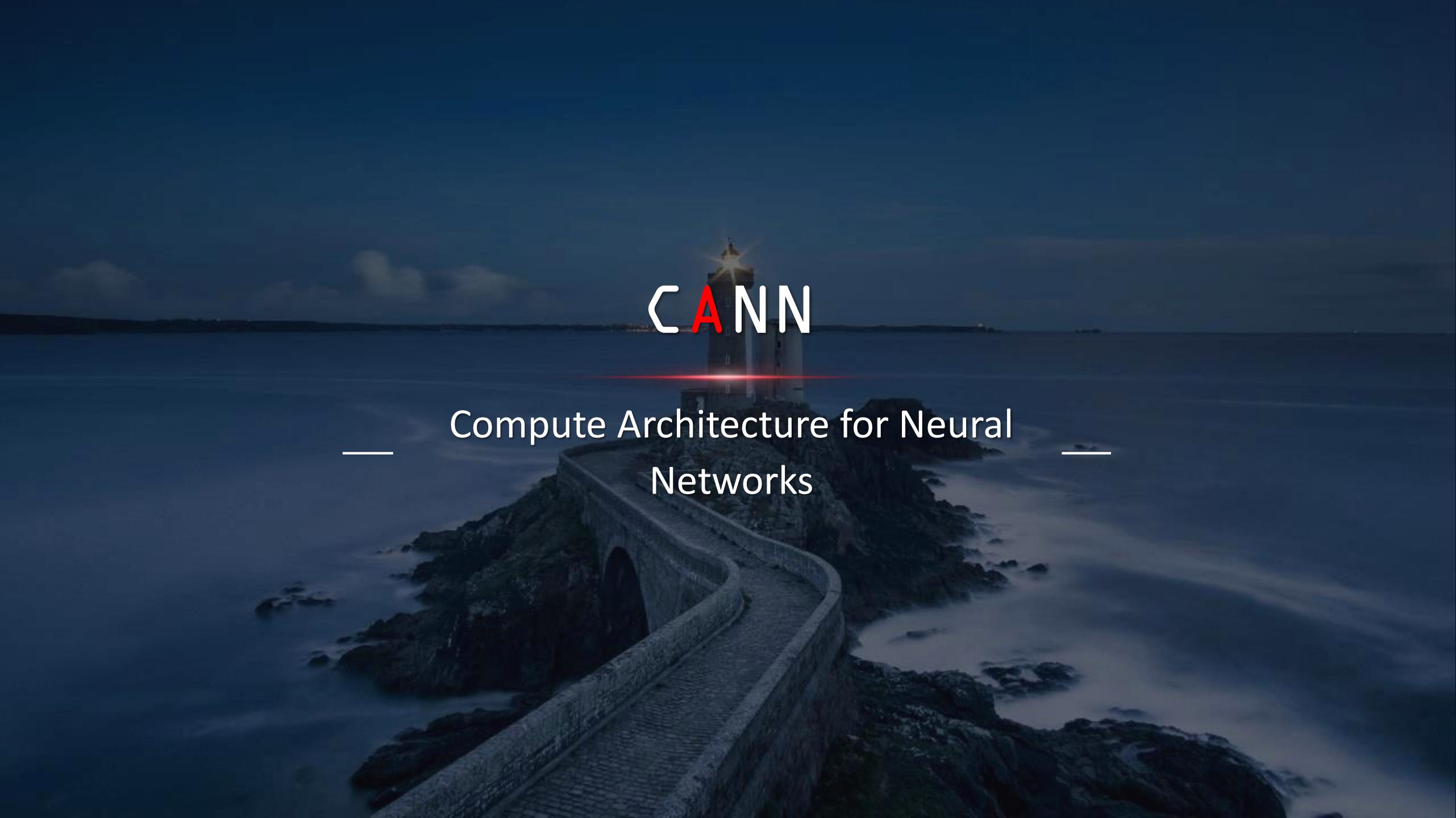
Better training performance of major models than that of NVIDIA

ResNet-50 (40% of ByteDance's computing power consumption)

Test Scenario	NVIDIA V100	Ascend 910	Performance Comparison Ratio
Parallel Processors	FP16+XLA img/s	FP16 img/s	
1	1276	1846	145%
8	8675	14060	162%
16	13191	27731	210%
32	24302	54150	223%

BERT-Large (25% of ByteDance's computing power consumption)

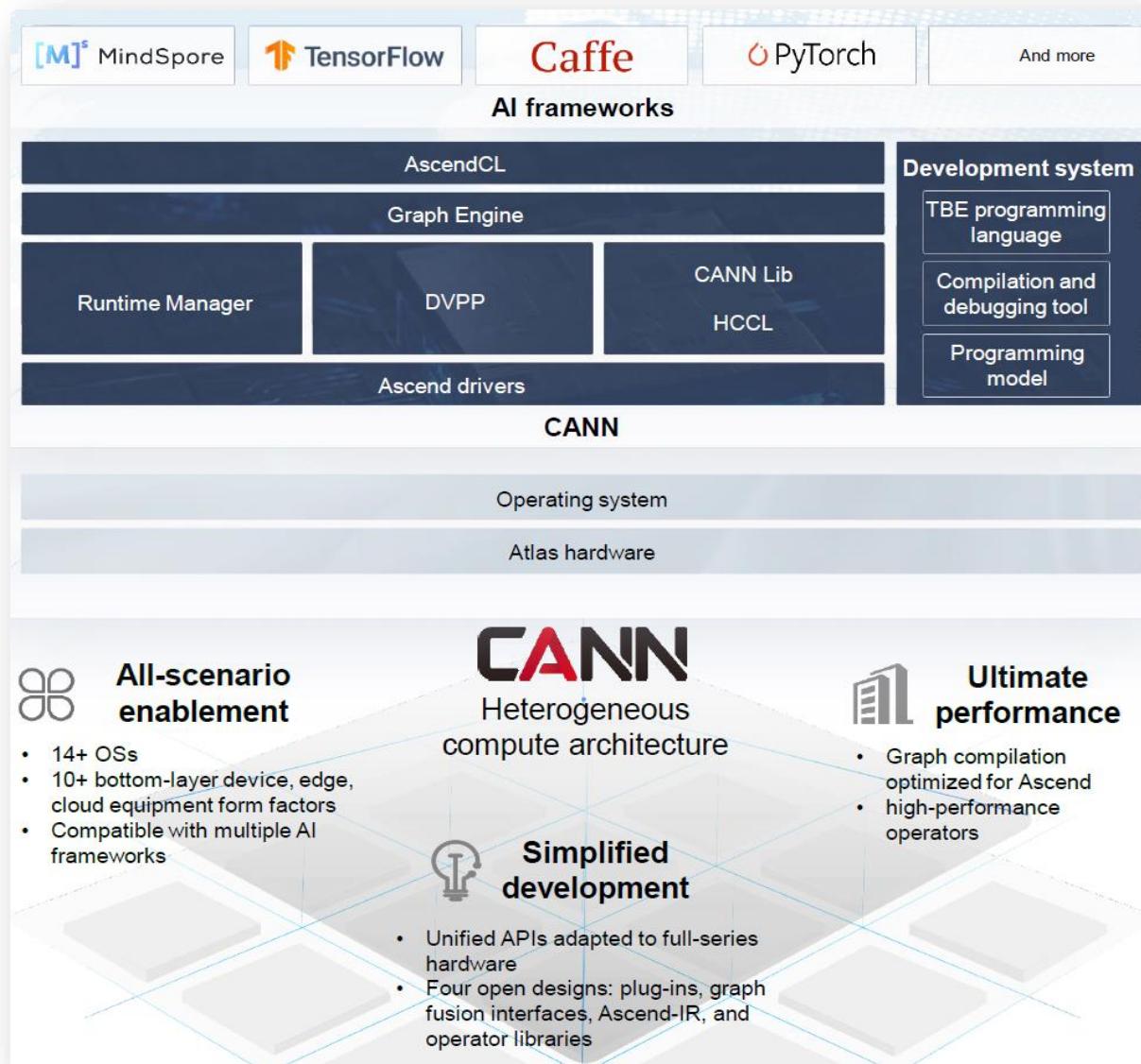
Test Scenario	NVIDIA V100	Ascend 910	Performance Comparison Ratio
Parallel Processors	FP16+XLA sentences/s	FP16 sentences/s	
1	62	119	192%
8	445	861	193%
16	800	1643	205%
32	1360	3050	224%

The background of the slide is a photograph of a coastal road at night. The road curves along a cliff edge, leading towards a lighthouse perched on a rocky outcrop. The sky is dark with some clouds, and the ocean is visible in the distance. A bright light from the lighthouse creates a lens flare effect.

CANN

— Compute Architecture for Neural
Networks —

What is the CANN & ACL?



The Compute Architecture for Neural Networks (CANN) is a AI heterogeneous compute architecture designed for deep learning. It provides hierarchical APIs such as AscendCL, DVPP, and HCCL to help you quickly build AI applications and services based on the Ascend platform.

A Unified Heterogeneous Compute Architecture Unleashes Ultimate Hardware Compute Power



