HUAWEI | MUNICH RESEARCH CENTER

Da Vinci

— A scalable architecture for neural network computing

Heiko Joerg Schick
Chief Architect | Advanced Computing

Salli Moustafa
Senior Software Solution Architect

Presenting the work of many people at Huawei

Version 7

HUAWEI TECHNOLOGIES DÜSSELDORF GmbH

HUAWEI

Agenda /1

Introduction

- Computation in brains and machines
- The hype roller coaster of artificial intelligence | Neural networks beat human performance
- Two distinct eras of compute usage in training AI systems
- Microprocessor trends | Rich variety of computing architectures
- Comparison of processors for deep learning | Preferred architectures for compute are shifting
- Data structure of digital images | Kernel convolution example | Architecture of LeNet-5

Applicability of artificial intelligence

- Ubiquitous and future AI computation requirements
- Artificial intelligence in modern medicine

Product realisation

- Scalable across devices
- Focus on innovation, continuous dedication and backward compatibility
- HiSilicon Ascend 310 | HiSilicon Ascend 910 | HiSilicon Kungpeng 920

Da Vinci architecture

- Building blocks and compute intensity
- Advantages of special compute units
- Da Vinci core architecture | Micro-architectural configurations

Heiko Joerg Schick
Chief Architect | Advanced Computing
Munich Research Center

HUAWEI TECHNOLOGIES
Duesseldorf GmbH
Riesstrasse 25, C3
80992 Munich

Mobile +49-151-54682218
E-mail heiko.schick@huawei.com

LinkedIn
https://www.linkedin.com/in/heikojoergschick/

Agenda /2

End-to-end lifecycle

- Implementation of end-to-end lifecycle in AI projects
- The Challenges to AI implementations

Software stack

- Ascend AI software stack | Logical architecture
- Software flow for model conversion and deployment | Framework manager | Digital vision pre-processing
- Mind Studio | Model Zoo (excerpt)
- Chip enablement layer and Ascend Computing Language (ACL)

Gain more practical experiences

- Atlas 200 DK developer board | Application examples | Getting started | Environment deployment
- Ascend developer community
- Getting started with Atlas 200 DK developer board

Preparing the Ubuntu-based development environment

- Environment deployment
- Hardware and software requirements | About version 1.73.0.0
- Install environment dependencies
- Install the toolkit packages
- Install the media module device driver
- Install Mind Studio

Heiko Joerg Schick
Chief Architect | Advanced Computing
Munich Research Center

HUAWEI TECHNOLOGIES
Duesseldorf GmbH
Riesstrasse 25, C3
80992 Munich

Mobile +49-151-54682218
E-mail heiko.schick@huawei.com

LinkedIn
https://www.linkedin.com/in/heikojoergschick

Agenda /3

Create and write SD card image

- Setting up the operating environment

Boot and connect to the Atlas 200 DK developer board

- Power on the Atlas 200 DK developer board

Install third-party packages

- Installation of additional packages (FFmpeg, OpenCV and Python)

Heiko Joerg Schick
Chief Architect | Advanced Computing
Munich Research Center

HUAWEI TECHNOLOGIES
Duesseldorf GmbH
Riesstrasse 25, C3
80992 Munich

Mobile +49-151-54682218
E-mail heiko.schick@huawei.com

LinkedIn
https://www.linkedin.com/in/heikojoergschick

Agenda /4

Create the first project: Colourful Image Colourisation

- Network architecture
- Model inference architecture
- Model conversion
- Download project source code
- Loading, converting and building the project
- Setting the target device
- Running inference

Second project: Object detection

Third project: Body pose detection

Salli Moustafa
Senior Software Solution Architect
Munich Research Center

HUAWEI TECHNOLOGIES
Duesseldorf GmbH
Riesstrasse 25, C3
80992 Munich

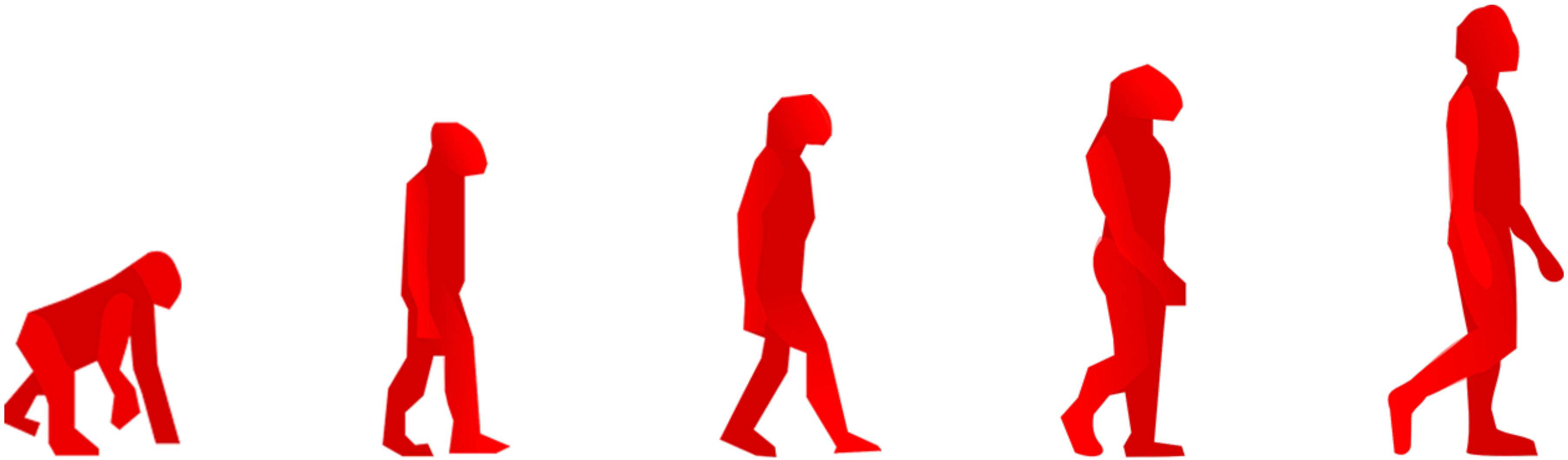
Mobile +49-176-1446713
E-mail salli.moustafa@Huawei.com

LinkedIn
https://www.linkedin.com/in/sallimoustafa

Introduction

"Nothing in the natural world makes sense — except when seen in the light of evolution."

– David Attenborough

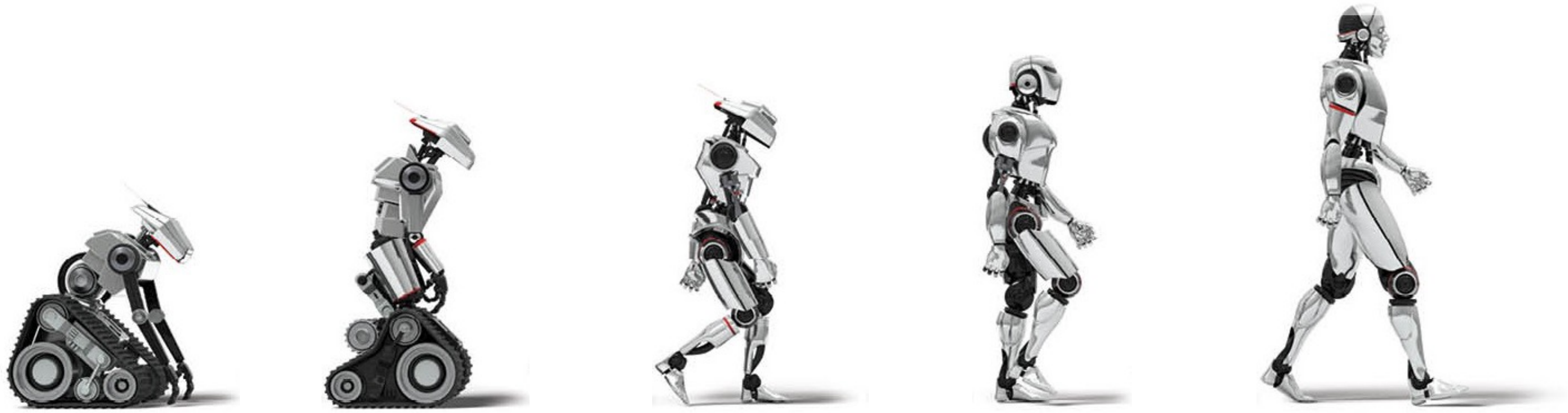


"It artificial intelligence would take off on its own and redesign itself at an ever increasing rate. Humans, who are limited by slow biological evolution, couldn't compete and would be superseded."

– *Stephen Hawking*

"Nothing vast enters the life of mortals without a curse."

– *Sophocles*

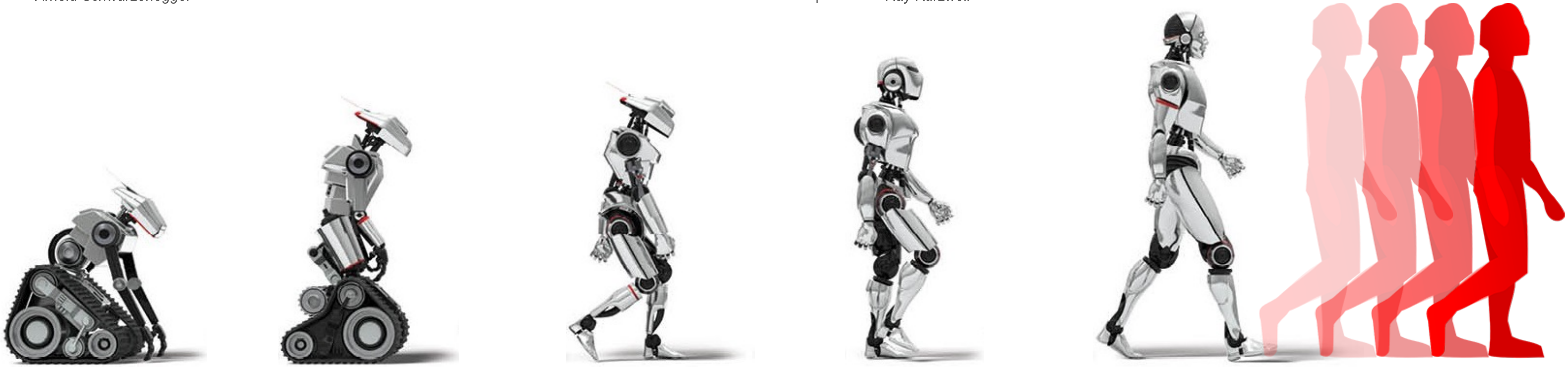


"We are going in the direction of artificial intelligence or hybrid intelligence where a part of our brain will get information from the cloud and the other half is from you, so all this stuff will happen in the future."

– Arnold Schwarzenegger

"Our technology, our machines, is part of our humanity. We created them to extend ourself, and that is what is unique about human beings."

– Ray Kurzweil

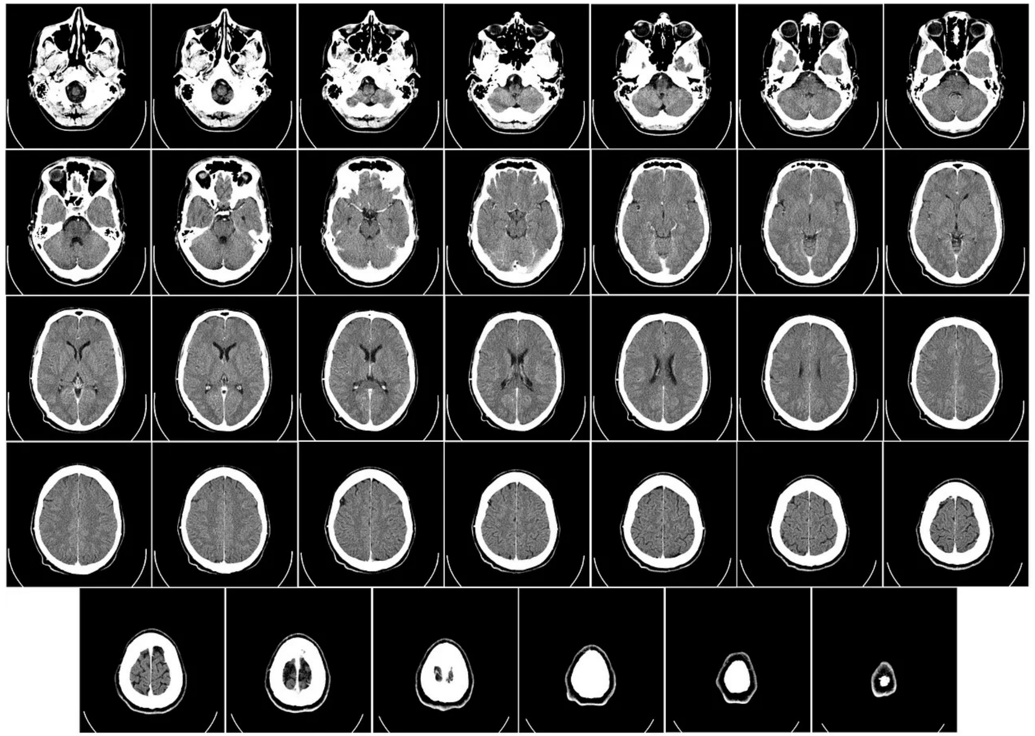


Computation in brains and machines [Conradt, 2014], [Wikipedia, 2020]

- **Getting to know your brain**
  - 1.3 kg, about 2% of body weight
  - 10^{11} neurons
  - Neuron growth:

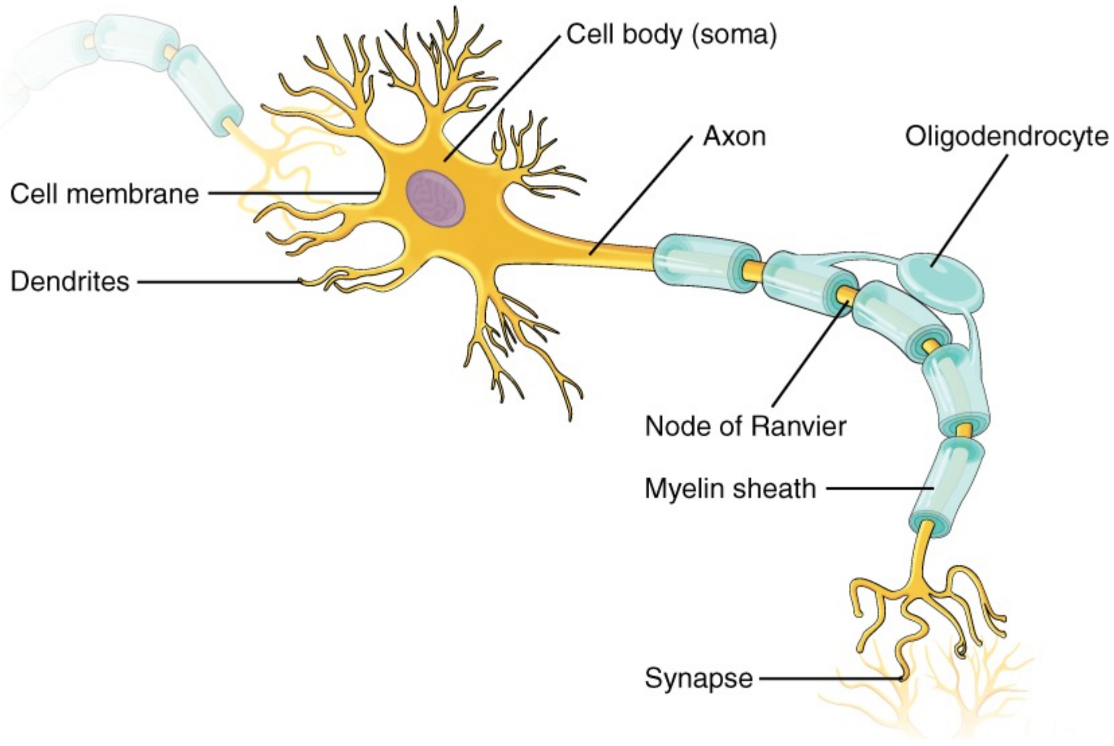
    250.000 / min (early pregnancy), but also loss

    1 neuron/second



Computation in brains and machines [Conradt, 2014], [Wikipedia, 2020]

- **Getting to know your brain**
  - 1.3 kg, about 2% of body weight
  - 10^{11} neurons
  - Neuron growth:
    250.000 / min (early pregnancy), but also loss
    1 neuron/second
- **Operating mode of neurons**
  - Analog leaky integration in soma
  - Digital pulses (spikes) along neurites
  - 10^{14} stochastic synapses
  - Typical operating “frequency”:
    ≦ 100 Hz, typically ~10 Hz, asynchronous



Computation in brains and machines [Conradt, 2014], [Wikipedia, 2020]

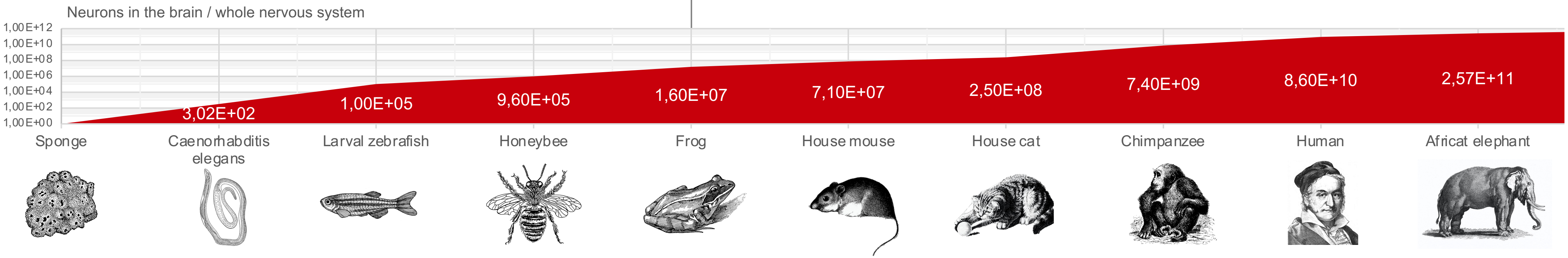
- **Getting to know your brain**
  - 1.3 kg, about 2% of body weight
  - 10^{11} neurons
  - Neuron growth:
    250.000 / min (early pregnancy), but also loss
    1 neuron/second
- **Operating mode of neurons**
  - Analog leaky integration in soma
  - Digital pulses (spikes) along neurites
  - 10^{14} stochastic synapses
  - Typical operating “frequency”:
    ≦ 100 Hz, typically ~10 Hz, asynchronous

- **Getting to know your computer’s processor**
  - 50g, irrelevant for most applications
  - 2,00E+10 transistors (HiSilicon Kunpeng 920)
  - Ideally no modification over lifetime
- **Operating mode of processors**
  - No analog components
  - Digital signal propagation
  - Reliable signal propagation
  - Typical operation frequency:
    several GHz, synchronous

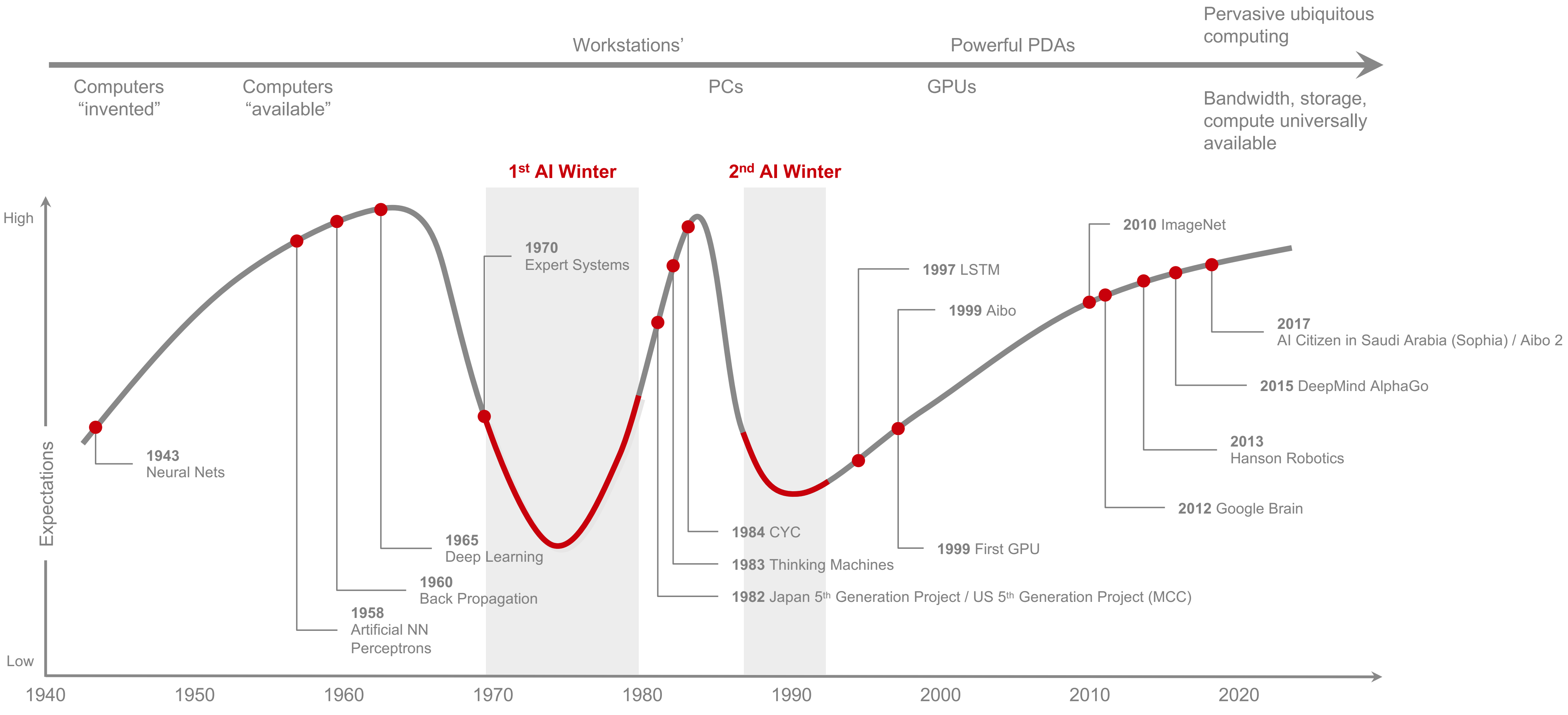
Computation in brains and machines [Conradt, 2014], [Wikipedia, 2020]

- **Getting to know your brain**
  - 1.3 kg, about 2% of body weight
  - 10^{11} neurons
  - Neuron growth:
    250.000 / min (early pregnancy), but also loss
    1 neuron/second
- **Operating mode of neurons**
  - Analog leaky integration in soma
  - Digital pulses (spikes) along neurites
  - 10^{14} stochastic synapses
  - Typical operating “frequency”:
    ≦ 100 Hz, typically ~10 Hz, asynchronous

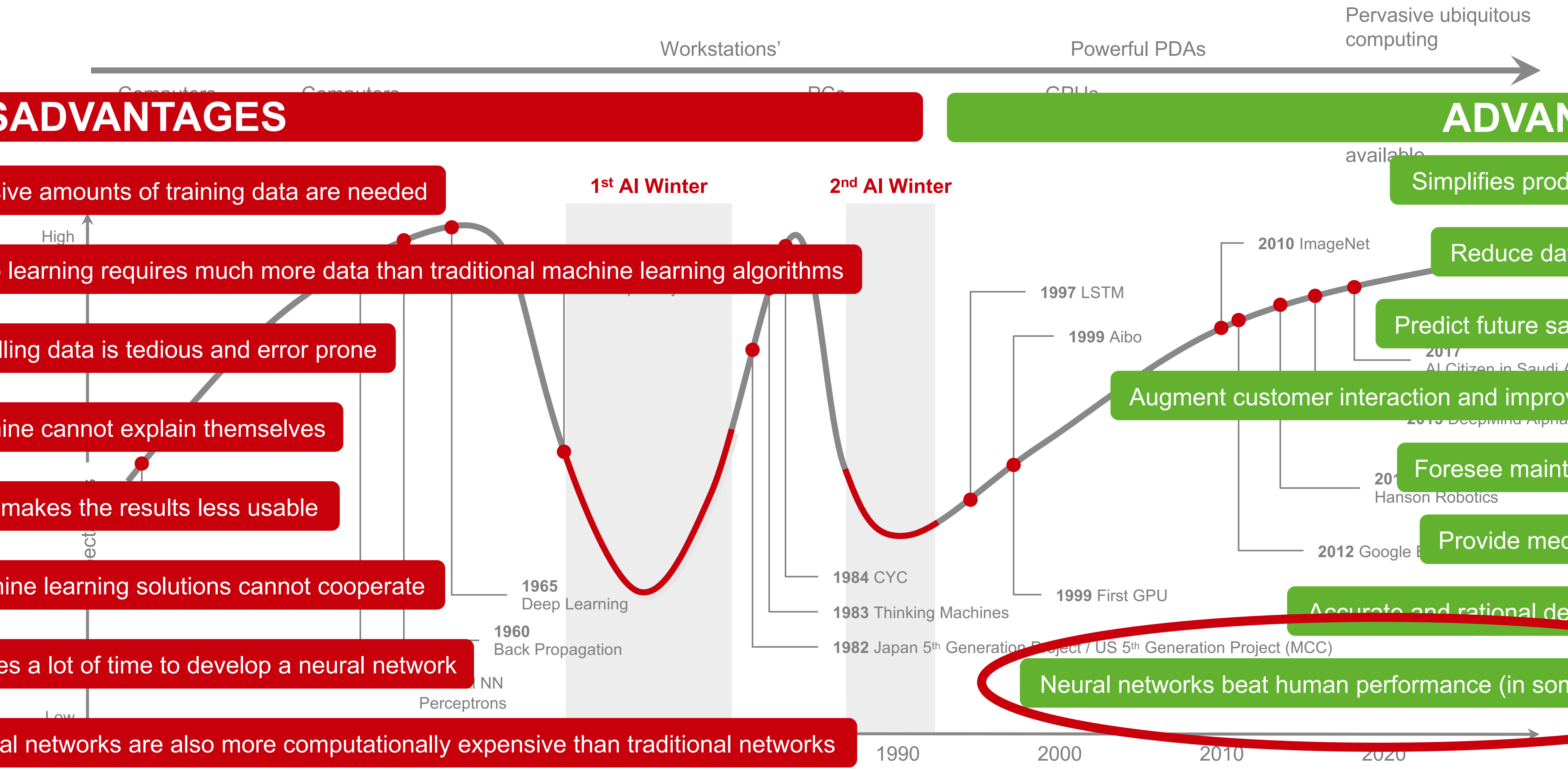
- **Getting to know your computer’s processor**
  - 50g, irrelevant for most applications
  - 2,00E+10 transistors (HiSilicon Kunpeng 920)
  - Ideally no modification over lifetime
- **Operating mode of processors**
  - No analog components
  - Digital signal propagation
  - Reliable signal propagation
  - Typical operation frequency:
    several GHz, synchronous



The hype roller coaster of artificial intelligence [Villain, 2019]



The hype roller coaster of artificial intelligence [Villain, 2019]



DISADVANTAGES

- Massive amounts of training data are needed
- Deep learning requires much more data than traditional machine learning algorithms
- Labelling data is tedious and error prone
- Machine cannot explain themselves
- Bias makes the results less usable
- Machine learning solutions cannot cooperate
- It takes a lot of time to develop a neural network
- Neural networks are also more computationally expensive than traditional networks

ADVANTAGES

- Simplifies product marketing
- Reduce data entry errors
- Predict future sales accurately
- Augment customer interaction and improve satisfaction
- Foresee maintenance needs
- Provide medical diagnosis
- Accurate and rational decision making
- Neural networks beat human performance (in some disciplines)

Neural networks beat human performance /1 [Giró-i-Nieto, 2016], [Gershgorn, 2017]

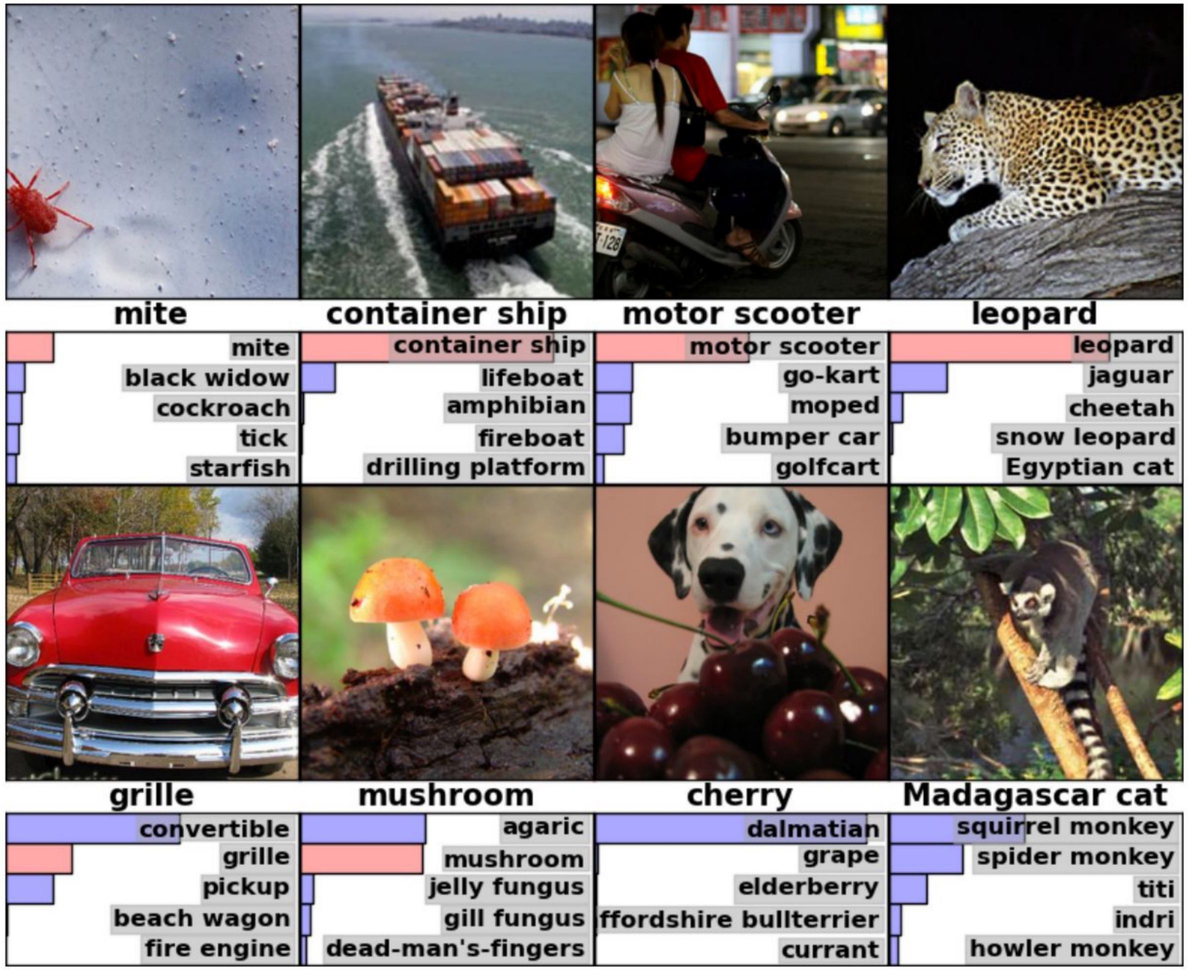
— <u>Example:</u> Image classification on ImageNet

15 million images in dataset, 22,000 object classes (categories) and 1 million images with bounding boxes.