Device-Edge-Cloud Synergy

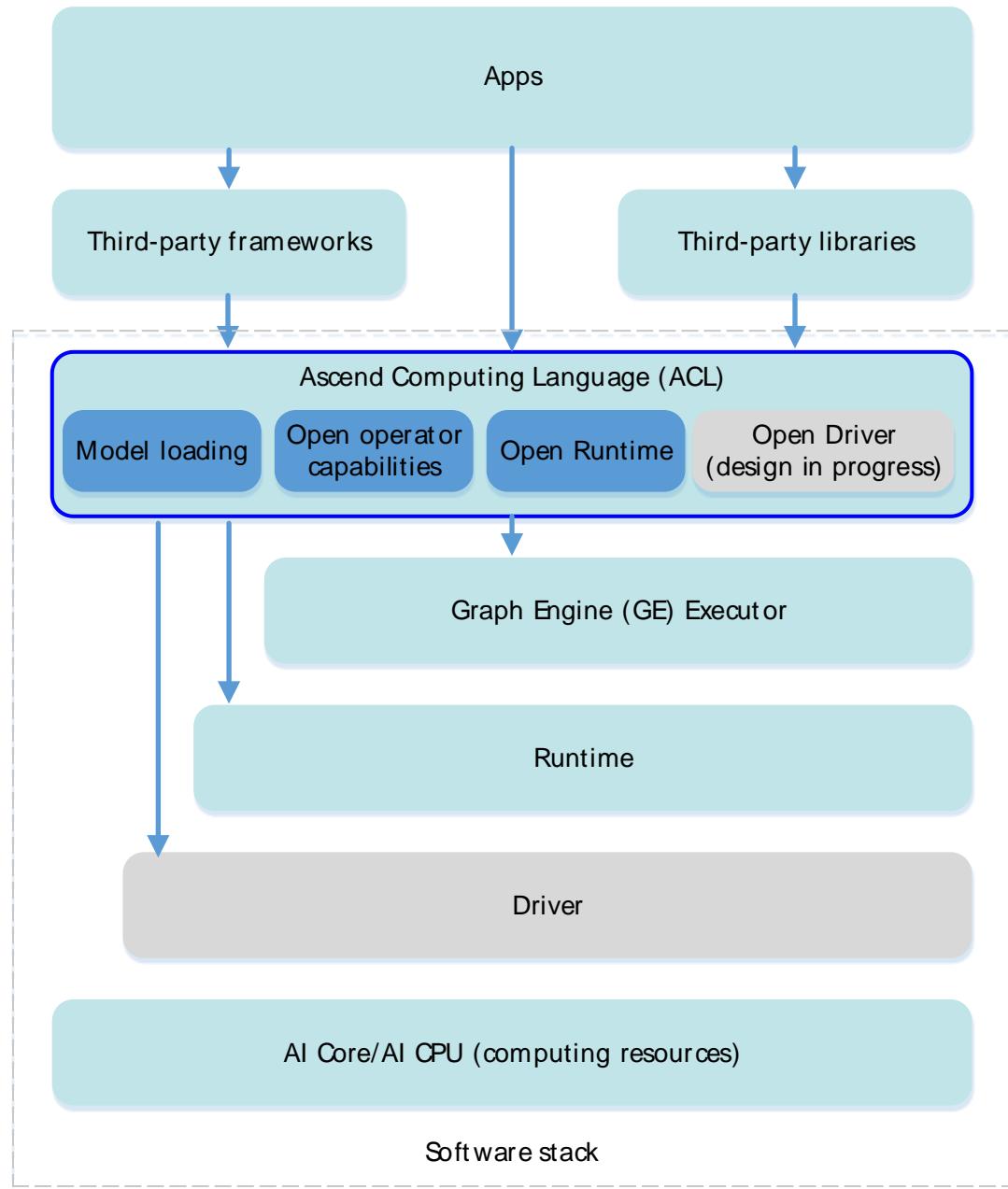


Efficient Development with AscendCL

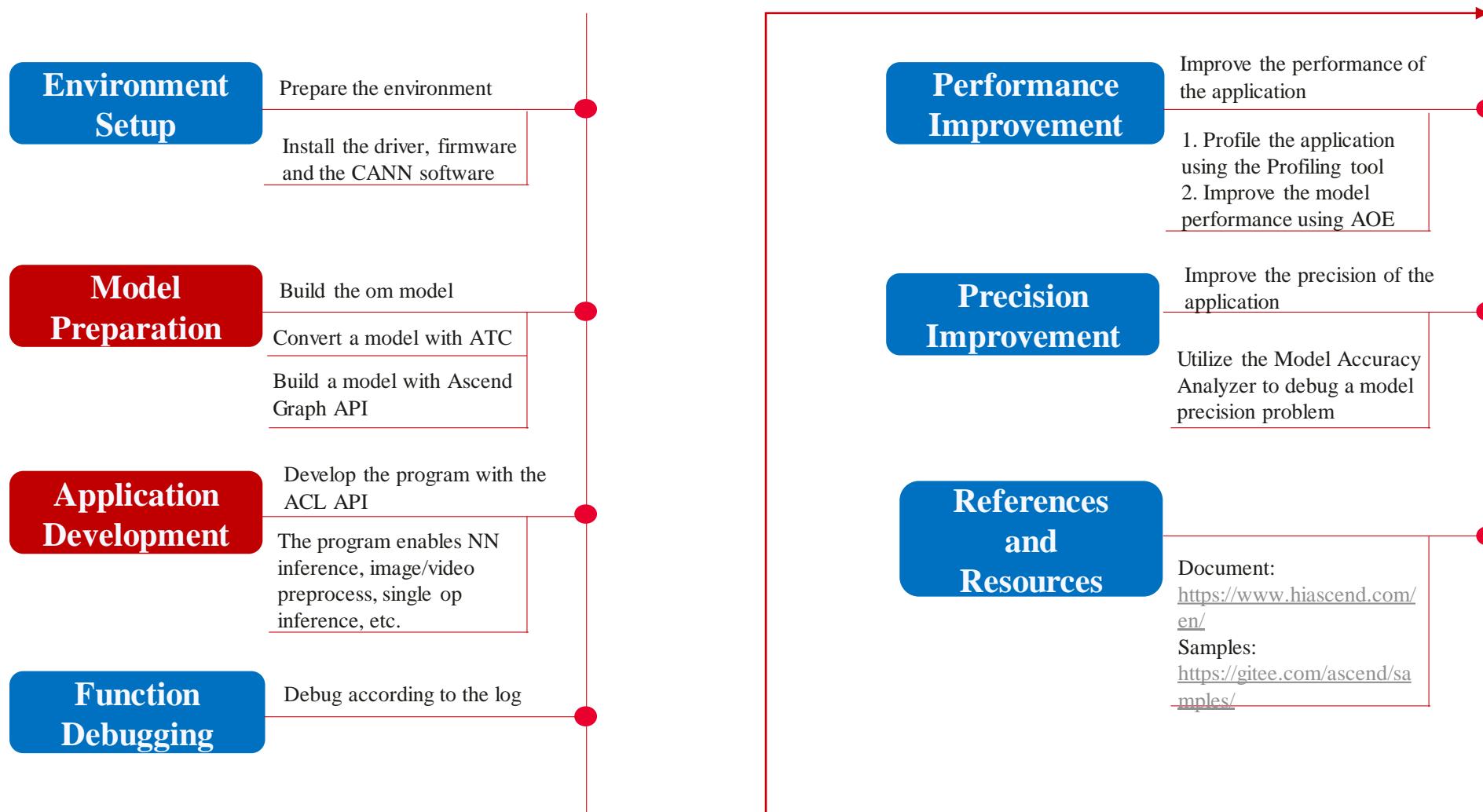


Unlimited Ascend Compute Power

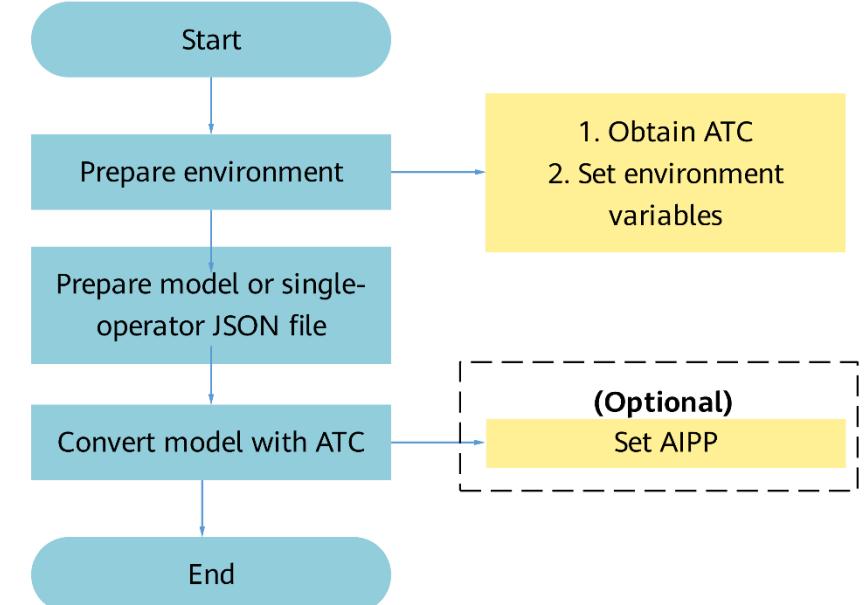
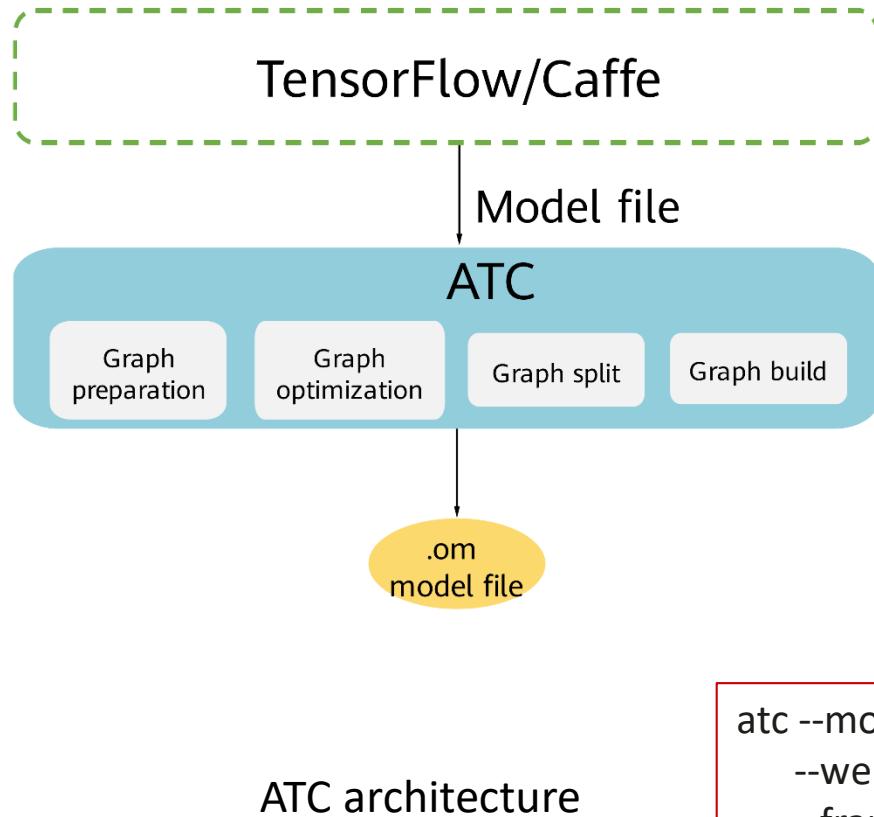
ACL Overview



Step of Application Development on CANN



Model Preparation with ATC

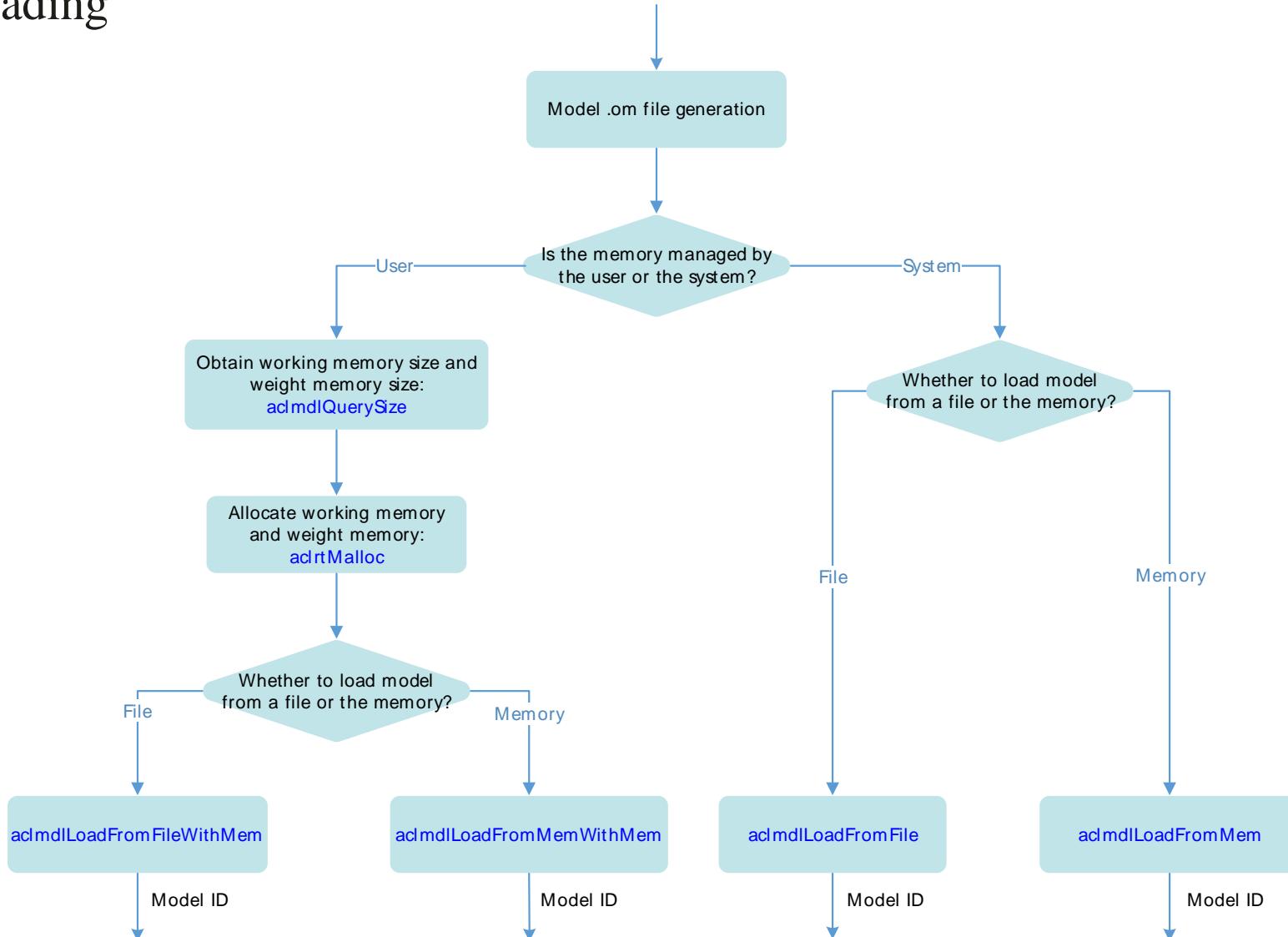


```
atc --model=cafe_model/resnet50.prototxt \
--weight= cafe_model/resnet50.cafemodel \
--framework=0 \
--output=model/resnet50 \
--soc_version=Ascend310 \
--input_format=NCHW \
--output_type=FP32 \
--out_nodes=prob:0
```