Neural networks beat human performance /1 [Giró-i-Nieto, 2016], [Gershgorn, 2017]

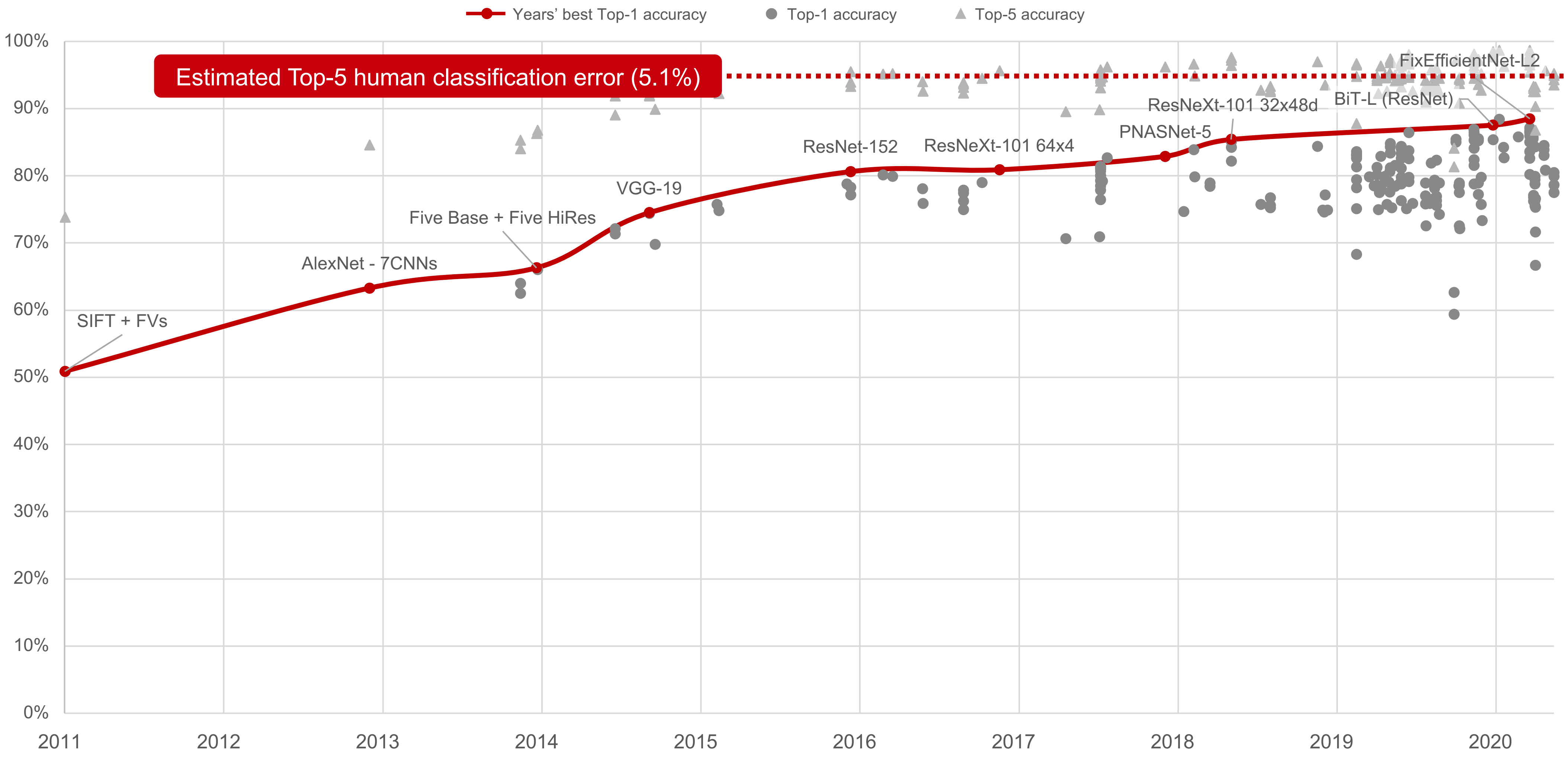
— **Example:** Image classification on ImageNet

15 million images in dataset, 22,000 object classes (categories) and 1 million images with bounding boxes.

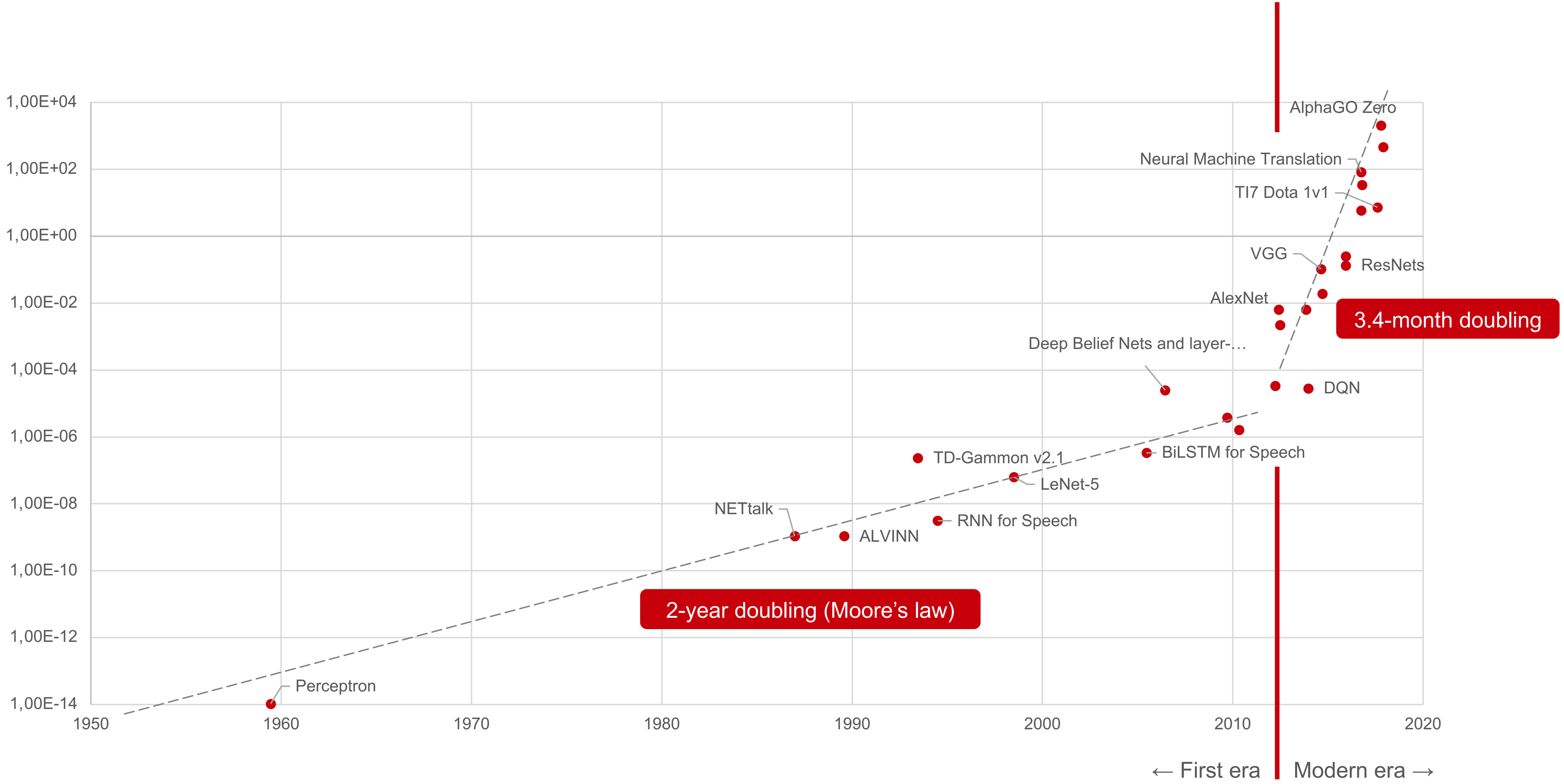


Neural networks beat human performance /2 [Russakovsky et al., 2015], [Papers With Code, 2020]

— **Example:** Image classification on ImageNet

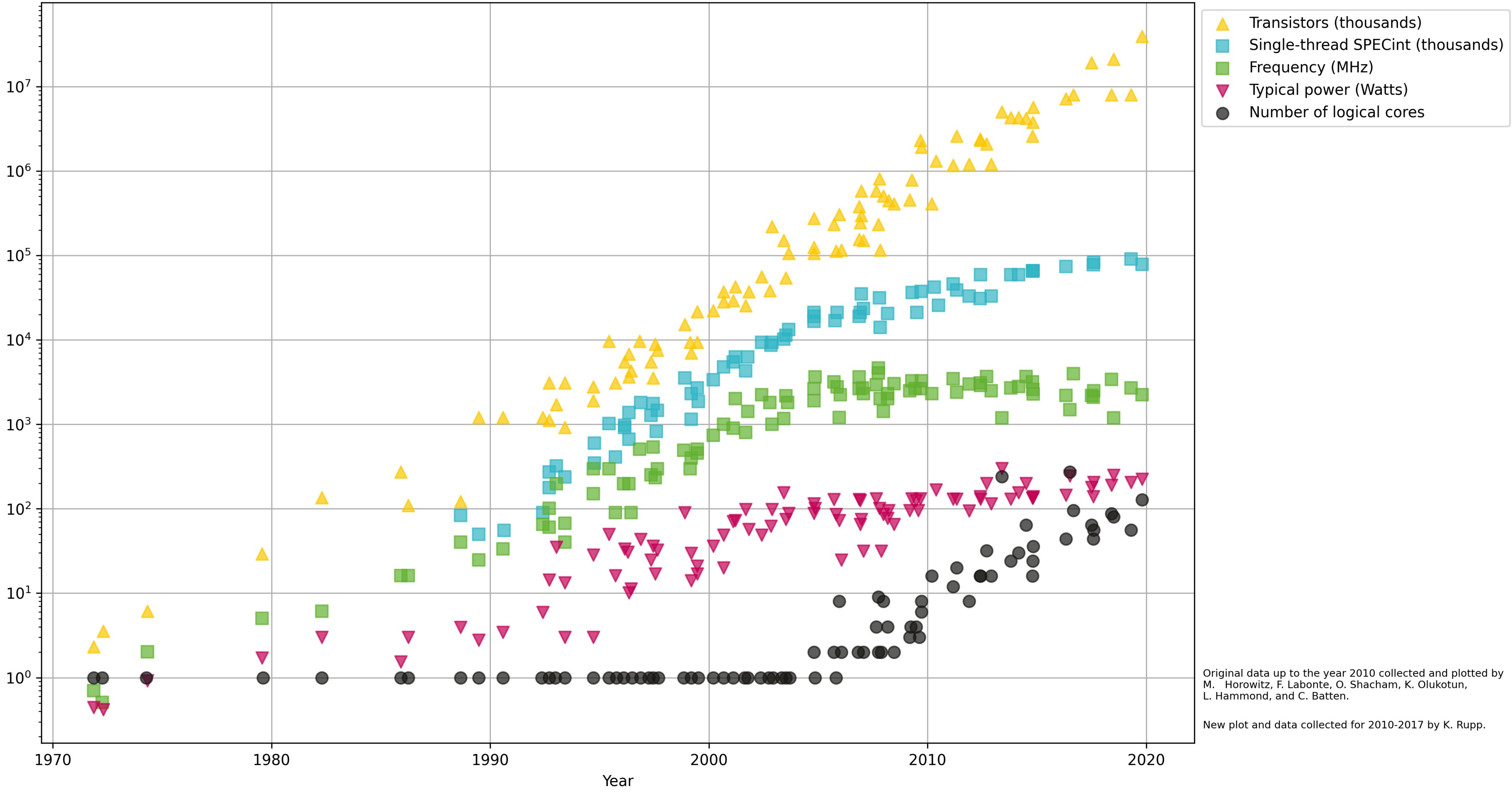


Two distinct eras of compute usage in training AI systems [McCandlish et al., 2018], [Amodei et al., 2019]



Microprocessor trends

[Brookes, 1986], [Sutter, 2005], [Rupp, 2015], [Rupp, 2018a], [Rupp, 2018b], [Hennessy et al., 2019]