ATC workflow

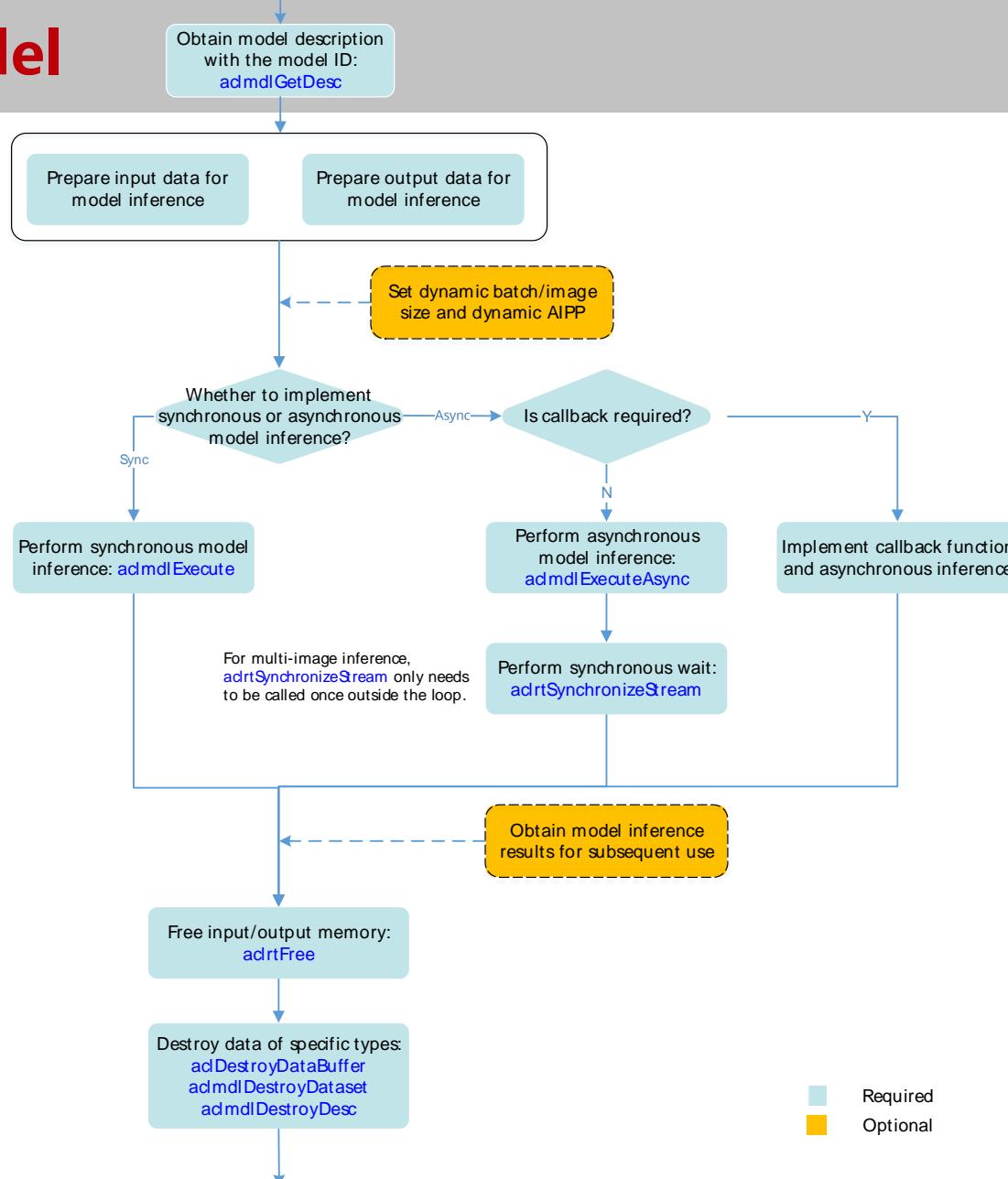
Typical APIs - Executing a Model

➤ Model Loading



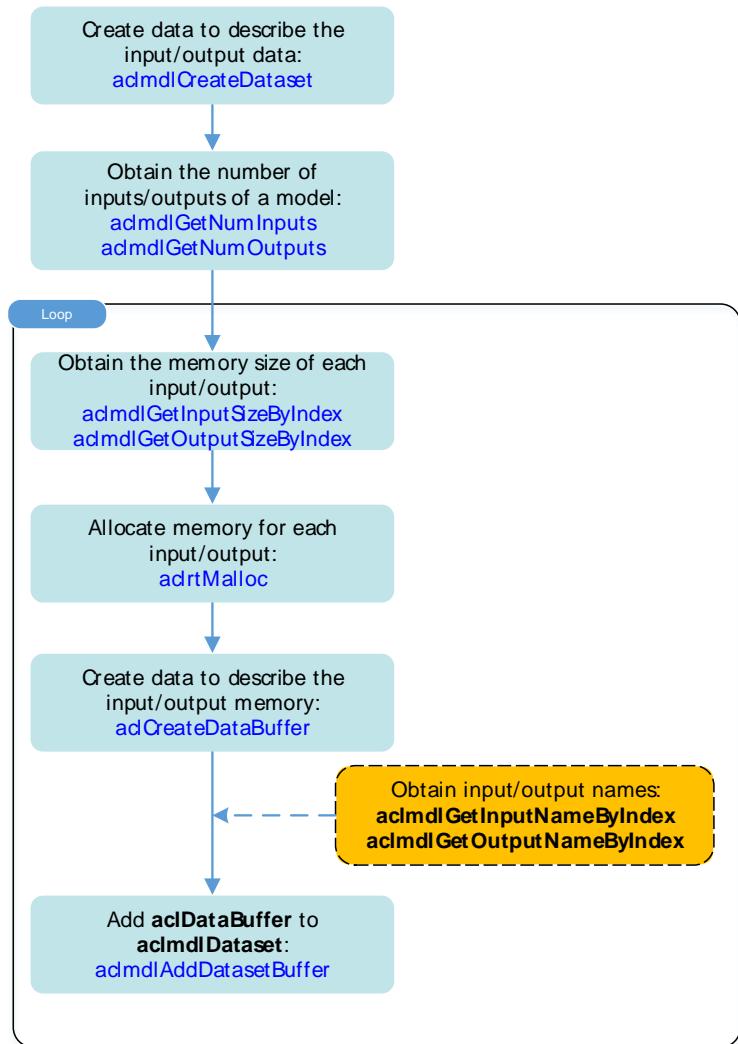
Typical APIs - Executing a Model

➤ Model Inference

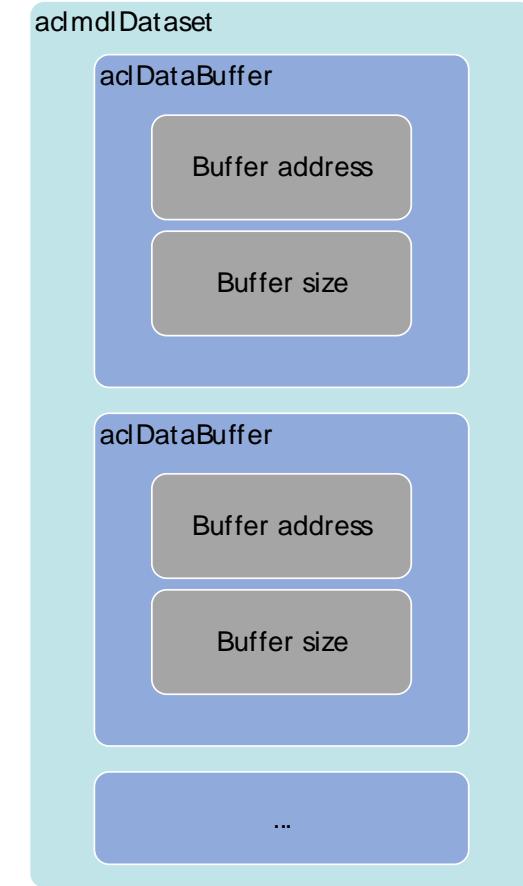


Typical APIs - Executing a Model

➤ Input and Output Data Preparation



Required
Optional



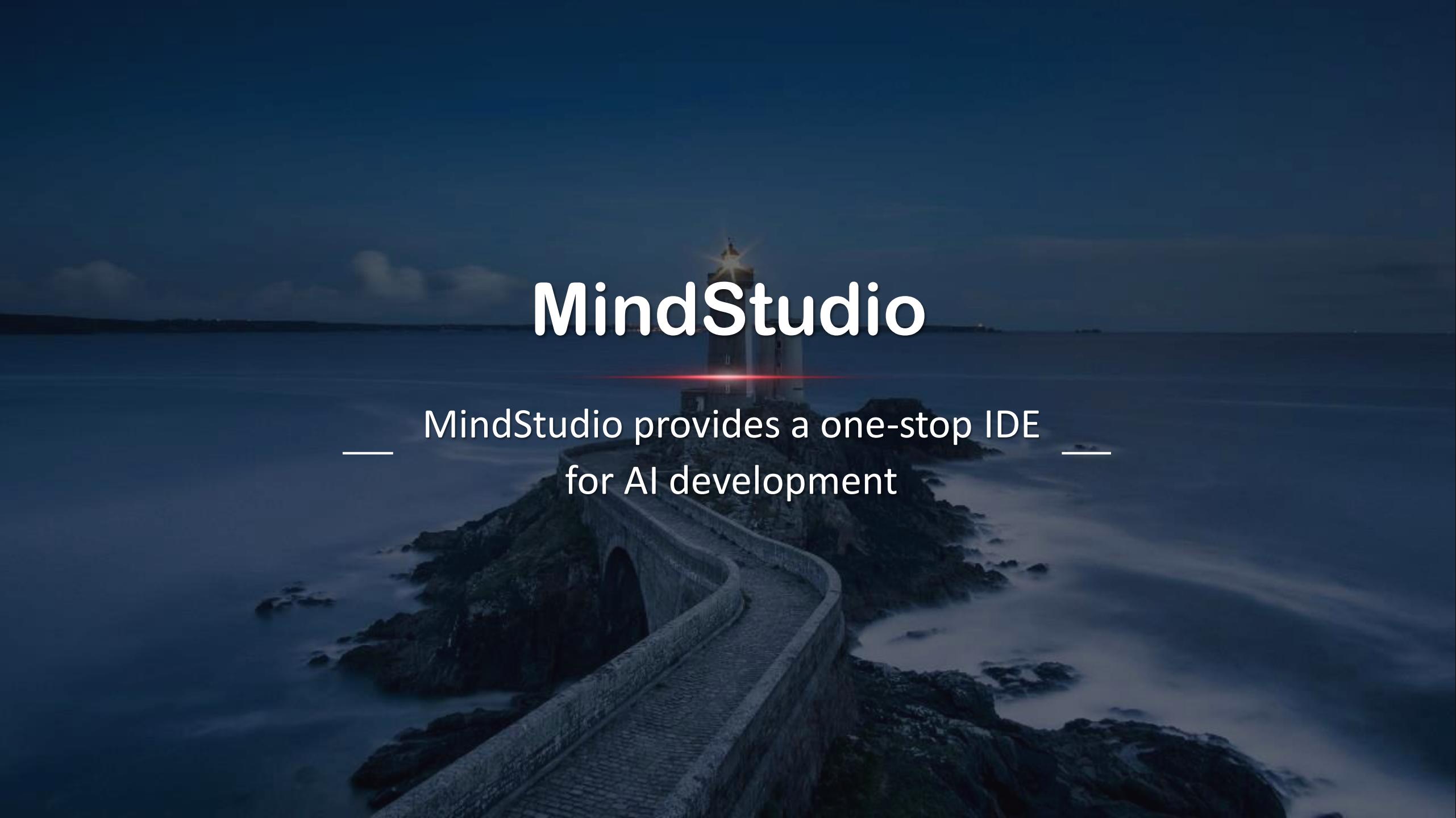
`aclmdlDataset` and `aclDataBuffer`

Contents



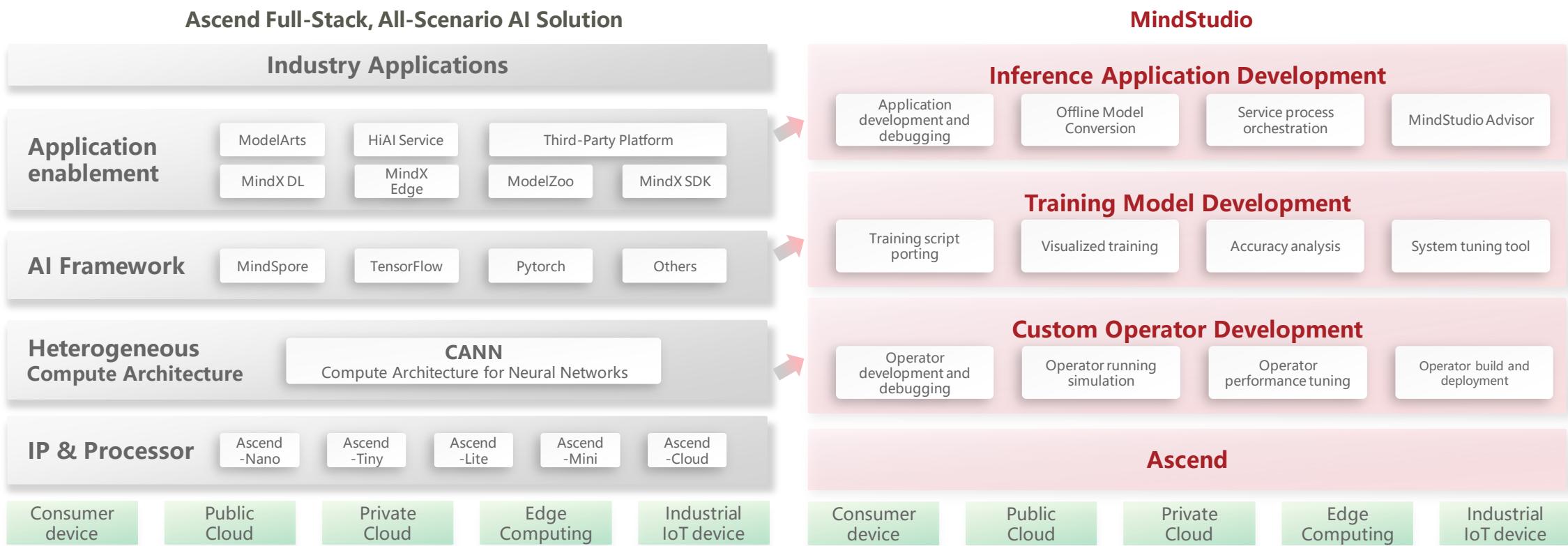
- 1. Huawei Da Vinci Architecture & Atlas Products**
- 2. CANN: Compute Architecture for Neural Networks**
- 3. ACL: Ascend Compute Language**
- 4. MindStudio: FullPline Development Toolchain**
- 5. MindX: Ascend Computing Application Enablement Kit**
- 6. Mindspore: Opensource AI Framework**