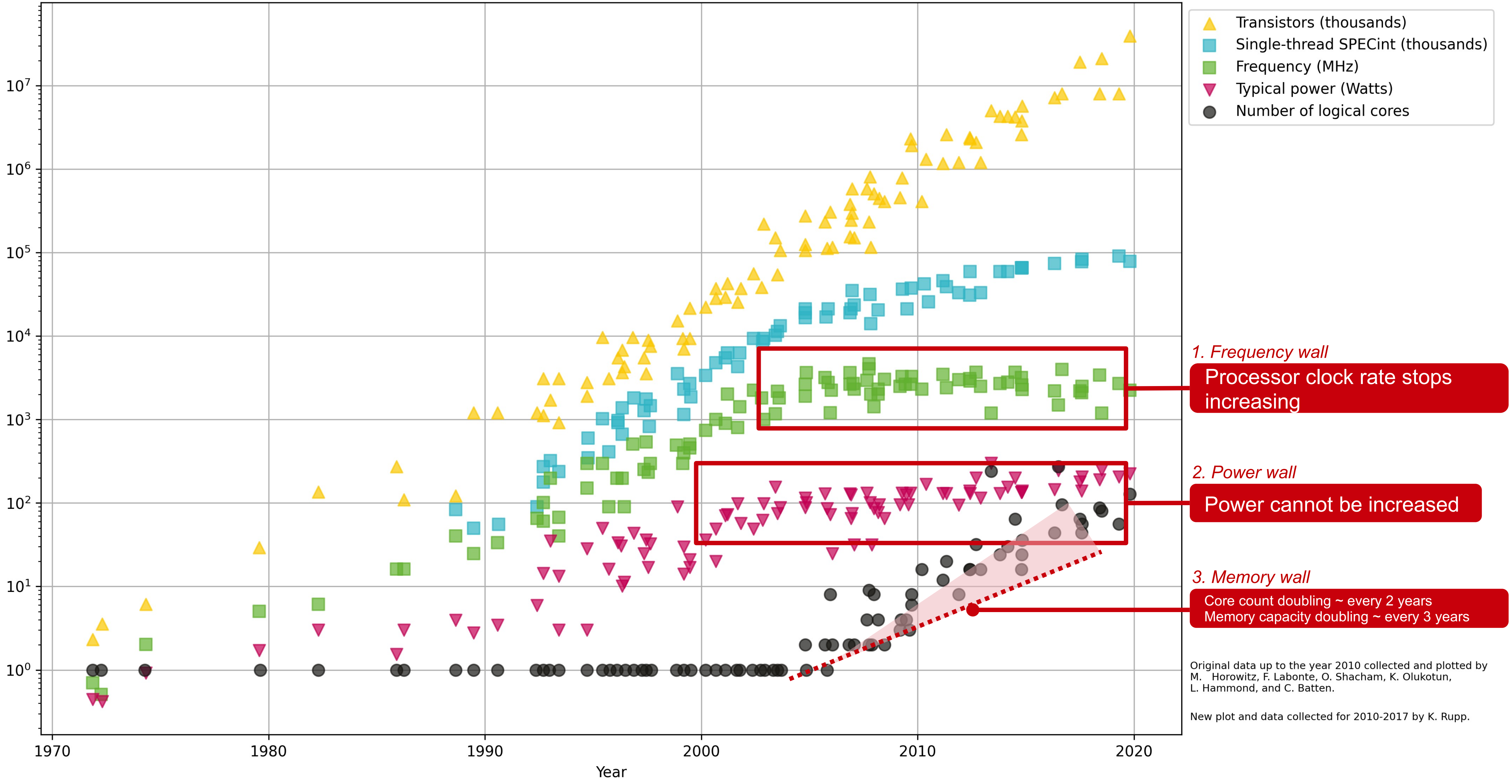


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten.

New plot and data collected for 2010-2017 by K. Rupp.

Microprocessor trends

[Brookes, 1986], [Sutter, 2005], [Rupp, 2015], [Rupp, 2018a], [Rupp, 2018b], [Hennessy et al., 2019]

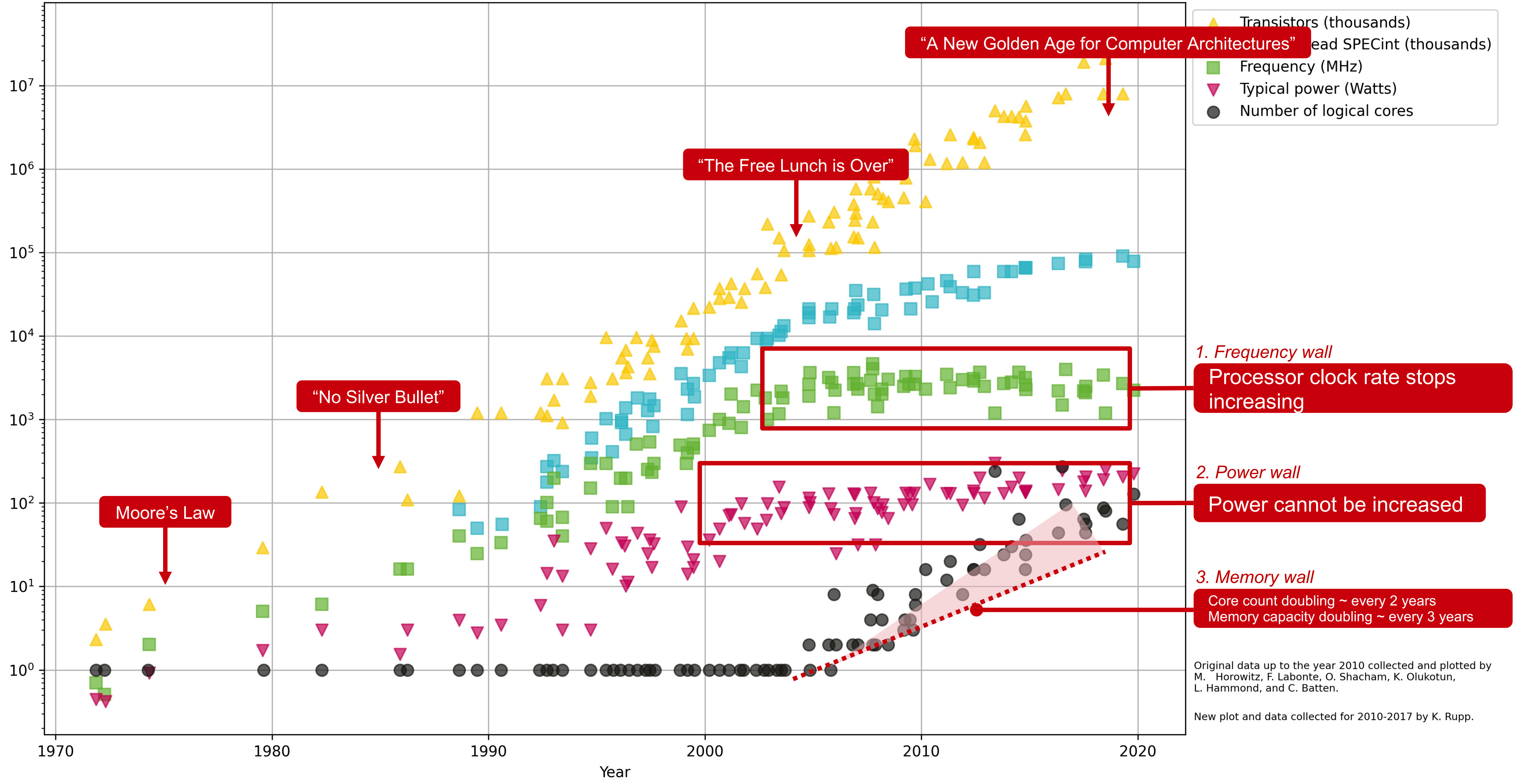


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten.

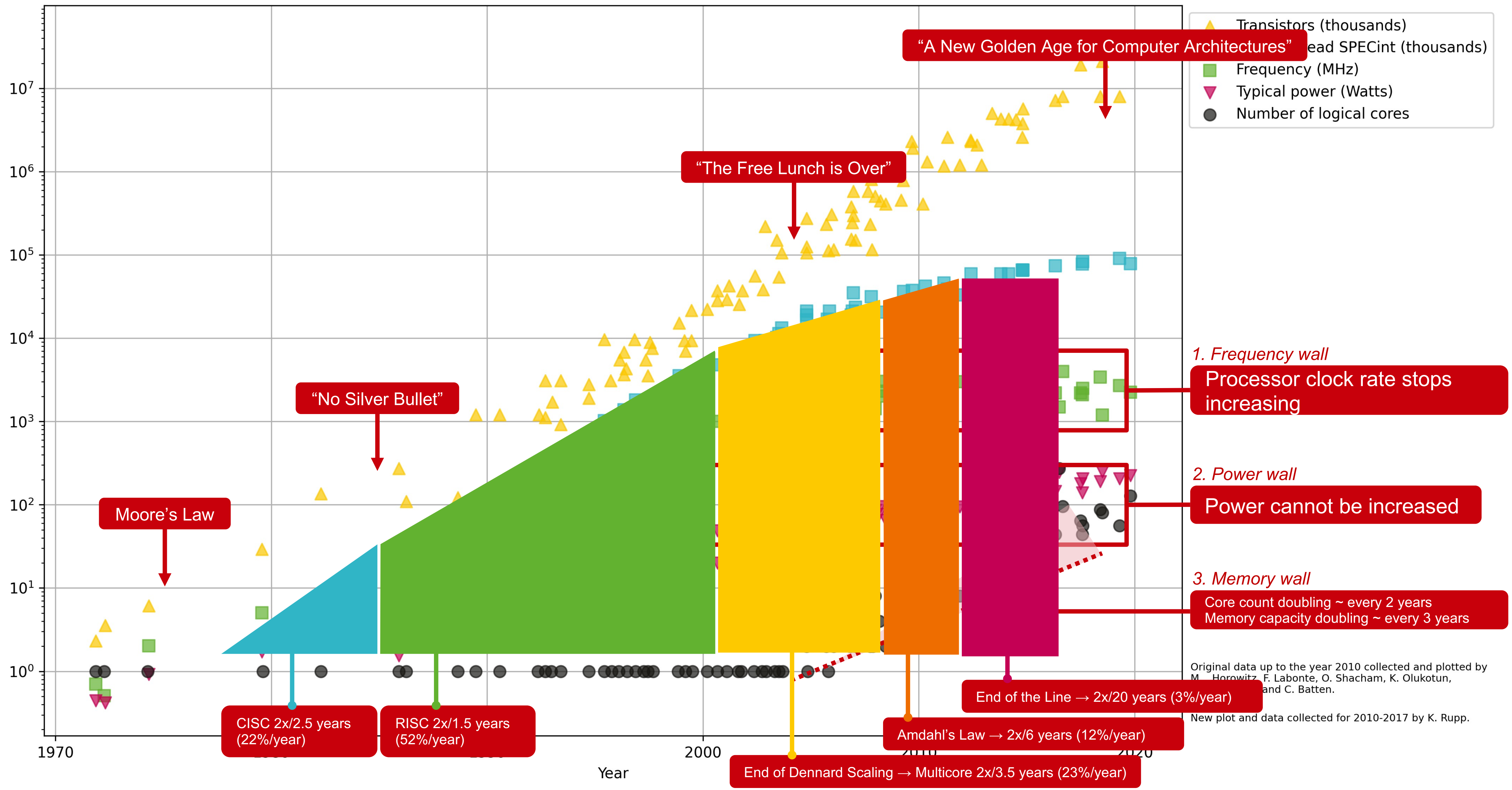
New plot and data collected for 2010-2017 by K. Rupp.

Microprocessor trends

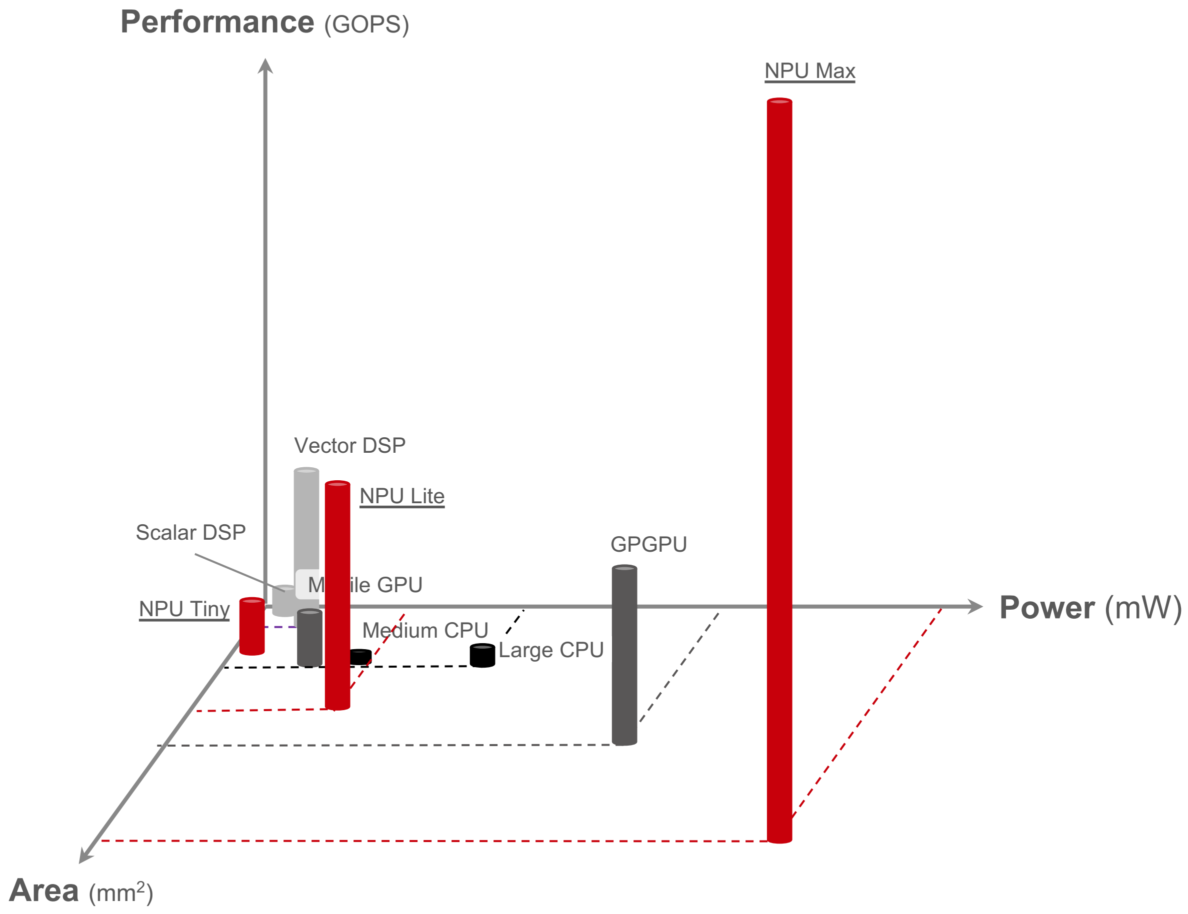
[Brookes, 1986], [Sutter, 2005], [Rupp, 2015], [Rupp, 2018a], [Rupp, 2018b], [Hennessy et al., 2019]



Microprocessor trends [Brookes, 1986], [Sutter, 2005], [Rupp, 2015], [Rupp, 2018a], [Rupp, 2018b], [Hennessy et al., 2019]



Rich variety of computing architectures

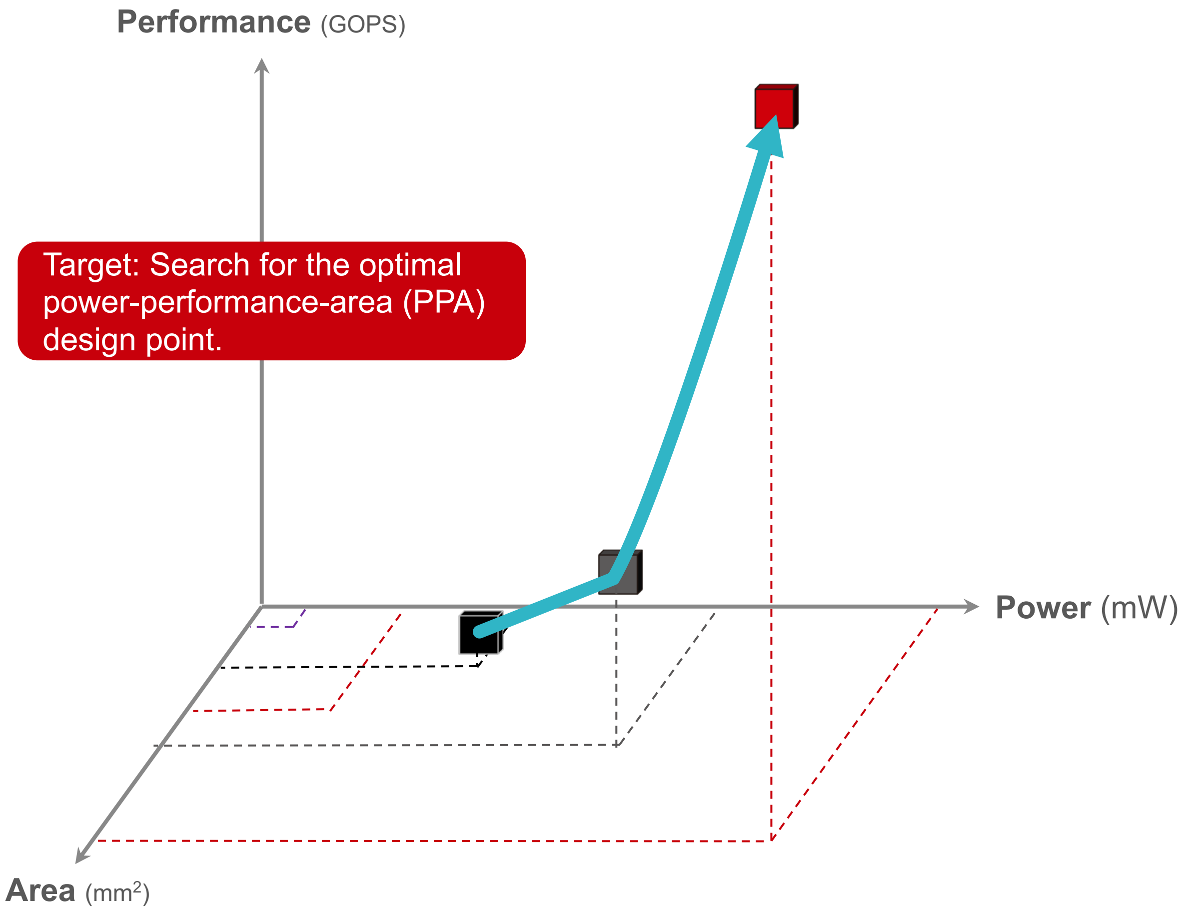


Note: Not exact proportion

- **Wide range of options to optimise for performance and efficiency:**
  - **Central processing unit (CPU)** executes general purpose applications (e.g. N-body methods, computational logic, map reduce, dynamic programming)
  - **General-purpose computing on graphics processing units (GPGPU)** accelerates compute intensive and time consuming applications for the CPU (e.g. dense linear algebra and sparse linear algebra)
  - **Digital signal processor (DSP)** accelerates signal processing for post camera operations (e.g. spectral methods)
  - **Image signal processor (ISP)** executes processing for camera sensor pipeline
  - **Vision processing unit (VPU)** accelerates machine vision tasks
  - **Network processor (NP)** accelerates packet processing
  - **Neural processing unit (NPU)** accelerates artificial intelligence applications (e.g. matrix-matrix multiplication, dot-products, scalar a times x plus y)

Each of these options represents different power, performance, and area trade-offs, which should be considered for specific application scenarios.

Rich variety of computing architectures



Note: Not exact proportion

- **Wide range of options to optimise for performance and efficiency:**
  - **Central processing unit (CPU)** executes general purpose applications (e.g. N-body methods, computational logic, map reduce, dynamic programming)
  - **General-purpose computing on graphics processing units (GPGPU)** accelerates compute intensive and time consuming applications for the CPU (e.g. dense linear algebra and sparse linear algebra)
  - **Digital signal processor (DSP)** accelerates signal processing for post camera operations (e.g. spectral methods)
  - **Image signal processor (ISP)** executes processing for camera sensor pipeline
  - **Vision processing unit (VPU)** accelerates machine vision tasks
  - **Network processor (NP)** accelerates packet processing
  - **Neural processing unit (NPU)** accelerates artificial intelligence applications (e.g. matrix-matrix multiplication, dot-products, scalar a times x plus y)

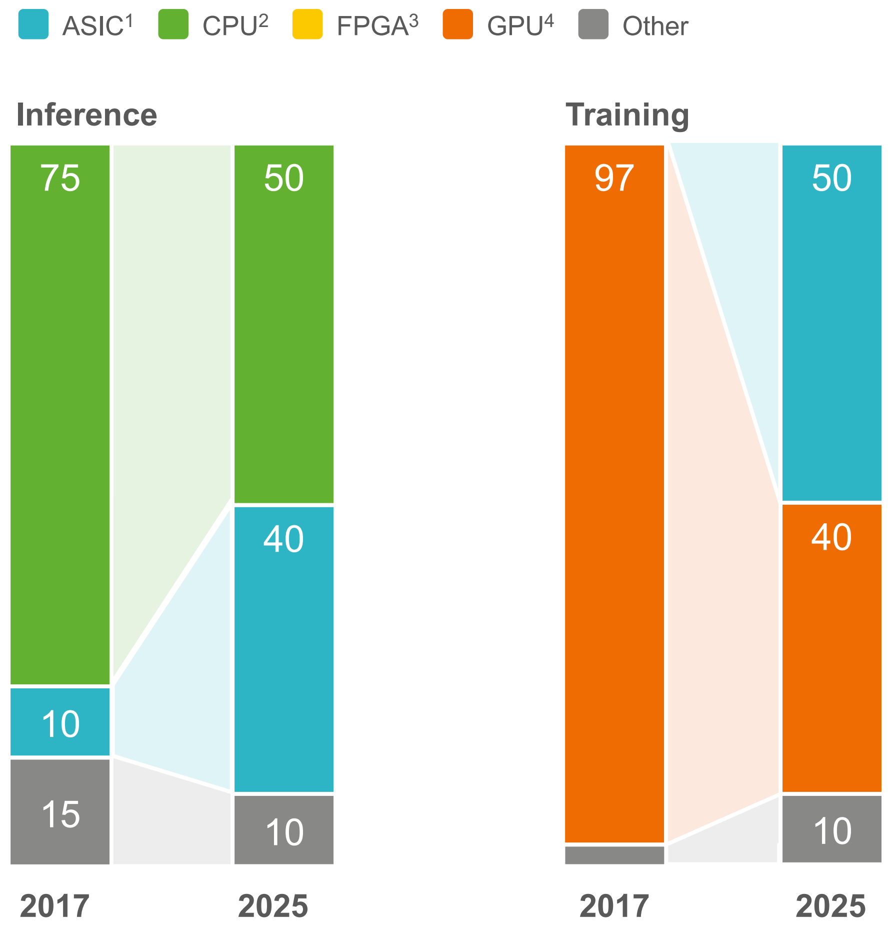
Each of these options represents different power, performance, and area trade-offs, which should be considered for specific application scenarios.

Comparison of processors for deep learning

	CPU	GPU	FPGA	ASIC
Proccessing peak power	Moderate	High	Very high	Highest
Power consumption	High	Very high	Very low	Low
Flexibility	Highest	Medium	Very high	Lowest
Training	Poor at training	Production-ready training hardware	Not efficient	Potential, best for training
Inference	Poor at inference but sometimes feasible	Average for inference	Good for inference	Potentially, best for inference focused
Improvement through	• Adding new instructions (e.g. AVX512, SVE) • Adding cores or increasing frequency (higher power consumption and cost)	• Adding new modules (e.g. support for multiple data types, tensor cores)		• Adding of special tensor cores for training and inference
Ecosystem	Rich ecosystem	Rich and mature ecosystem	Proprietary ecosystem (HDL programmable)	Allows leaveraging the existing ecosystem
Challenges	Computation and power efficiency is low	High cost, low energy efficiency ration and high latency	Long development period, high barrier to entry	High manufacturing cost High risks

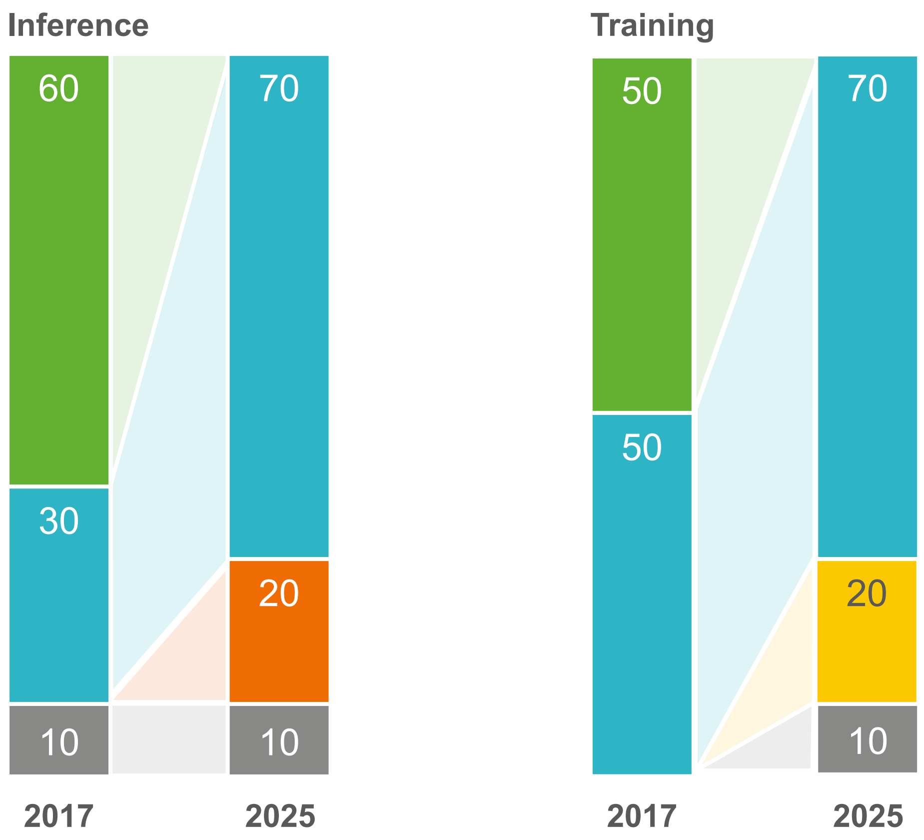
Preferred architectures for compute are shifting [Batra et al., 2018]

Data-centre architecture (%)



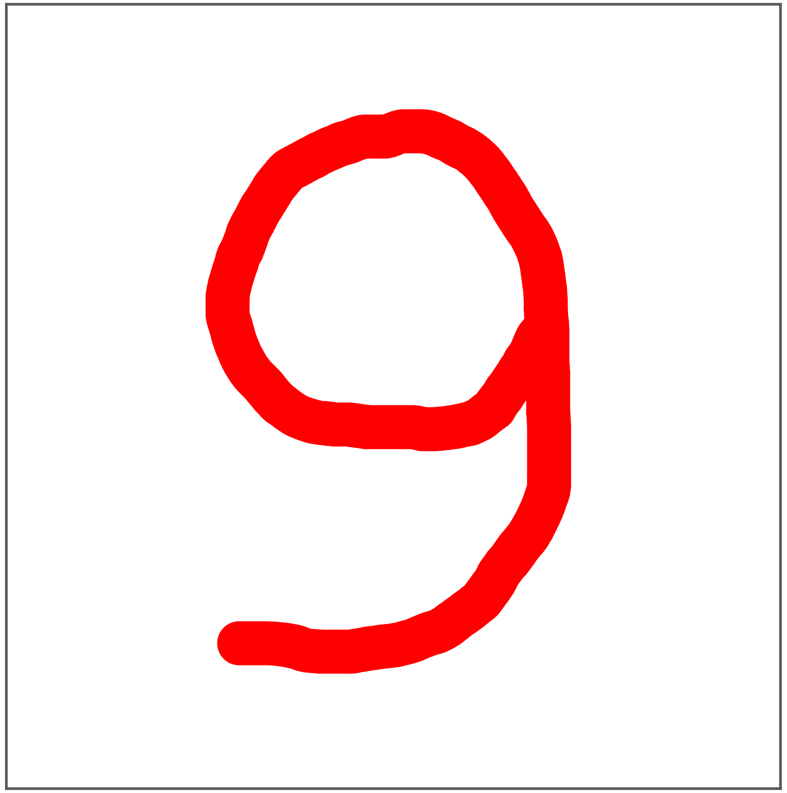
Edge architecture (%)

[1] Application-specific integrted circuit
[2] Central processing unit
[3] Field programmable gate array
[4] Graphics processing unit



Convolutional Neural Networks

Data structure of digital images



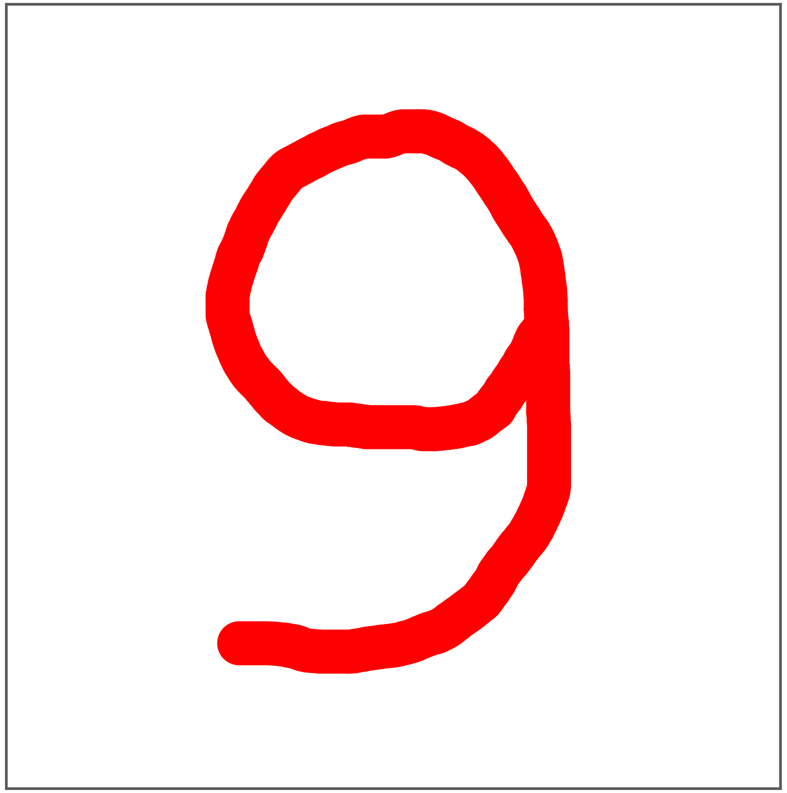
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	0	0	255	0	0
0	0	255	0	0	255	0	0
0	0	0	255	255	255	0	0
0	0	0	0	0	255	0	0
0	0	255	255	255	0	0	0
0	0	0	0	0	0	0	0

Blue

Green

Red

Data structure of digital images



Blue

Green

Red

0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	0	0	255	0	0
0	0	255	0	0	255	0	0
0	0	0	255	255	255	0	0
0	0	0	0	0	255	0	0
0	0	255	255	255	0	0	0
0	0	0	0	0	0	0	0

Kernel convolution example /1

Input image

10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$*$

Kernel

1	2	1
0	0	0
-1	-2	-1

$=$

Feature map

0	0	0	0
40	40	40	40
40	40	40	40
0	0	0	0

Subsequent feature map values are calculated according to the following formula, where the input image is denoted by f and our kernel by h. The indexes of rows and columns of the feature map (result matrix) are marked with m and n respectively.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

Kernel convolution example /2

Input image

10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

$*$

Kernel

1	2	1
0	0	0
-1	-2	-1

$=$

Feature map

0	0	0	0
40	40	40	40
40	40	40	40
0	0	0	0

Subsequent feature map values are calculated according to the following formula, where the input image is denoted by f and our kernel by h. The indexes of rows and columns of the feature map (result matrix) are marked with m and n respectively.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

$$G[1,1] = 10 * 1 + 10 * 2 + 10 * 1 + 10 * 0 + 10 * 0 + 10 * 0 + 0 * (-1) + 0 * (-2) + 0 * (-1) = 40$$

Kernel convolution example /3

Input image

Carl Friedrich Gauß (1777–1855)

*

Kernel

-1	-1	-1
-1	8	-1
-1	-1	-1

=

Feature map

An **outline** kernel (also called an "edge" kernel) is used to highlight large differences in pixel values. A pixel next to neighbour pixels with close to the same intensity will appear black in the new image while one next to neighbour pixels that differ strongly will appear white.