MindStudio

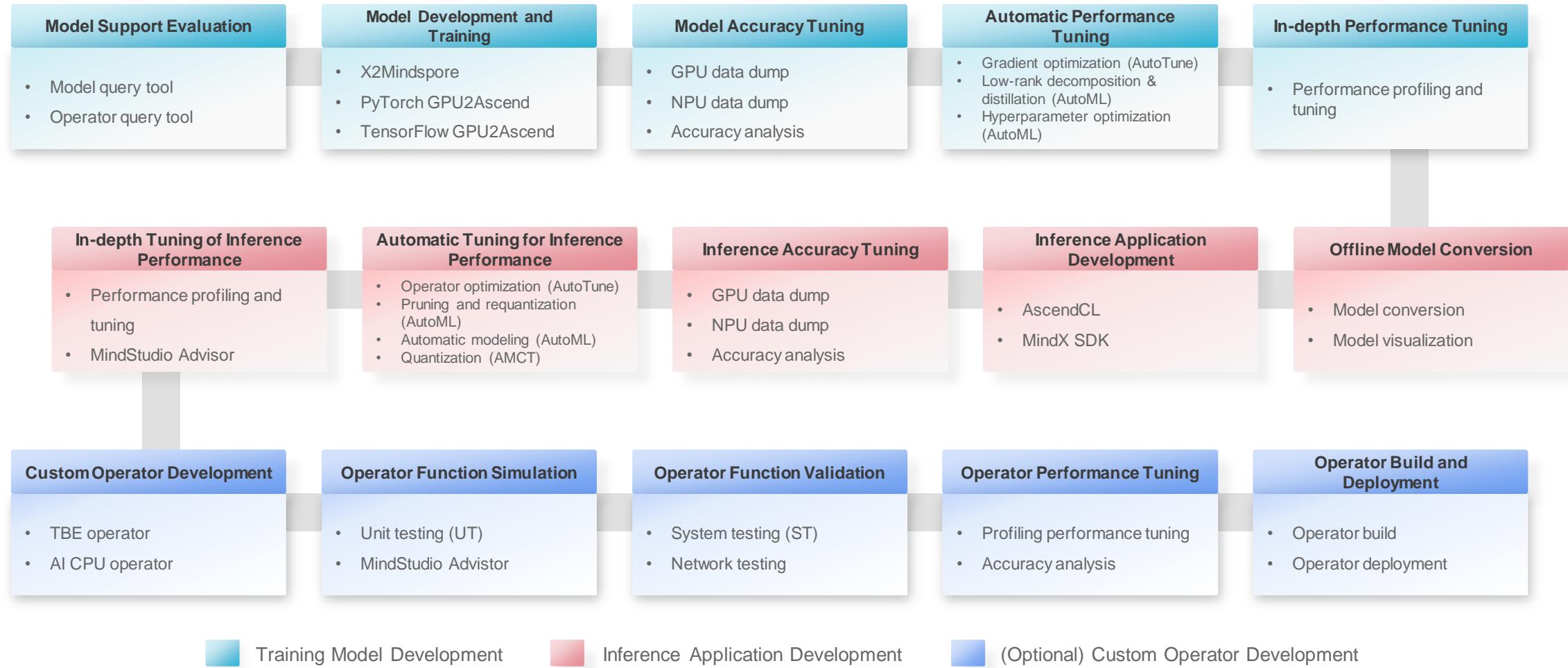


— MindStudio provides a one-stop IDE
for AI development —

MindStudio is an integrated development environment (IDE). It supports basic functions such as code development, build, debugging, and running in Python and C/C++. As a full-pipeline development toolchain in the Ascend AI full stack, MindStudio provides E2E tools for developing **training models**, **inference applications**, and **custom operators**, greatly improving the efficiency.



Functions of MindStudio





MindStudio

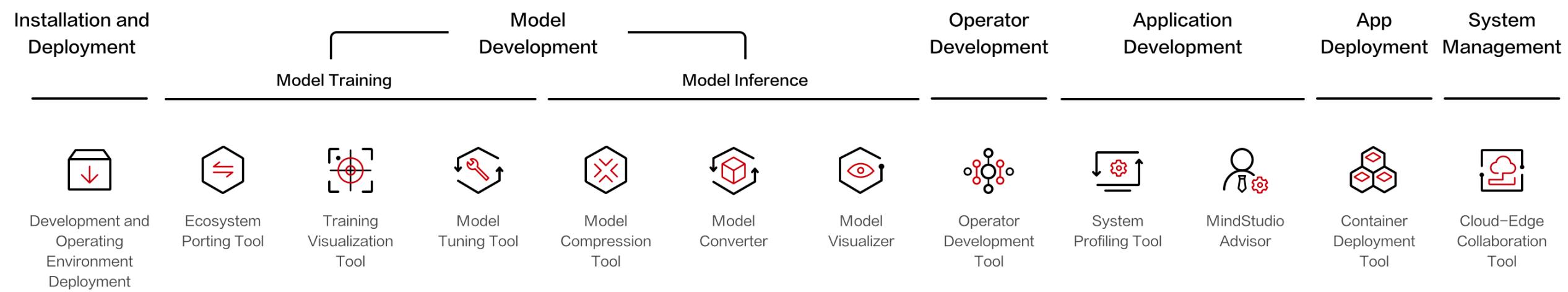
Model Development



Ascend | ©2020 Huawei Technologies Co., Ltd.

Feature of MindStudio

Development Process



Model Training on MindStudio

MindStudio support multiple AI frameworks. You can easily port your 3th party (**PyTorch**, **Tensorflow**) model with no or minimal changes to code.

Support for multiple AI frameworks

TensorFlow

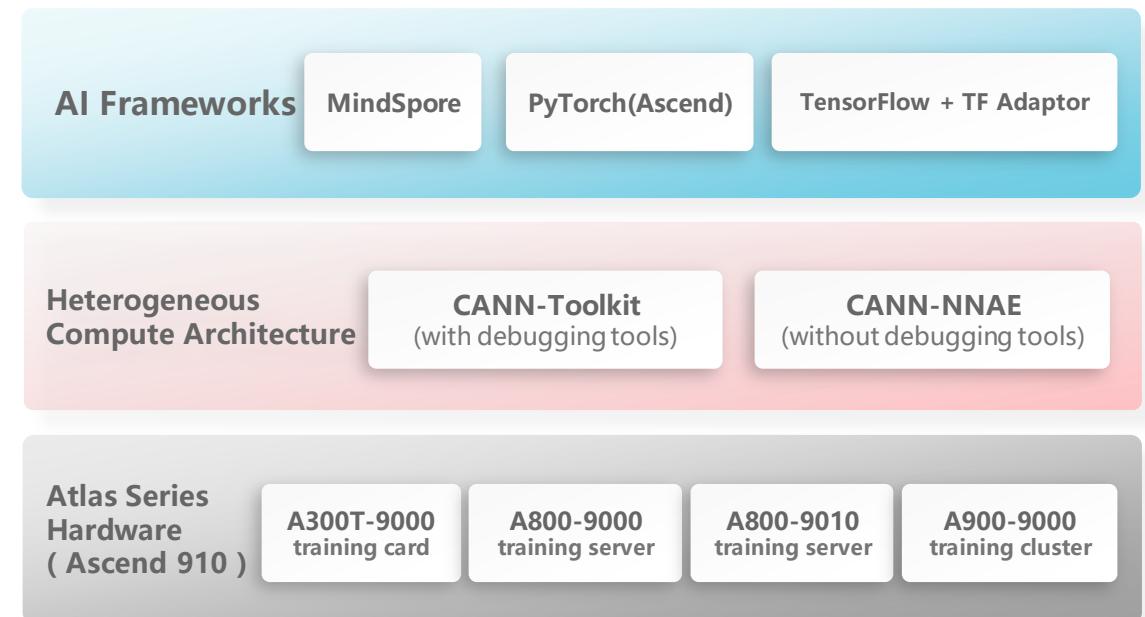
TensorFlow and TF Adapter are combined to support Ascend hardware version 1.15.

PyTorch

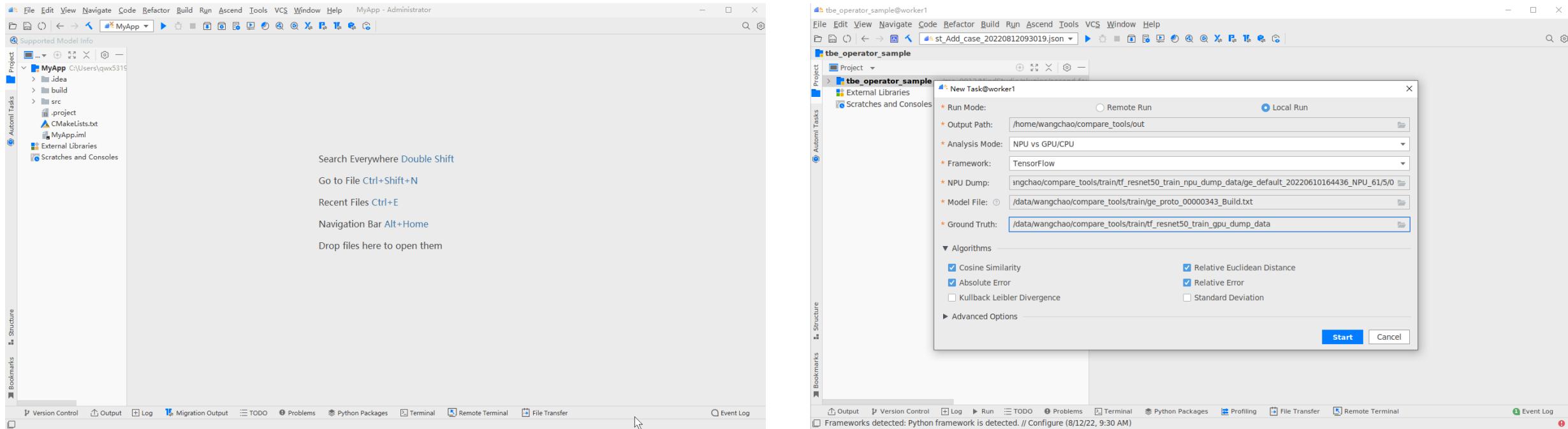
The PyTorch source code is modified to support Ascend hardware (NPU) versions 1.5 and 1.8.

MindSpore

The Huawei-developed AI framework that perfectly matches Ascend hardware.

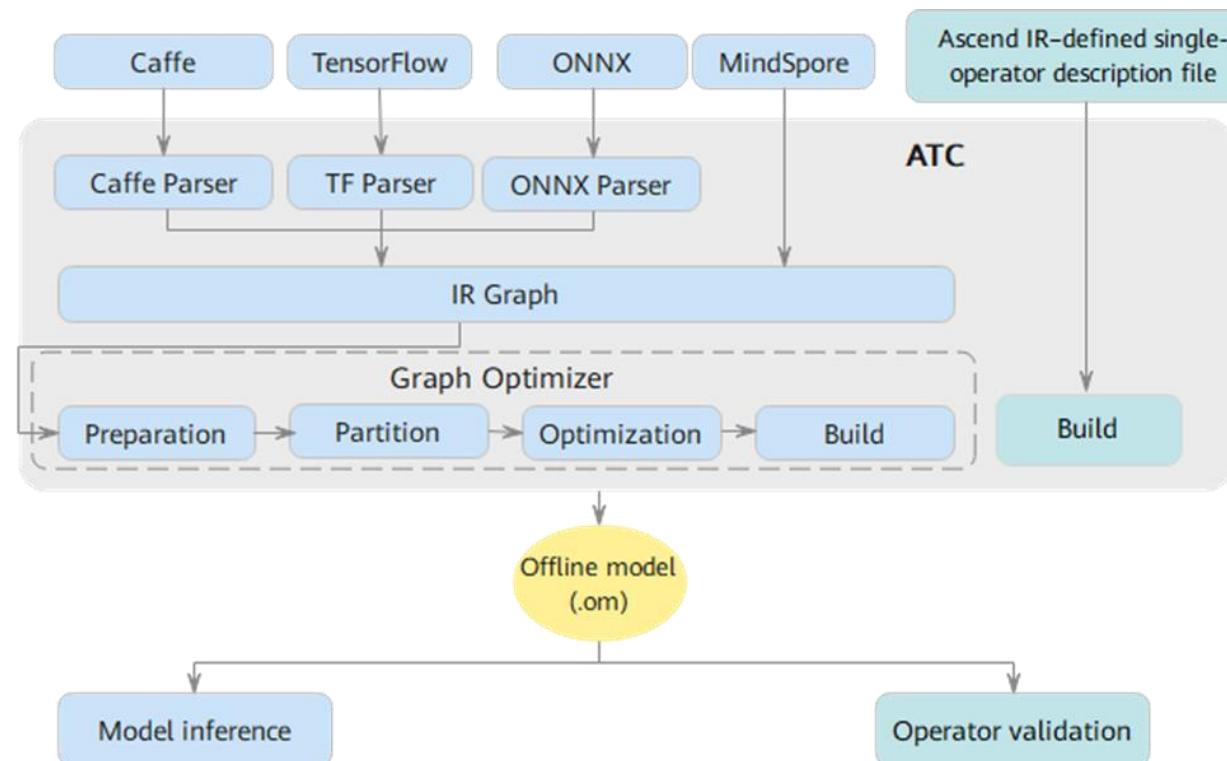


Model Training on MindStudio



Model Inference on MindStudio

Inference application development more specifically based on offline models. Before inference execution, you need to use the **ATC** (Ascend Tensor Compiler) to convert the network model of the open-source framework into online model supported by the Ascend AI processor.