Please have a look at Gimp's documentation on general filters using convolution matrices. You can also apply your custom filters in Photoshop by going to **Filter** ▸ **Other** ▸ **Custom**.

Kernel convolution example /4

Input image

Carl Friedrich Gauß (1777–1855)

*

Kernel

-1	-2	-1
0	0	0
1	2	1

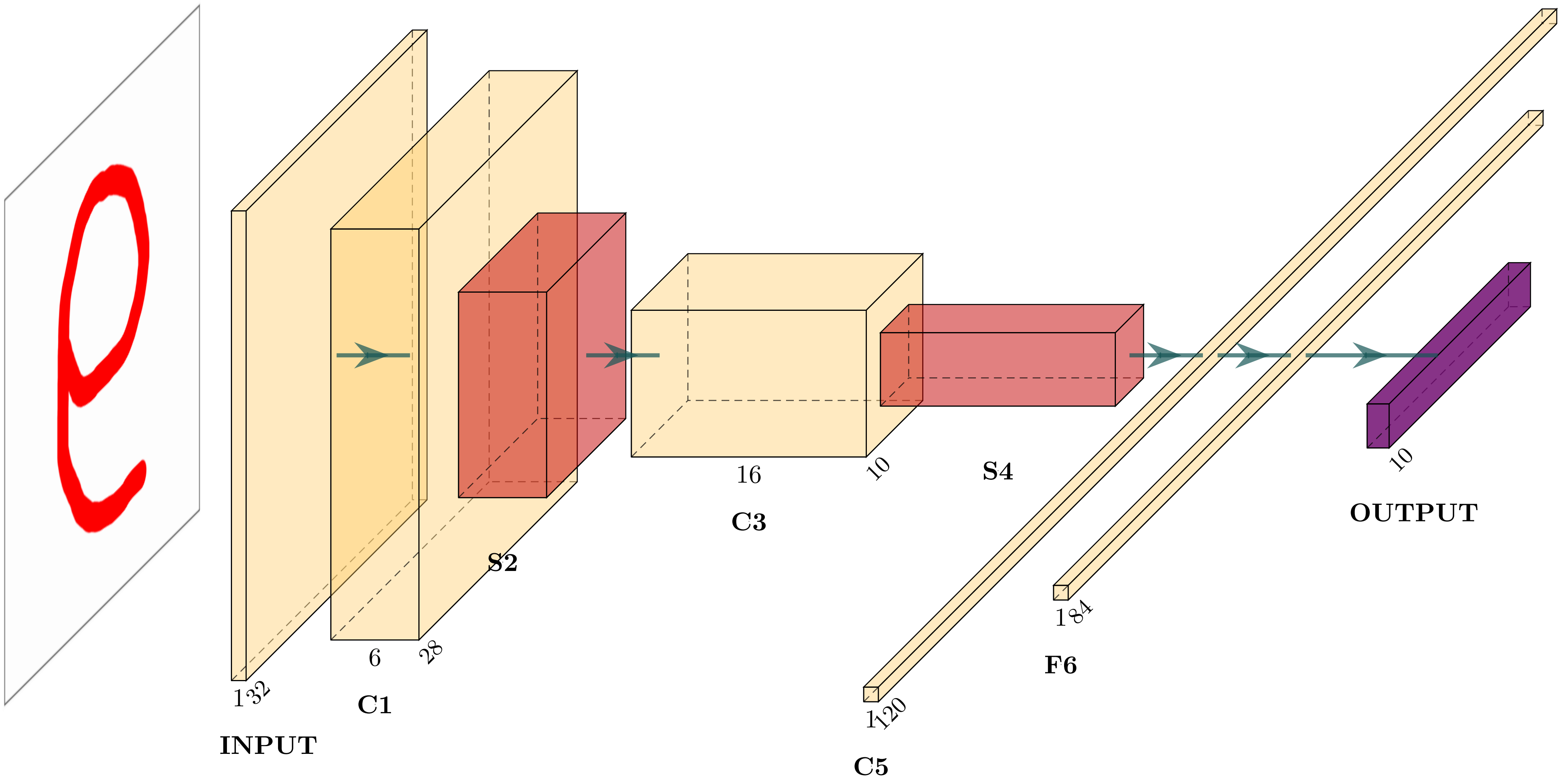
=

Feature map

The **sobel** kernels are used to show only the differences in adjacent pixel values in a particular direction.

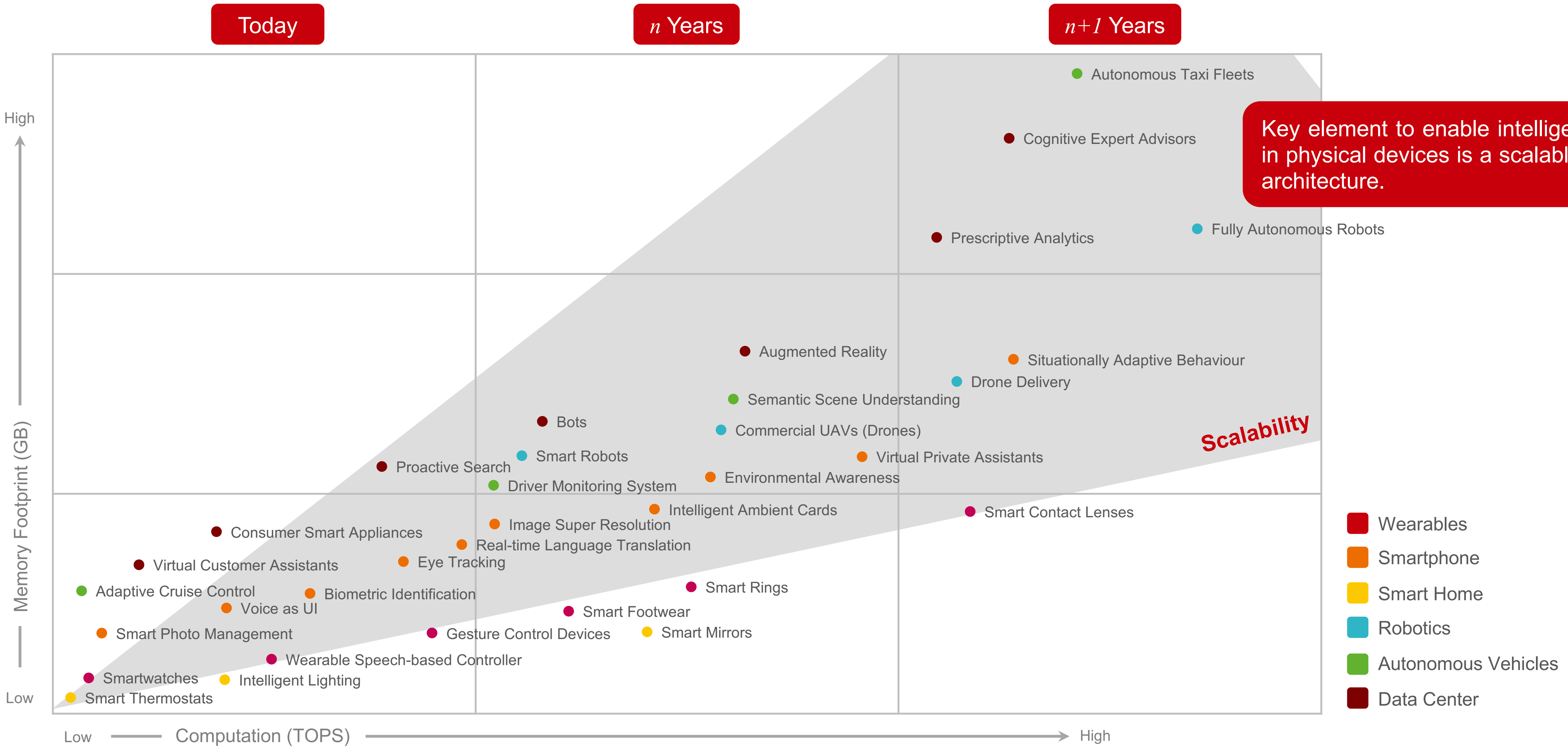
Please have a look at Gimp's documentation on general filters using convolution matrices. You can also apply your custom filters in Photoshop by going to **Filter** ▸ **Other** ▸ **Custom**.

Architecture of LeNet-5



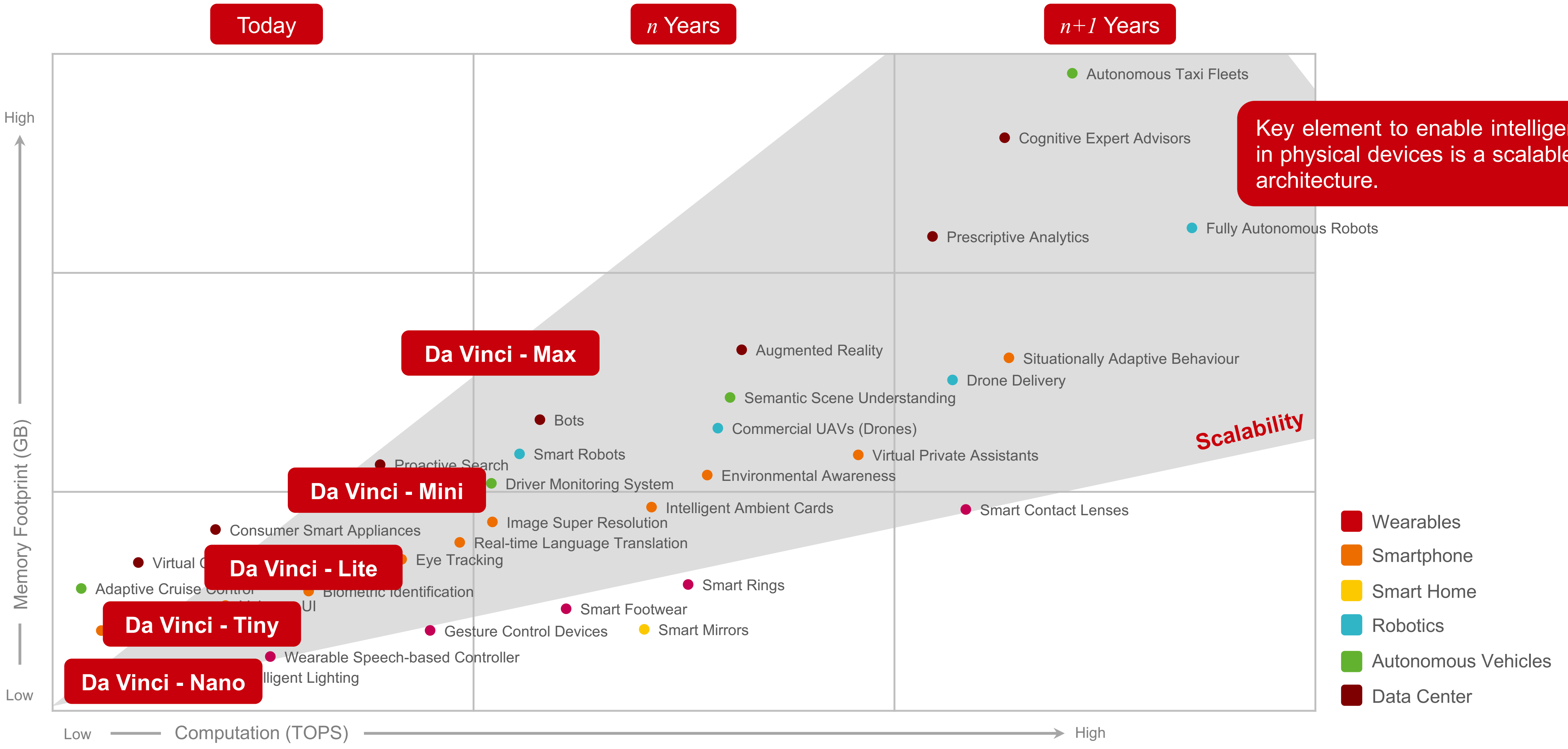
Applicability of artificial intelligence

Ubiquitous and future AI computation requirements



Key element to enable intelligence in physical devices is a scalable AI architecture.