Model Inference on MindStudio

The image displays three sequential steps of the Model Converter process:

- Model Information:** Shows the input model file as `/home/mindstudio/models/yolov3/yolov3_tf.pb`, target SoC version as `Ascend910A`, and output path as `/home/mindstudio/modelzoo/yolov3_tf_bs1_fp16/Ascend910A`. It also lists input nodes and output nodes.
- Data Pre-Processing:** Configures data preprocessing mode to `Static`, sets input node to `(input/input_data)`, defines input image format as `YUV420 sp` and resolution as `H: 416, W: 416`, and specifies model image format as `RGB`. It includes options for Crop, Padding, and Normalization, and shows conversion type as `UINT8->FP16` with mean values (R: 104, G: 117, B: 123) and variance values (R: 0.0039216, G: 0.0039216, B: 0.0039216).
- Advanced Options:** Displays command preview showing the ATC command generated for the conversion, including parameters like `--soc_version=Ascend910A` and `--model=/home/mindstudio/models/yolov3/yolov3_tf.pb`.

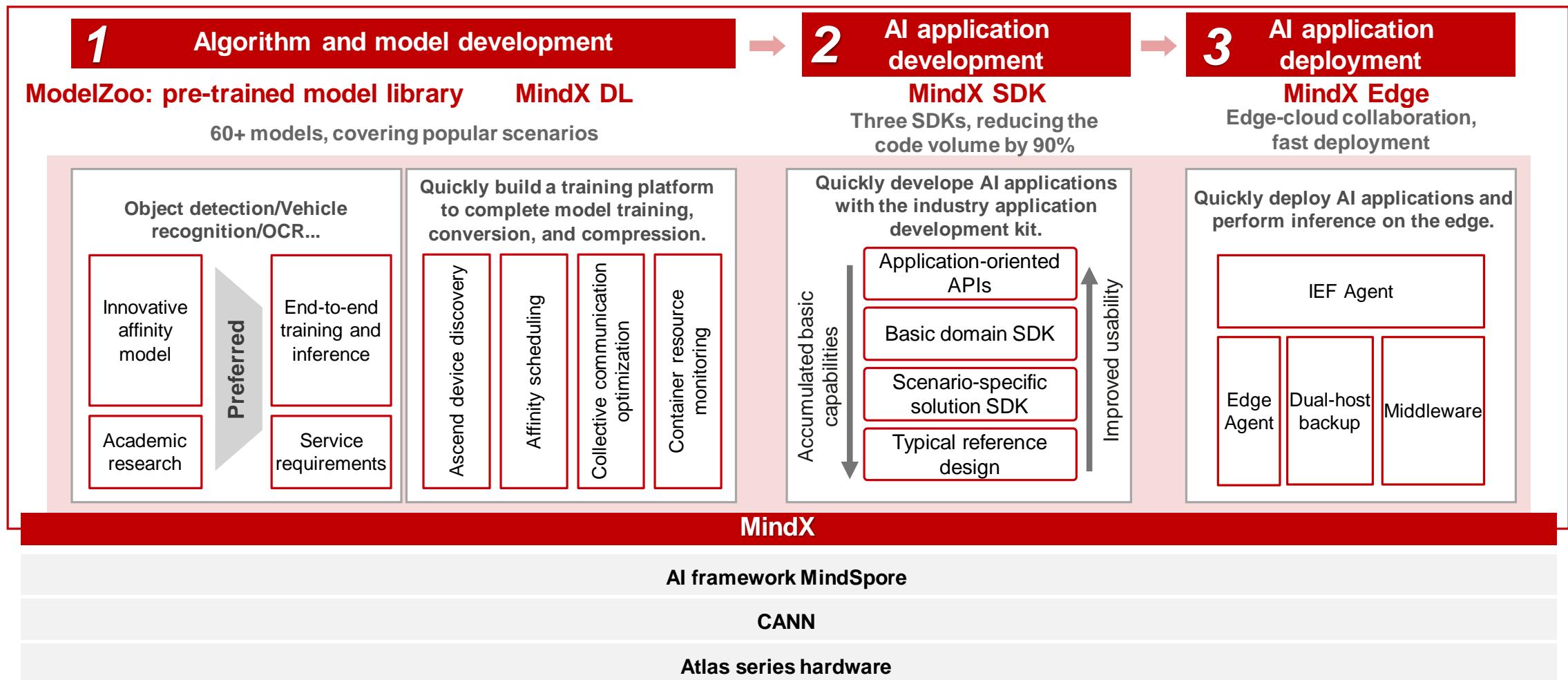
MindX

A photograph of a lighthouse at night. The lighthouse is illuminated, casting a bright beam of light that cuts through the dark, cloudy sky. Below the lighthouse, a long, curved pier or causeway extends into the ocean, which is dark and reflects the surrounding clouds. The overall atmosphere is mysterious and dramatic.

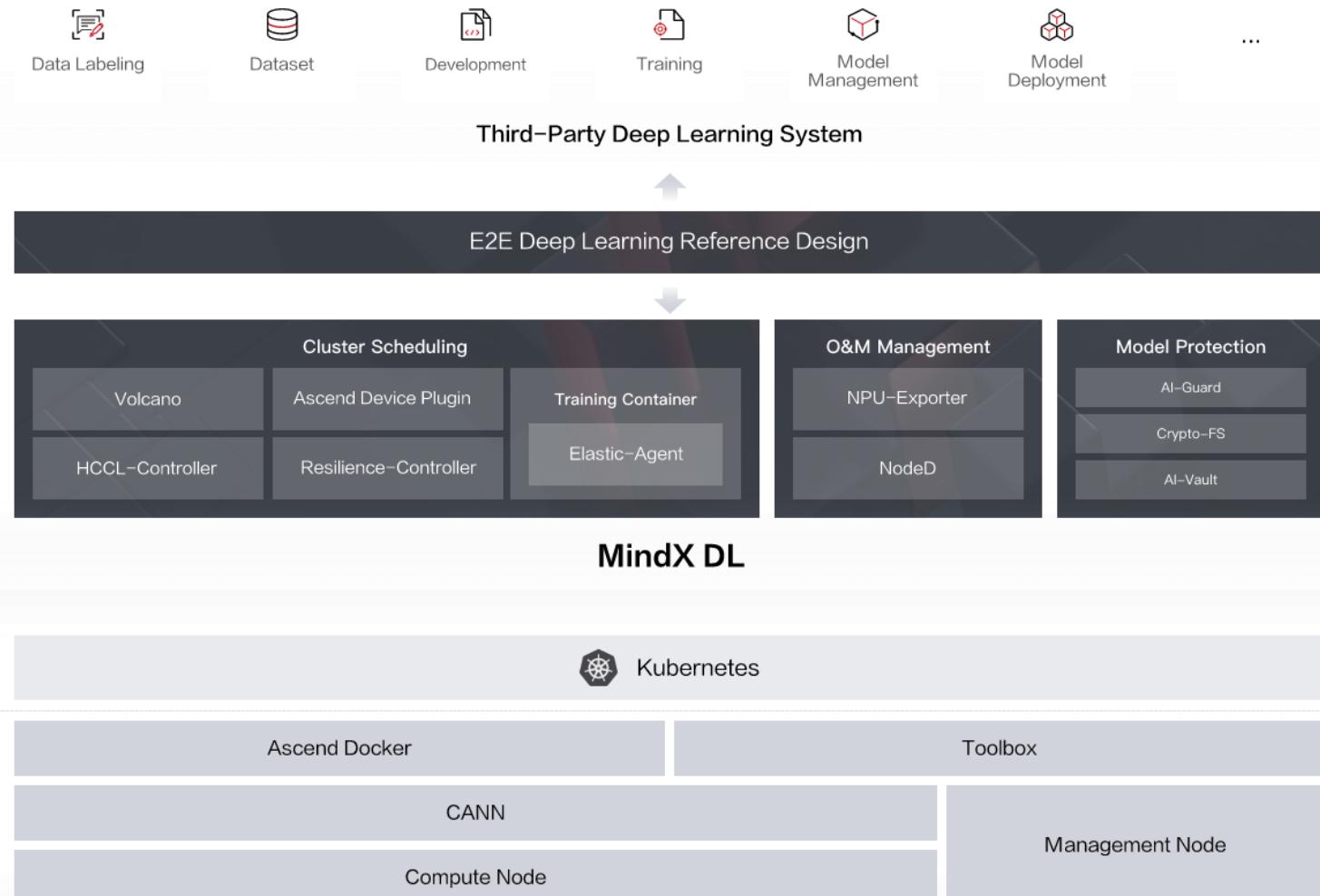
Bridging the Gap in AI Application
Implementation and Enabling Quick
Rollout of AI Applications

Overview of MindX

Huawei Ascend computing application enablement kit, bridges the gap in AI application implementation and enables quick rollout of AI applications.

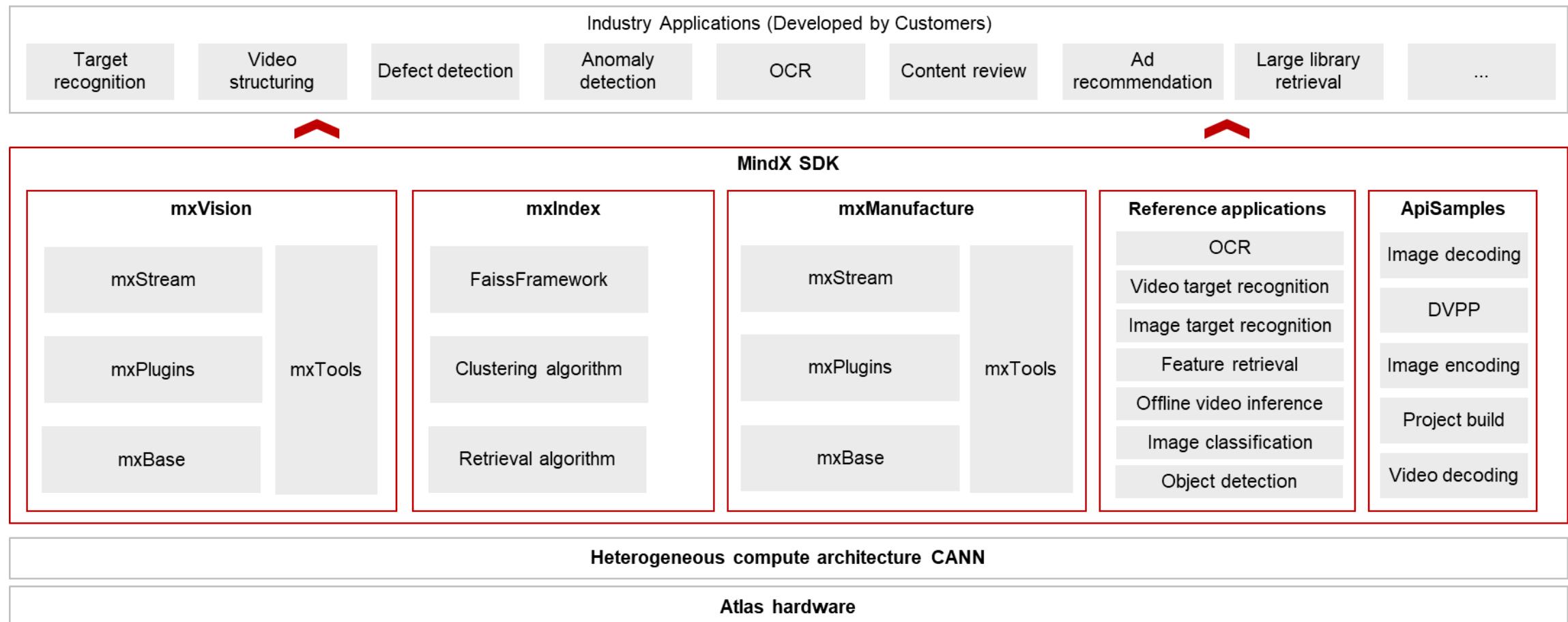


Overview of MindX DL



Overview of MindX SDK

MindX SDK accelerates high-performance AI application deployment and enables industries. It delivers various AI software development kits (SDKs) for Ascend AI Processor acceleration and provides simplified APIs to accelerate the development of high-performance AI applications for thousands of industries.