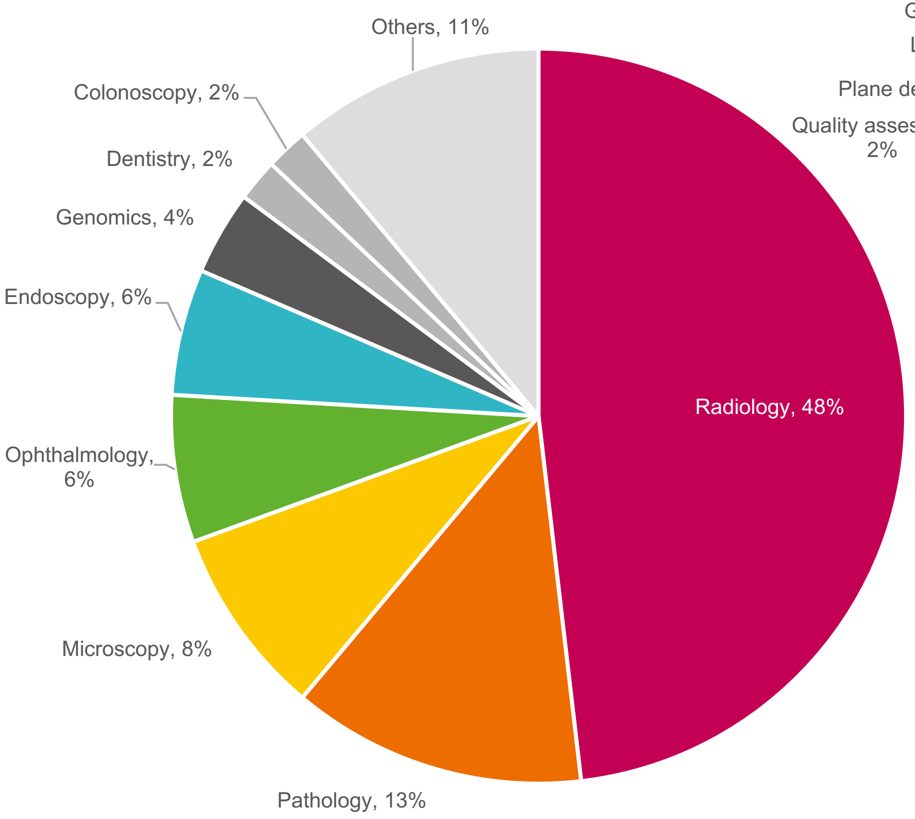
Ubiquitous and future AI computation requirements



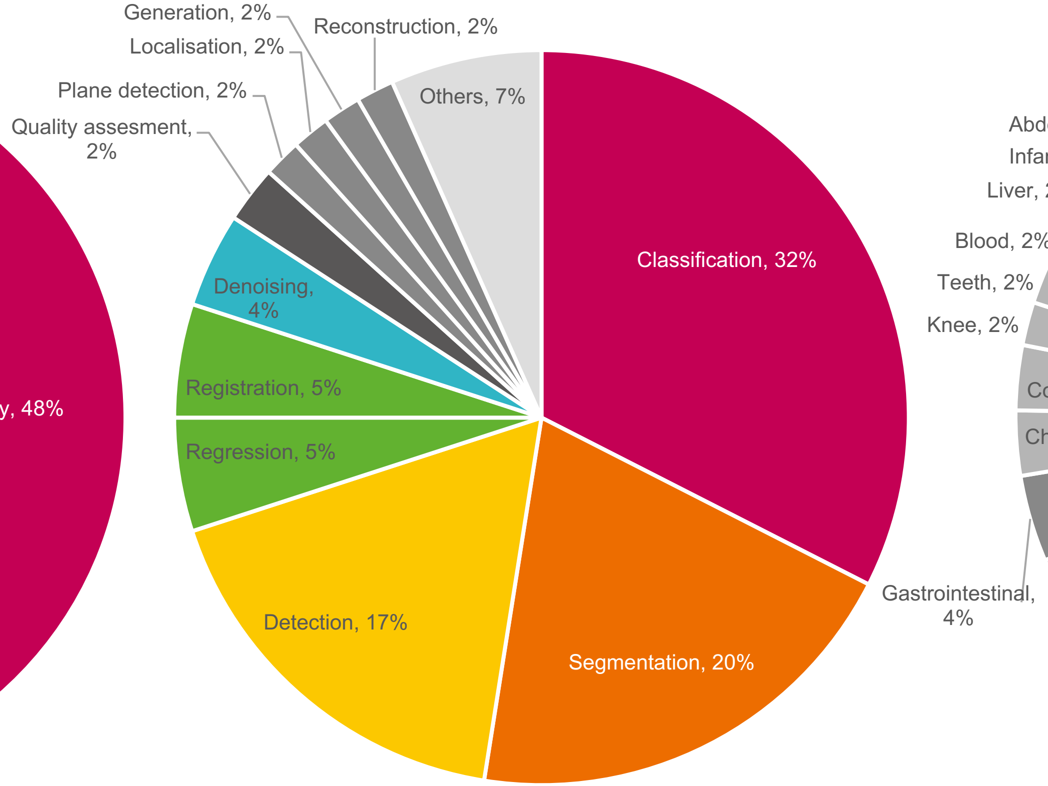
Key element to enable intelligence in physical devices is a scalable AI architecture.

Artificial intelligence in modern medicine [Medical AI Index, 2020]

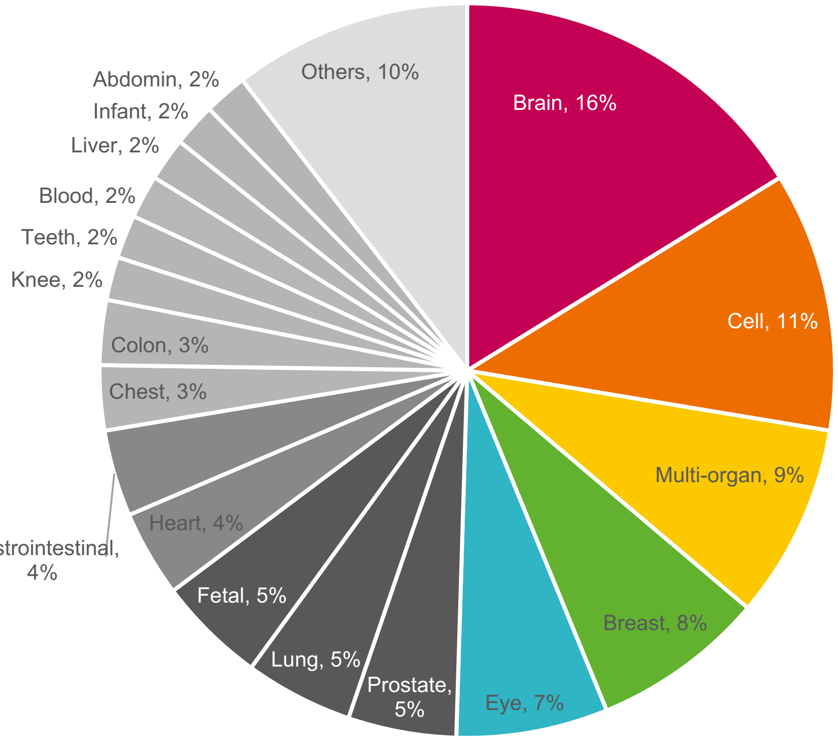
Medical domains



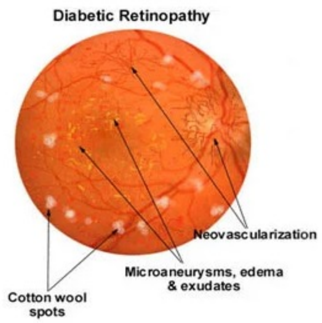
Medical tasks



Anatomy



Artificial intelligence in modern medicine



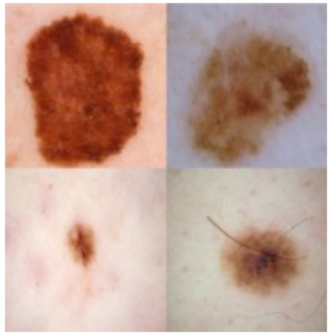
Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs

[Gulshan et al., 2016]

2016

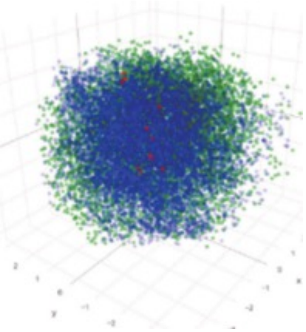
Automated 5-year mortality prediction using deep learning and radiomics features from chest computed tomography

[Carneiro et al., 2017]



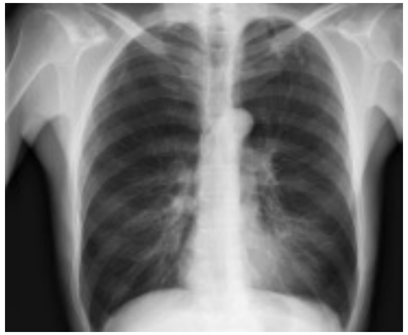
Dermatologist-level classification of skin cancer with deep neural networks

[Esteva et al., 2017]



Applying deep adversarial autoencoders for new molecule development in oncology

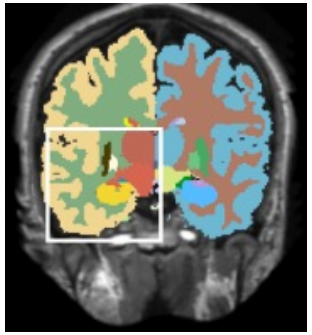
[Kadurin et al., 2017]



Radiologist-level pneumonia detection on chest X-rays with deep learning

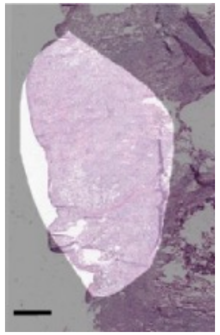
[Rajprkar et al., 2017]

2017



A fully convolutional network for quick and accurate segmentation of neuroanatomy

[Roy et al., 2018]



Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning

[Coudray et al., 2018]

2018

Artificial intelligence in modern medicine

Deep learning predicts hip fracture using confounding patient and healthcare variables

[Badgeley et al., 2018]

2018

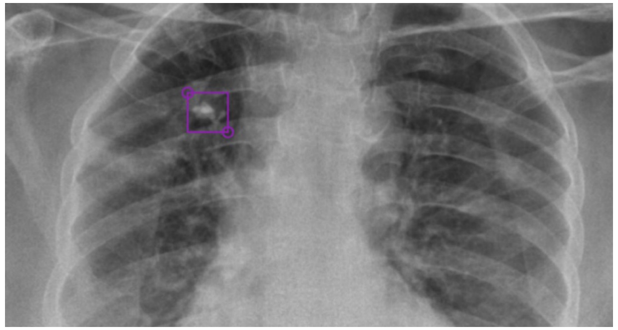
AI-driven version of the Berg Balance Scale

[Wolff, 2019]

Real-time automatic detection system increases colonoscopic polyp and adenoma detection rates: a prospective randomised controlled study

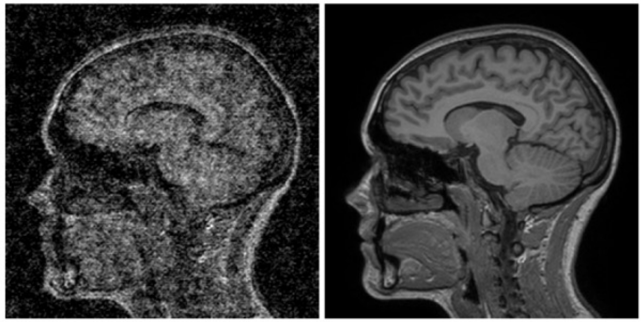
[Wang et al., 2019]

2019



DeepTek detects tuberculosis from X-rays

[Salian, 2020]

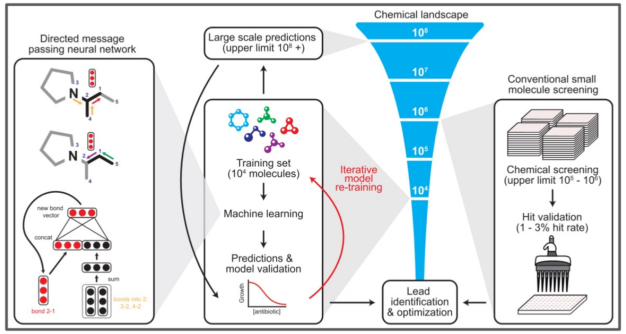


FDA approve the first MRI super-resolution product

[Oakden-Rayner, 2020], [Subtle Medical Inc., 2020]

2020

Artificial intelligence in modern medicine



A deep learning approach to antibiotic discovery

To be continued …

[Stokes et al., 2020]]

2020

Upscaling and colourisation of video footage /1

Source: https://www.reddit.com/r/interestingasfuck/comments/idbtrg/i_upscaled_and_colorized_the_footage_about_the/

Despina Manaki is the earliest-born person on film. In 1905, when she was 114 years old (born 1791), she was filmed by her grandsons, Yanaki and Milton Manaki, cinema pioneers in the Balkans and the Ottoman Empire. This video shows the full force of artificial intelligence restoring old footages.

Upscaling and colourisation of video footage /2

Following his historic spaceflight, Yuri Gagarin, who was a former foundry worker, was invited to visit Manchester by the Foundry Worker's Union. The Macmillan Government extended the invitation to London, adding an extra few days to the planned tour in July 1961. Major Yuri Gagarin answers questions put to him by Richard Dimbleby, Tom Margerison, science editor of the Sunday Times, and Yuri Fokin of the Soviet Television Service during a live broadcast from the Russian exhibition at Earls Court in 1961. Boris Belitsky of Moscow Radio interprets his answers.

Original

Source: https://www.bbc.co.uk/programmes/p00fwcbn

60 fps, upscale, deflickering, denoise, colourisation and facial restoration

Source: https://www.youtube.com/watch?v=4DN5WjU8km0&list=UUD8J_xbbBuGobmw_N5ga3MA

Product realisation

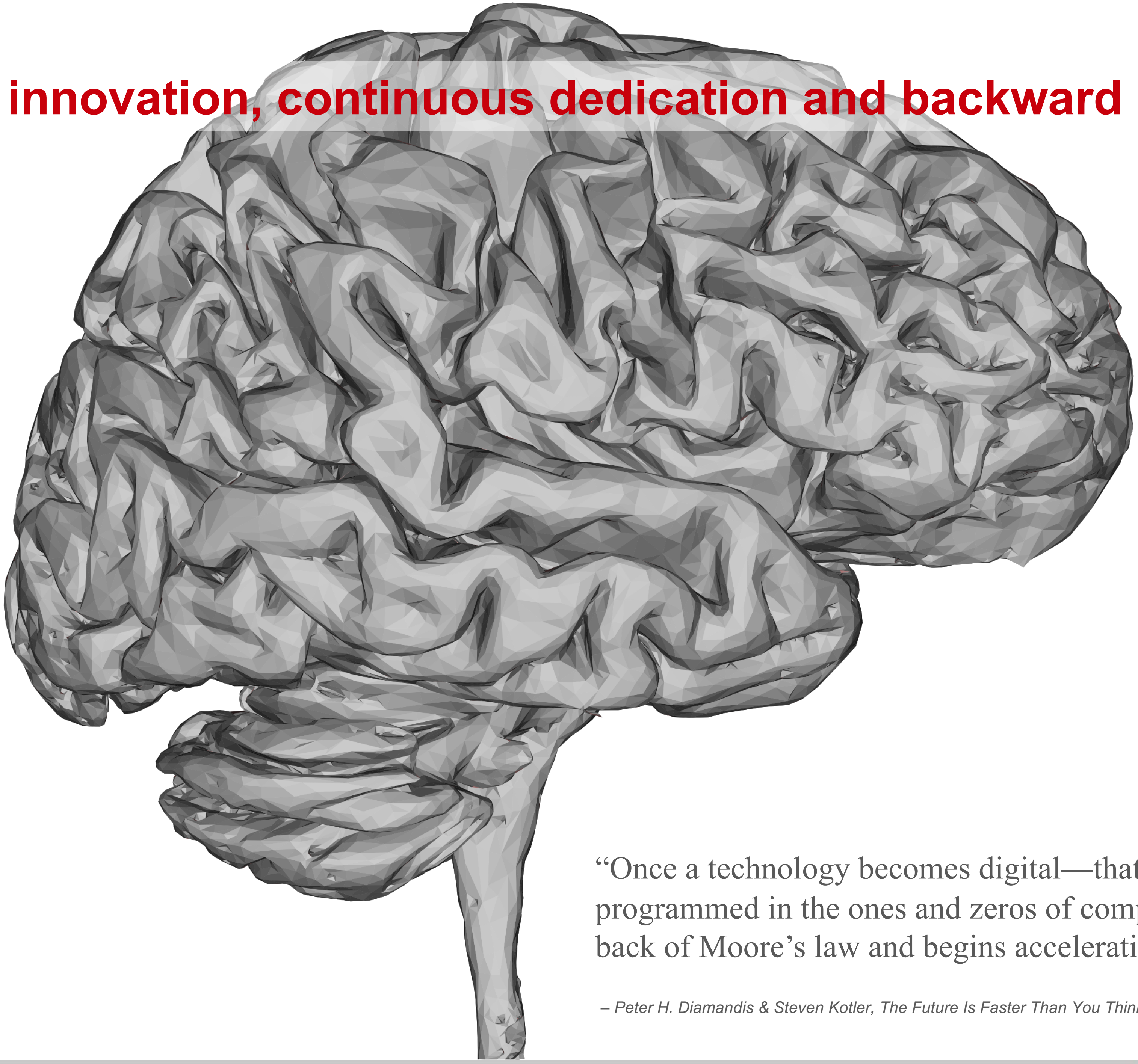
Scalable across devices






	Device				Edge		Cloud
	Earphone	**Always-on**	**Smartphone**	**Laptop**	**IPC**	**Edge Server**	**Data Centre**
Compute	20 MOPS	100 GOPS	1 - 10 TOPS	10 - 20 TOPS	10 - 20 TOPS	10 - 100 TOPS	200+ TOPS
Power budget	1 mW	10 mW	1 - 2 W	3 - 10 W	3 - 10 W	10 - 100 W	200+ W
Model size	10 KB	100 KB	10 MB	10 - 100 MB	10 -100 MB	100+ MB	300+ MB
Latency	< 10 ms	~10 ms	10 - 100 ms	10 - 500 ms	10 - 500 ms	ms ~ s	ms ~ s
Inference?	Y	Y	Y	Y	Y	Y	Y
Training?	N	N	Y	Y	Y	Y	Y
SoC scale	Nano	Tiny	Lite	Mini Ascend 310	Mini Ascend 310	Multi-Mini Ascend 310	Max Ascend 910

Focus on innovation, continuous dedication and backward compatibility



"Once a technology becomes digital—that is, once it can be programmed in the ones and zeros of computer code—it hops on the back of Moore's law and begins accelerating exponentially."