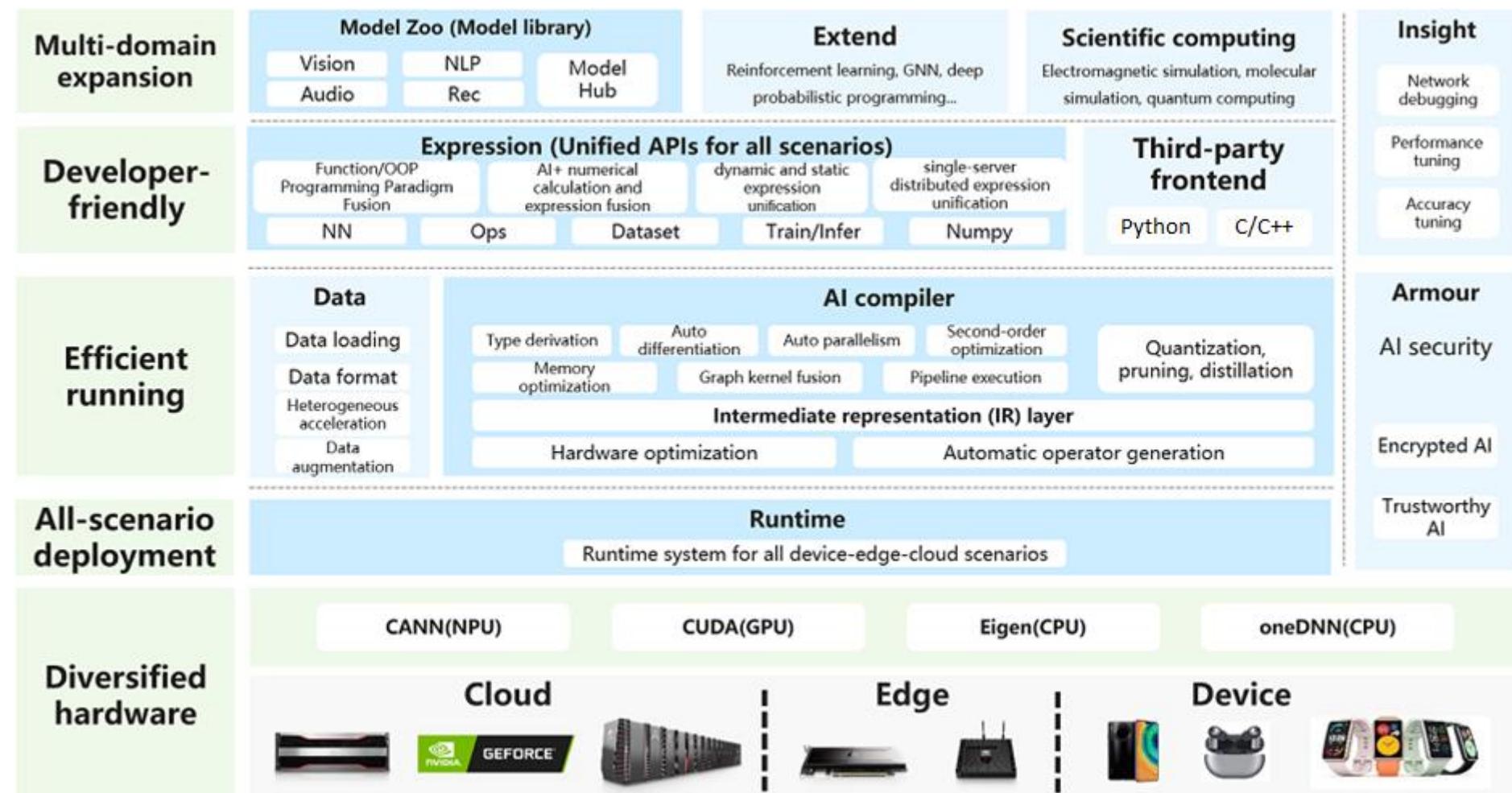


MindSoore

One-stop development platform
designed for AI developers

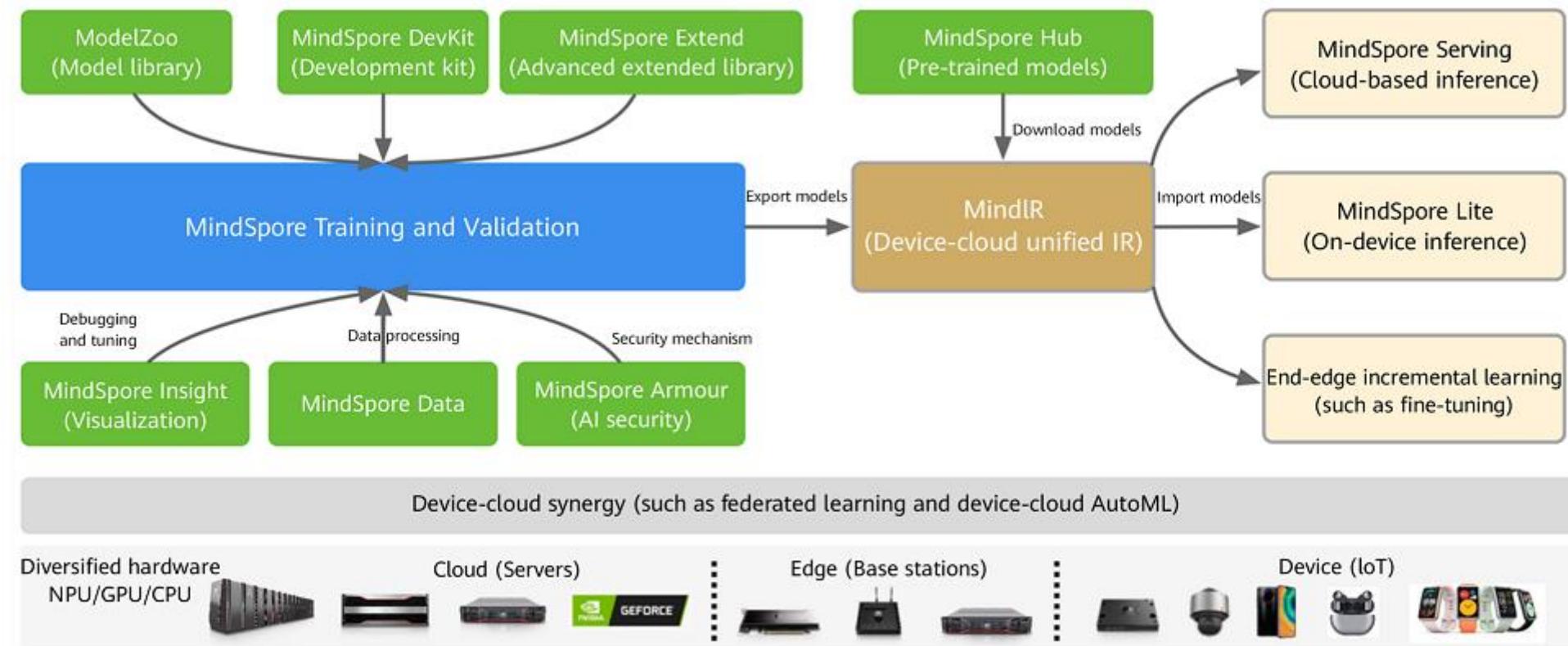
Overview of MindSpore

MindSpore is a deep learning framework in all scenarios, aiming to achieve easy development, efficient execution, and all-scenario coverage.

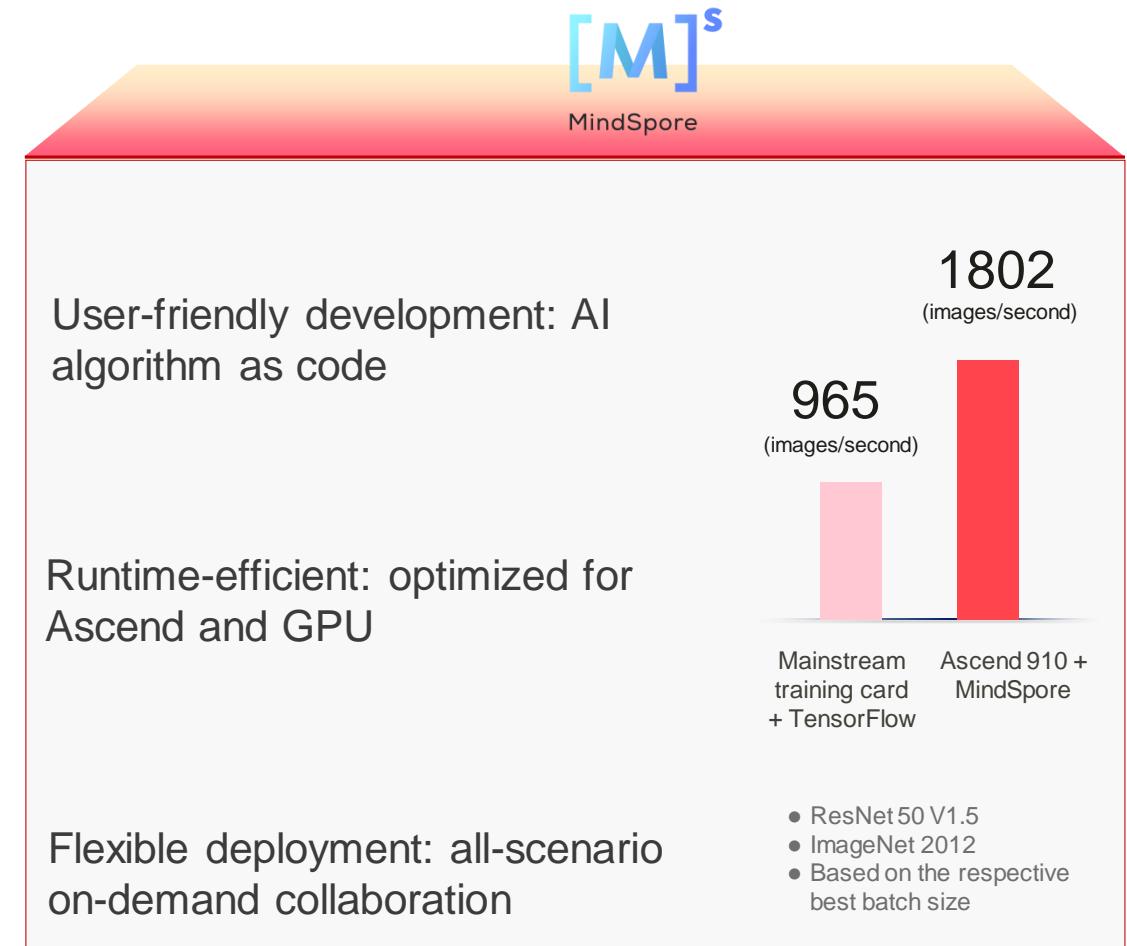
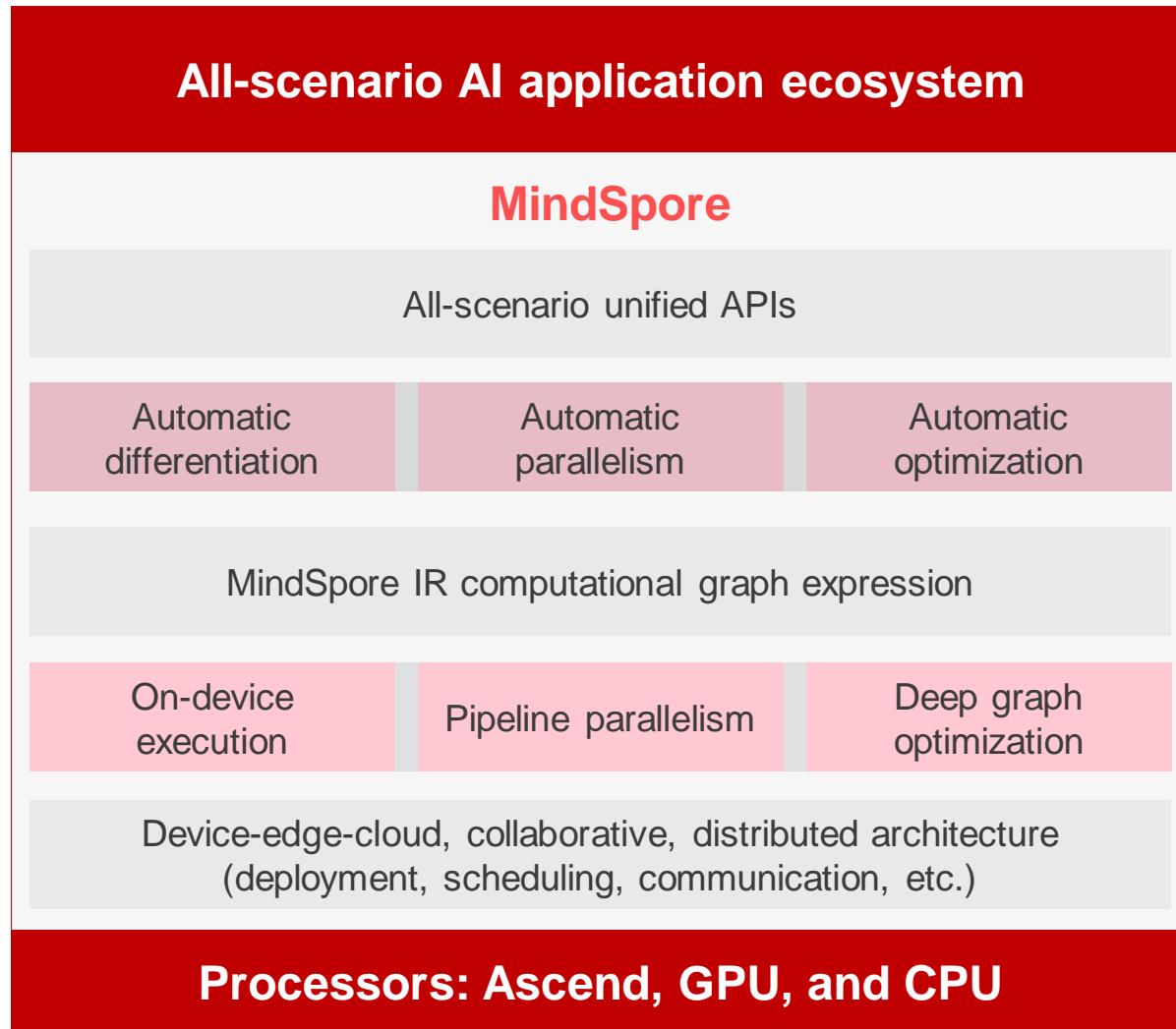


Executing Process of MindSpore

With an understanding of the overall architecture of MindSpore, we can look at the overall coordination relationship between the various modules, as shown in the figure:



Open Source AI Framework MindSpore: All-Scenario AI Computing Framework

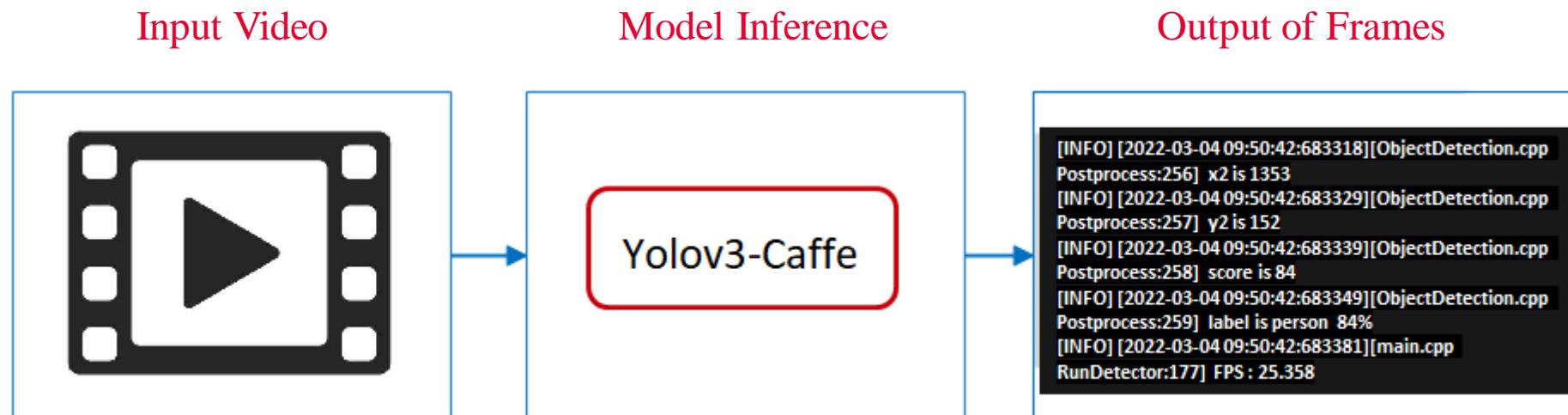


Time to Hands On

Ascend Turbocharges Intelligent Transformation

Overview of The ACL Single Stream Inference

- An application developed by ACL for single stream CV detection task using yolov3-caffe model.
- In this example, we will adapt a Caffe model to NPU infrastructure using ATC and then finally we will run this OM model using ACL.
- The throughput of model will be better than the original model, but there will be some tiny reduction (like 0.2) in accuracy.
- Thereby, we will able to run your models on GPUs with high performance and distribute it on edge and cloud devices.



Preparing Code, Model and Video

Step1: Get the source code model and video;

Source code:

https://github.com/haicgu/training/tree/main/AI/hands-on/inference/offline/acl_multi_stream_inference

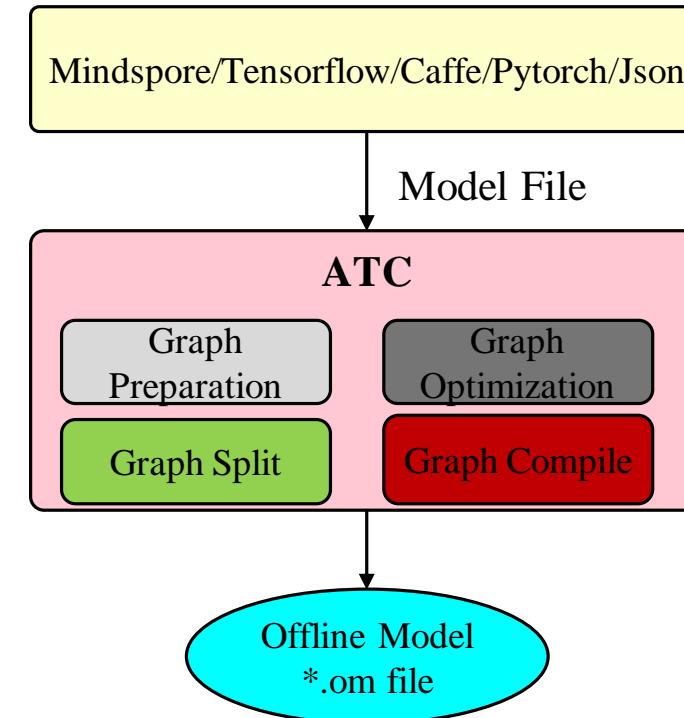
Caffe model:

https://modelzoo-train-atc.obs.cn-north-4.myhuaweicloud.com/003_Atc_Models/AE/ATC%20Model/Yolov3/yolov3.caffemodel

Video:

https://obs-9be7.obs.cn-east-2.myhuaweicloud.com/models/YOLOV4_coco_detection_car_video/test_video/test.mp4

Step2: Model conversion



```
atc --model=yolov3.prototxt \
--weight=yolov3.caffemodel \
--framework=0 \
--output==/data/model/yolov3 \
--soc_version=Ascend310 \
--insert_op_conf=./aipp_yolov3_416_no_csc.cfg
```

Run The Sample

Step3: Compile and execute the Demo

Compile

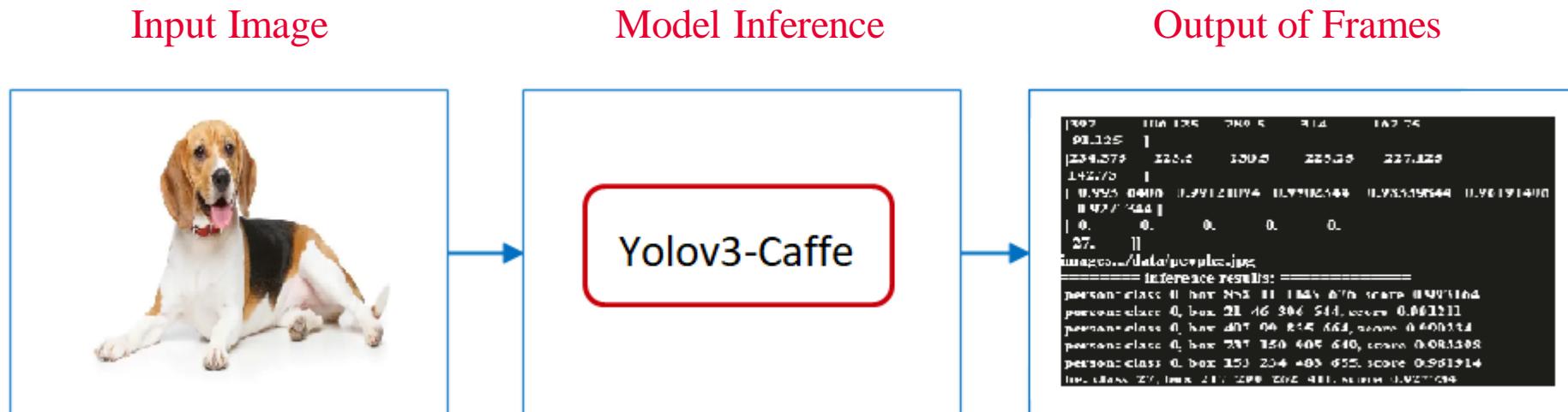
```
./build.sh  
cd dist  
./main
```

Execute

```
[INFO] [2022-03-04 09:50:42:683318][ObjectDetection.cpp Postprocess:256] x2 is 1353  
[INFO] [2022-03-04 09:50:42:683329][ObjectDetection.cpp Postprocess:257] y2 is 152  
[INFO] [2022-03-04 09:50:42:683339][ObjectDetection.cpp Postprocess:258] score is 84  
[INFO] [2022-03-04 09:50:42:683349][ObjectDetection.cpp Postprocess:259] label is person 84%  
[INFO] [2022-03-04 09:50:42:683381][main.cpp RunDetector:177] FPS : 25.358  
[INFO] [2022-03-04 09:50:42:807613][ObjectDetection.cpp Postprocess:254] x1 is 1223  
[INFO] [2022-03-04 09:50:42:807655][ObjectDetection.cpp Postprocess:255] y1 is 1  
[INFO] [2022-03-04 09:50:42:807667][ObjectDetection.cpp Postprocess:256] x2 is 1351  
[INFO] [2022-03-04 09:50:42:807677][ObjectDetection.cpp Postprocess:257] y2 is 152
```

Overview of The PyACL Object Detection Inference

- An application developed by PyACL for single stream CV detection task using yolov3-caffe model.
 - In this example, we will adapt a Caffe model to NPU infrastructure using ATC and then finally we will run this OM model using PyACL.
 - The throughput of model will be better than the original model, but there will be some tiny reduction (like 0.2) in accuracy.
 - Thereby, we will able to run our models on GPUs with high performance and distribute it on edge and cloud devices.



Preparing Code, Model and Video

Step1: Get the source code model;

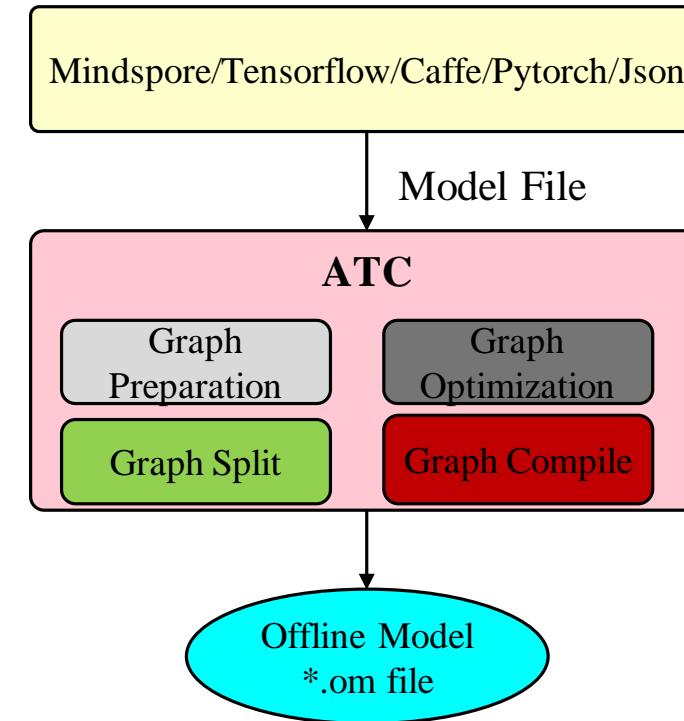
Source code:

https://github.com/haicgu/training/tree/main/AI/hands-on/inference/offline/acl_yolov3_detection

Caffe model:

https://modelzoo-train-atc.obs.cn-north-4.myhuaweicloud.com/003_Atc_Models/AE/ATC%20Model/Yolov3/yolov3.caffemodel

Step2: Model conversion



```
atc --model=yolov3.prototxt \
--weight=yolov3.caffemodel \
--framework=0 \
--output==/data/model/yolov3 \
--soc_version=Ascend310 \
--insert_op_conf=./aipp_yolov3_416_no_csc.cfg
```

Run The Sample

Step3: Execute the Demo

Run

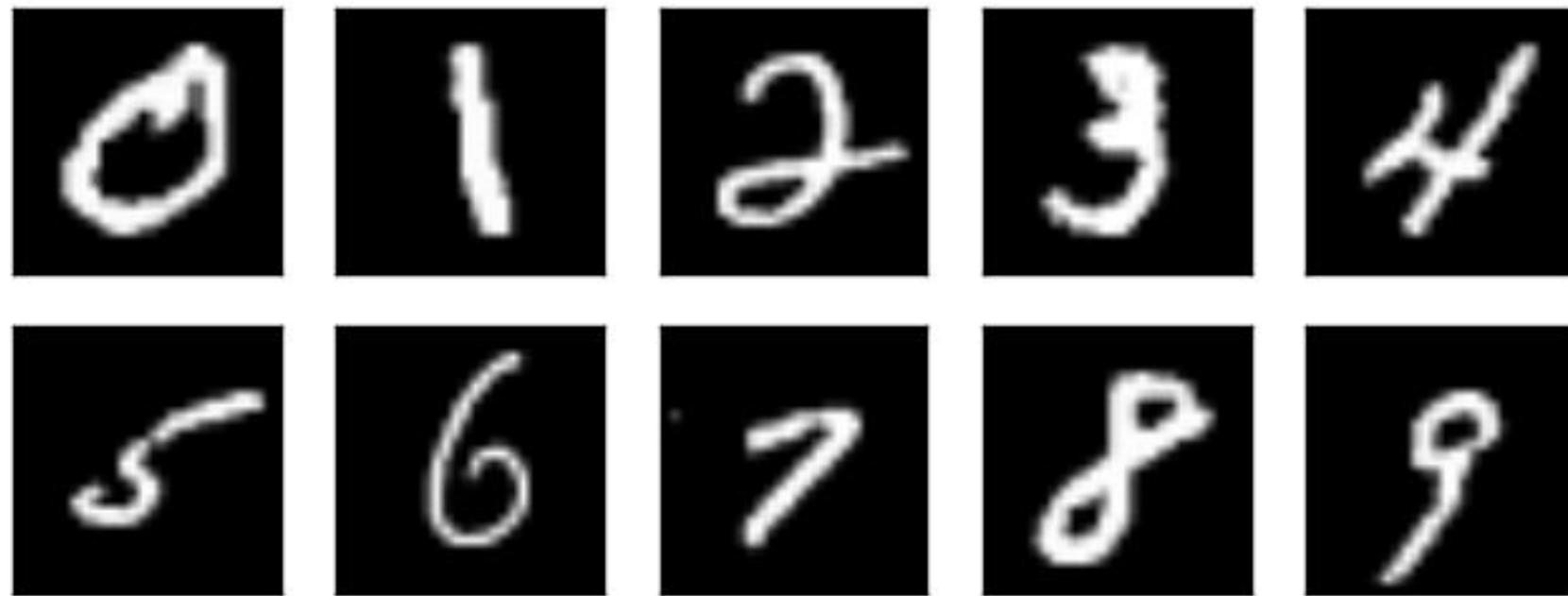
```
python3 object_detect.py ./data/
```

Execute

```
[397.    106.125   289.5    314.    167.75  
 91.125  ]  
[234.375   223.5    230.5    225.25   227.125  
 142.75  ]  
[ 0.99316406  0.99121094  0.9902344   0.98339844  0.96191406  
 0.9277344]  
[ 0.      0.      0.      0.      0.  
 27.    ]]  
images:./data/peoples.jpg  
===== inference results: =====  
person: class 0, box 858 11 1145 676, score 0.993164  
person: class 0, box 21 46 306 644, score 0.991211  
person: class 0, box 407 99 835 664, score 0.990234  
person: class 0, box 737 150 905 649, score 0.983398  
person: class 0, box 153 234 483 655, score 0.961914  
tie: class 27, box 217 290 262 411, score 0.927734
```

Overview of The Sample

- The goal of this lab is to familiarize with the concepts of AI framework NPU training. For this purpose, a simple training example is considered. Here is the Tensorflow LeNet MNIST training example.



Run The Sample

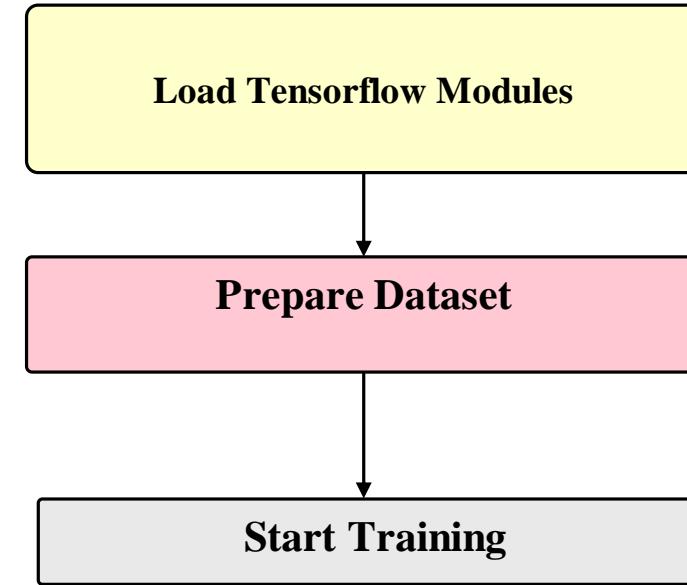
Source code:

<https://github.com/haicgu/training/tree/main/AI/hands-on/training/tensorflow lenet on npu>

```
module load GCC/9.5.0 OpenMPI TensorFlow-CANN/1.15.0
```

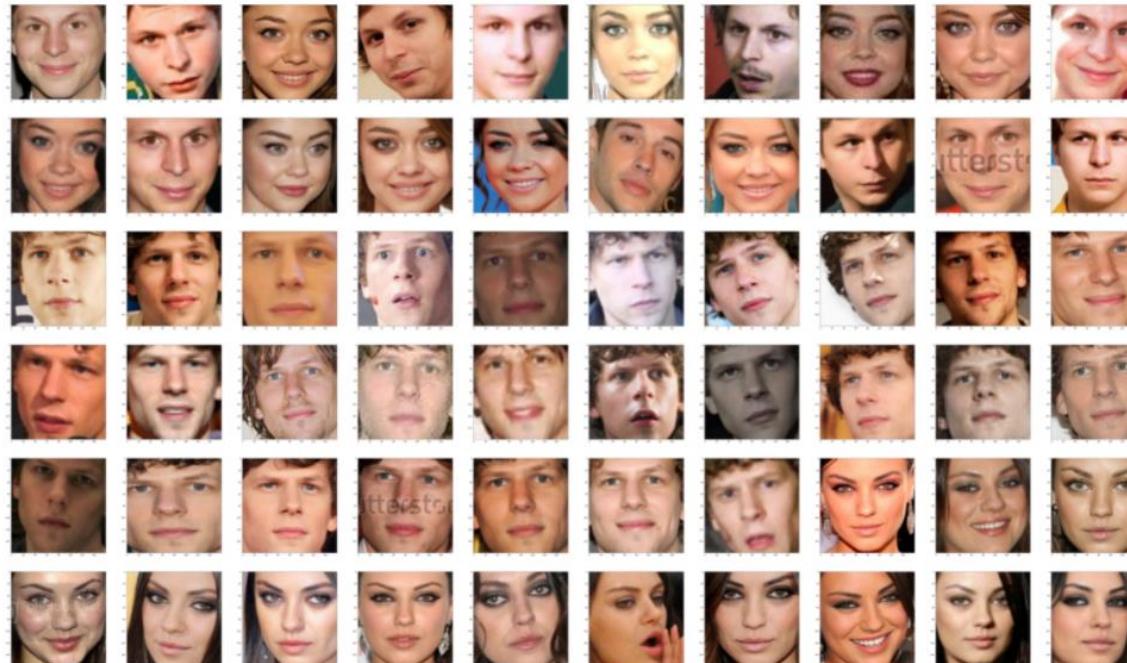
```
bash get_dataset.sh
```

```
#!/bin/bash
#SBATCH --partition=a800-9000
#SBATCH --time=00:10:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
npu-smi info
python3 LeNet.py ---data_path ./MNIST
EOF
```



Overview of The Sample

- Creating face recognition is considered to be a very easy task in the field of computer vision, but it is extremely tough to have a pipeline that can predict faces with complex backgrounds when you have multiple faces, different lighting conditions, and different scales of images. . Here is the Tensorflow Facenet inference example.



Run The Sample

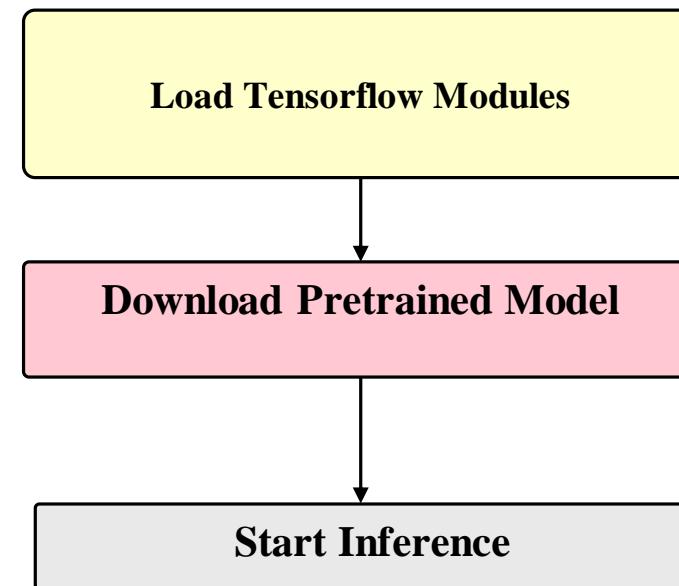
Source code:

https://github.com/haicgu/training/tree/main/AI/hands-on/inference/online/tensorflow_facenet_inference

```
module load GCC/9.5.0 OpenMPI TensorFlow-CANN/1.15.0
```

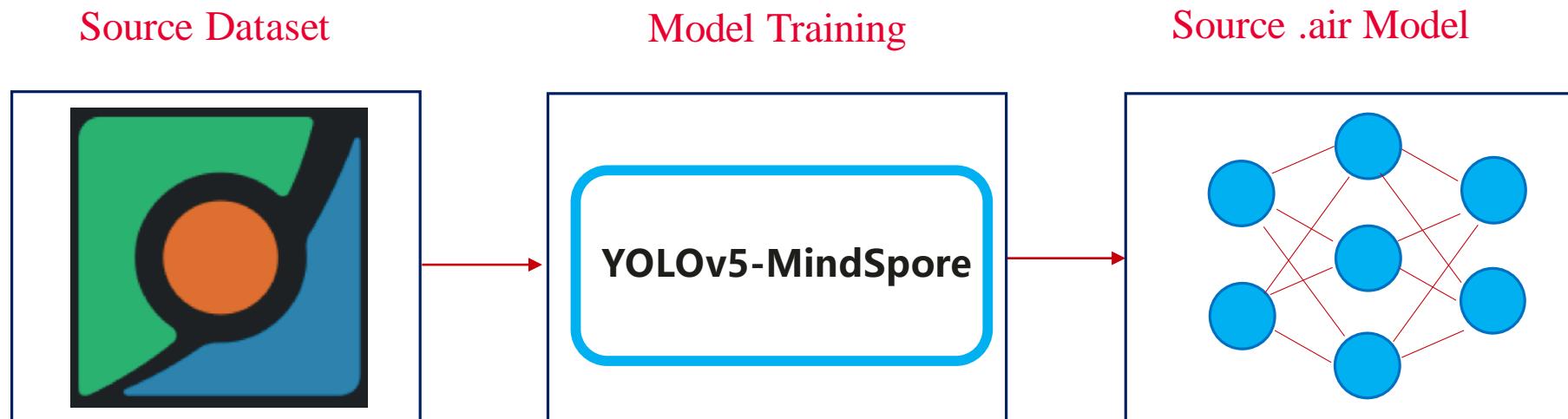
```
cd ./models  
bash get_model.sh
```

```
#!/bin/bash  
#SBATCH --partition=a800-3000  
#SBATCH --time=00:10:00  
#SBATCH --ntasks=1  
#SBATCH --nodes=1  
npu-smi info  
python3 main.py --model_path ./models/facenet_tf.pb --input_tensor_name  
input:0 --output_tensor_name embeddings:0 --image_path ./facenet_data  
EOF
```



Overview of MindSpore YOLOv5 Training Sample

- In this example, we will train a MindSpore model on NPU.
- Before starting the training, we need to define the dataset path in the yaml configuration file. Then we will start the model training.
- After the training is completed, the trained model will be an .air model which is designed for MindSpore.



Run MindSpore YOLOv5 Training Sample

Step1: Get the source code and the dataset

Source code:

https://github.com/haicgu/training/tree/main/AI/hands-on/training/mindspore_yolov5

COCO dataset:

<https://cocodataset.org/#download>

<http://images.cocodataset.org/zips/train2017.zip>

http://images.cocodataset.org/annotations/annotations_trainval2017.zip

Step2: Edit the configuration file



```
# General Options
yolov5_version: "yolov5s"
data_dir: "/home/share/coco/"
output_dir: "./output"
```

Run The Sample

Step2: Execute the Demo

Execute

```
python3 train.py \
--device_target="Ascend" \
--data_dir=/home/share/coco/ \
--yolov5_version='yolov5s' \
--is_distributed=0 \
--lr=0.01 \
--T_max=320 \
--max_epoch=320 \
--warmup_epochs=4 \
--train_per_batch_size=32
```

Output

```
INFO:epoch[1], iter[1401], loss:235.398219, fps:122.95 imgs/sec, lr:0.0009559224708937109, per step time: 260.26368141174316ms
INFO:epoch[1], iter[1501], loss:230.998582, fps:120.20 imgs/sec, lr:0.0010241538984701037, per step time: 266.2326240539551ms
INFO:epoch[1], iter[1601], loss:228.316826, fps:109.34 imgs/sec, lr:0.0010923853842541575, per step time: 292.66693592071533ms
INFO:epoch[1], iter[1701], loss:225.158565, fps:114.16 imgs/sec, lr:0.0011606168700382113, per step time: 280.30664920806885ms
INFO:epoch[1], iter[1801], loss:220.219454, fps:115.76 imgs/sec, lr:0.0012288482394069433, per step time: 276.4246368408203ms
INFO:epoch[1], iter[1901], loss:236.302552, fps:113.82 imgs/sec, lr:0.0012970797251909971, per step time: 281.1565065383911ms
INFO:epoch[1], iter[2001], loss:230.399118, fps:109.73 imgs/sec, lr:0.001365311094559729, per step time: 291.6242551803589ms
INFO:epoch[1], iter[2101], loss:225.763290, fps:112.56 imgs/sec, lr:0.001433542580343783, per step time: 284.28987979888916ms
```

Thank you.

Atlas Details

<https://www.hiascend.com/en/>

email

kubilay.tuna@huawei.com

serkan.celik@huawei.com

alper.balmumcu@huawei.com



Discussion