– Peter H. Diamandis & Steven Kotler, The Future Is Faster Than You Think

Focus on innovation, continuous dedication and backward compatibility

Inference | **Training**

Ascend 310

AI SoC with ultimate efficiency

- Half precision (FP16): 8 TFLOPS
- Integer precision (INT8): 16 TOPS
- 16-channel full-HD video decoder: H.264/265
- 1-channel full-HD video encoder: H.264/265
- Max. power consumption: 8 W
- 12nm

Ascend 910

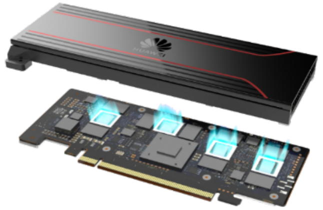
Highest compute density on a single chip

- Half precision (FP16): 256 TFLOPS
- Integer precision (INT8): 512 TOPS
- 128-channel full-HD video decoder: H.264/265
- Max. power consumption: 350 W
- 7nm

"Once a technology becomes digital—that is, once it can be programmed in the ones and zeros of computer code—it hops on the back of Moore's law and begins accelerating exponentially."

– Peter H. Diamandis & Steven Kotler, The Future Is Faster Than You Think

Focus on innovation, continuous dedication and backward compatibility



Atlas 300

AI Accelerator Card

- 64 TOPS of INT8 @ 67 W
- 32 GB memory
- 64-channel HD video real-time analytics
- Standard half-height half-length PCIe card form factor, applicable to general-purpose servers

reddot award 2019 winner

Atlas 200

AI Accelerator Module

- 16 TOPS of INT8
- 16-channel HD video real-time analytics, JPEG decoding
- 4 GB/8 GB memory, PCIe 3.0 x4 interface
- Operating temperature: -25°C to +80°C

52 mm x 38 mm x 10 mm

Atlas 500

AI Edge Stations

- 16 TOPS of INT8
- 25–40 W
- Wi-Fi & LTE
- 16-channel HD video real-time analytics
- Fanless design, -40°C to +70°C environments

reddot award 2019 winner

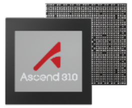
Atlas 800

Deep Learning System

- Plug-and-play installation
- Ultimate Performance
- Integrated Management

Ascend 310

AI SoC with ultimate efficiency



- Half precision (FP16): 8 TFLOPS
- Integer precision (INT8): 16 TOPS
- 16-channel full-HD video decoder: H.264/265
- 1-channel full-HD video encoder: H.264/265
- Max. power consumption: 8 W
- 12nm

Ascend 910

Highest compute density on a single chip

- Half precision (FP16): 256 TFLOPS
- Integer precision (INT8): 512 TOPS
- 128-channel full-HD video decoder: H.264/265
- Max. power consumption: 350 W
- 7nm

Atlas 900 AI Cluster

The pinnacle of computing power

- Thousands of Ascend 910 AI processors
- High-speed interconnection
- Delivers up to 256 to 1024 PetaFLOPS at FP16
- Can complete model training based on ResNet-50 within 59.8 seconds
- 15% faster than the second-ranking product
- Faster AI model training with images and speech

Atlas 200 DK

Quickly build development environments in 30 minutes

- 16 TOPS of INT8 @ 24 W
- 1 USB type-C, 2 camera interfaces, 1 GE port, 1 SD card slot
- 4 GB/8 GB memory

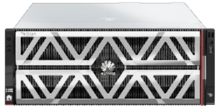
Kunpeng 920

The industry's highest-performance ARM-based server CPU

- ARM v8.2-architecture
- up to 64 cores, 2.6 GHz
- 8 DDR4 memory channels
- PCIe 4.0 and CCIX
- Integrated 100GE LOM and encryption and compression engines
- Supports 2- or 4-socket interconnects

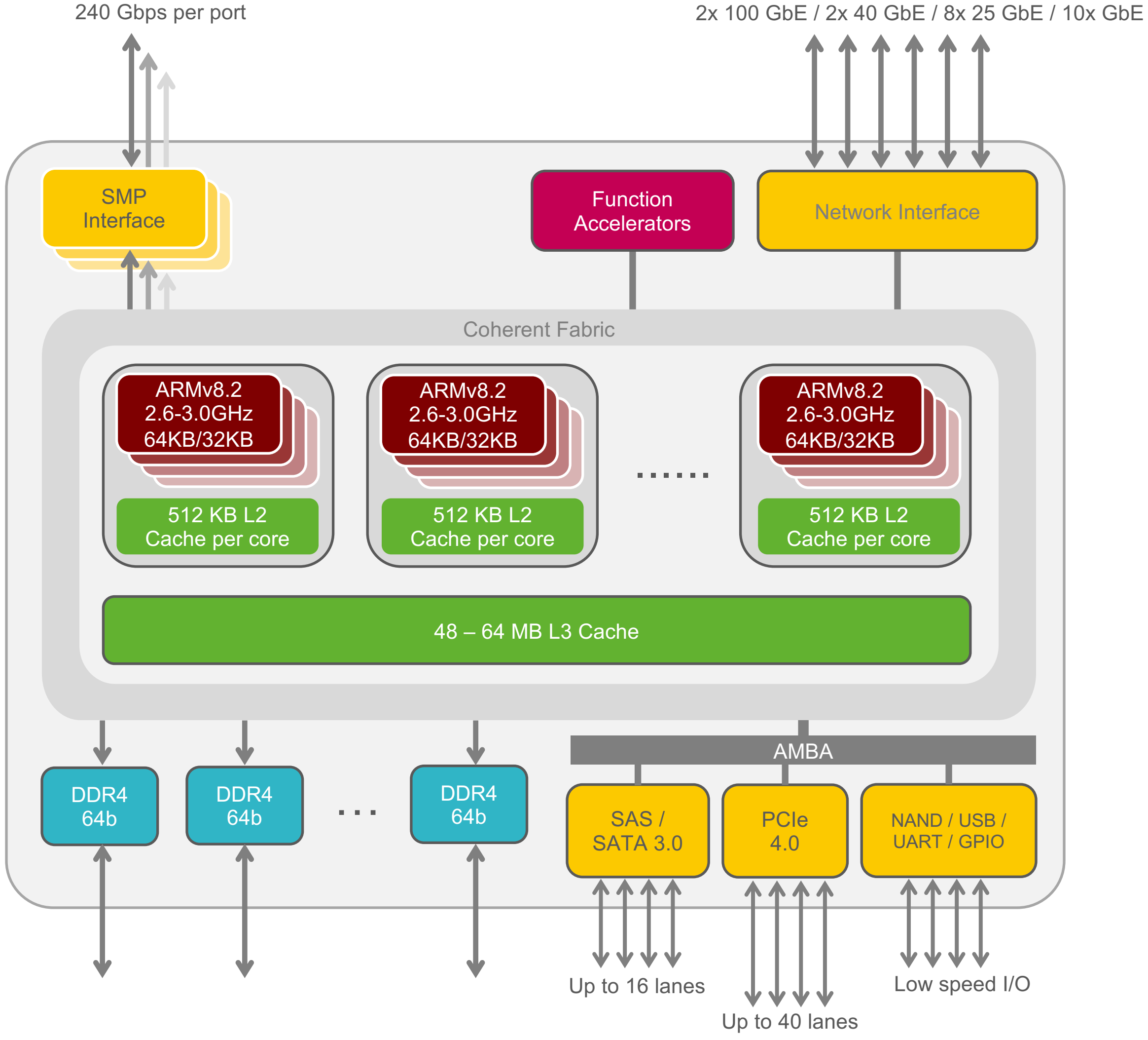
Storage-intensive — **Computing-intensive**

5290	5280	2280	1280	2480	X6000
4U 72-drive storage model	4U 40-drive storage model	2U 2S balanced model	1U 2S high-density model	2U 4S high-performance model	2U 4-node high-density model

HiSilicon Kunpeng 920

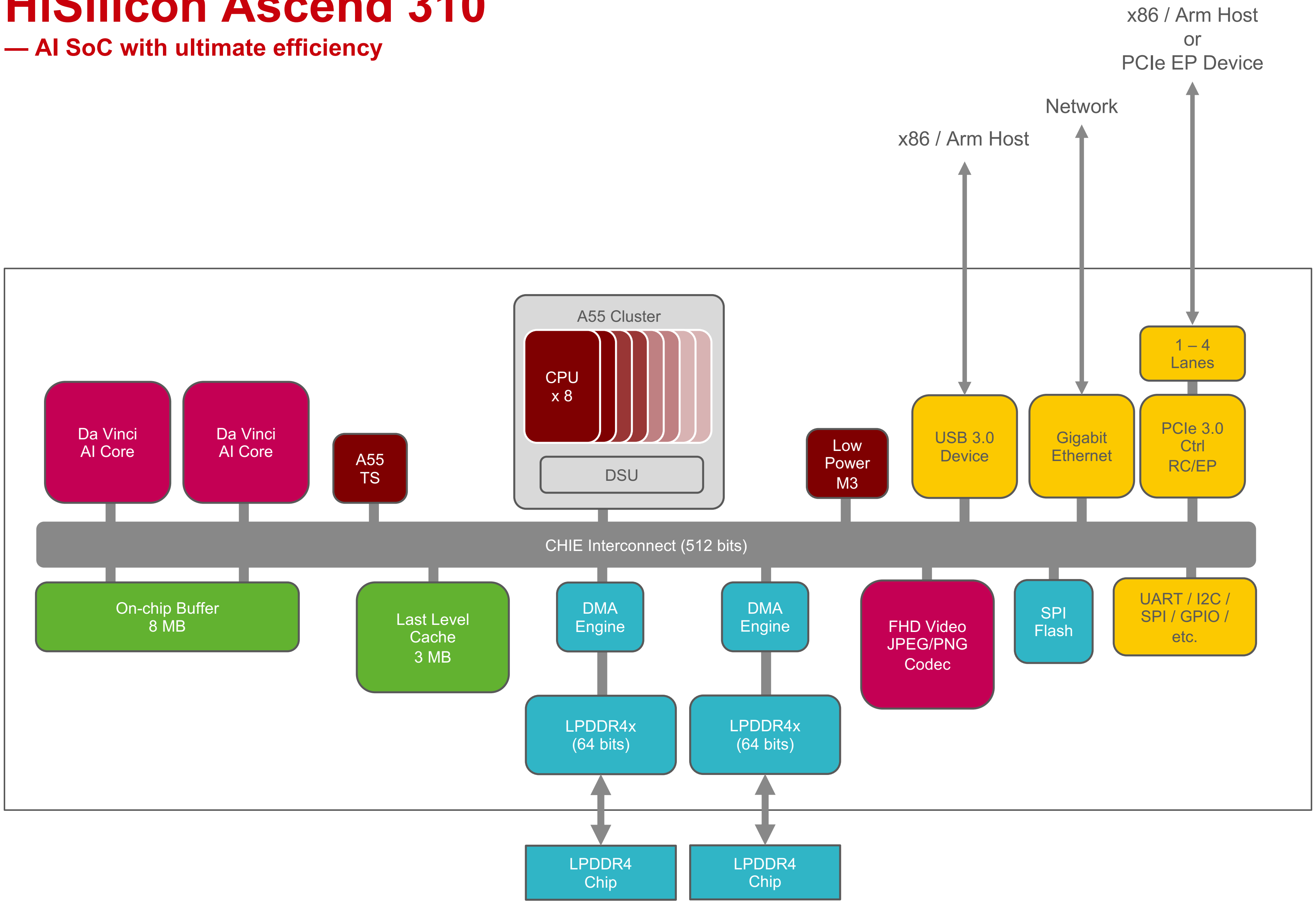
— The industry's highest-performance ARM-based server CPU

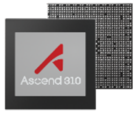


CPU	• 48x Cores, ARMv8.2, 3.0 GHz, 48-bit physical address space • 4x Issue out-of-order superscalar design • 64 KB L1 instruction cache and 64 KB L1 data cache
L2 Cache	• 512 KB private per core
L3 Cache	• 48 MB shared for all (1 MB/core)
Memory	• 8-channel DDR4-2400/2666/2933/3200
PCIe	• 40x PCI Express 4.0 lanes
Integrated I/O	• 8x Ethernet lanes, combo MACs, supporting 2x 100GbE, 2x 40GbE, 8x 25GbE/10GbE,10xGbE • RoCEv1 and RoCEv2 • x4 USB ports • 16x SAS 3.0 lanes and 2x SATA 3.0 lanes
CCIX	• Cache coherency interface for Xilinx FPGA accelerator • (collaboration with Xilinx)
Management Engine	• Isolated management subsystem (co-works with ARM's SCP & MCP firmware)
Scale-up	• Coherent SMP interface for 2P/4P configurations • Up to 240 Gbit/s per port
Power	• 180 Watt (64x cores with 2.6 GHz) • 150 Watt (48x cores with 2.6 GHz)

HiSilicon Ascend 310

— AI SoC with ultimate efficiency



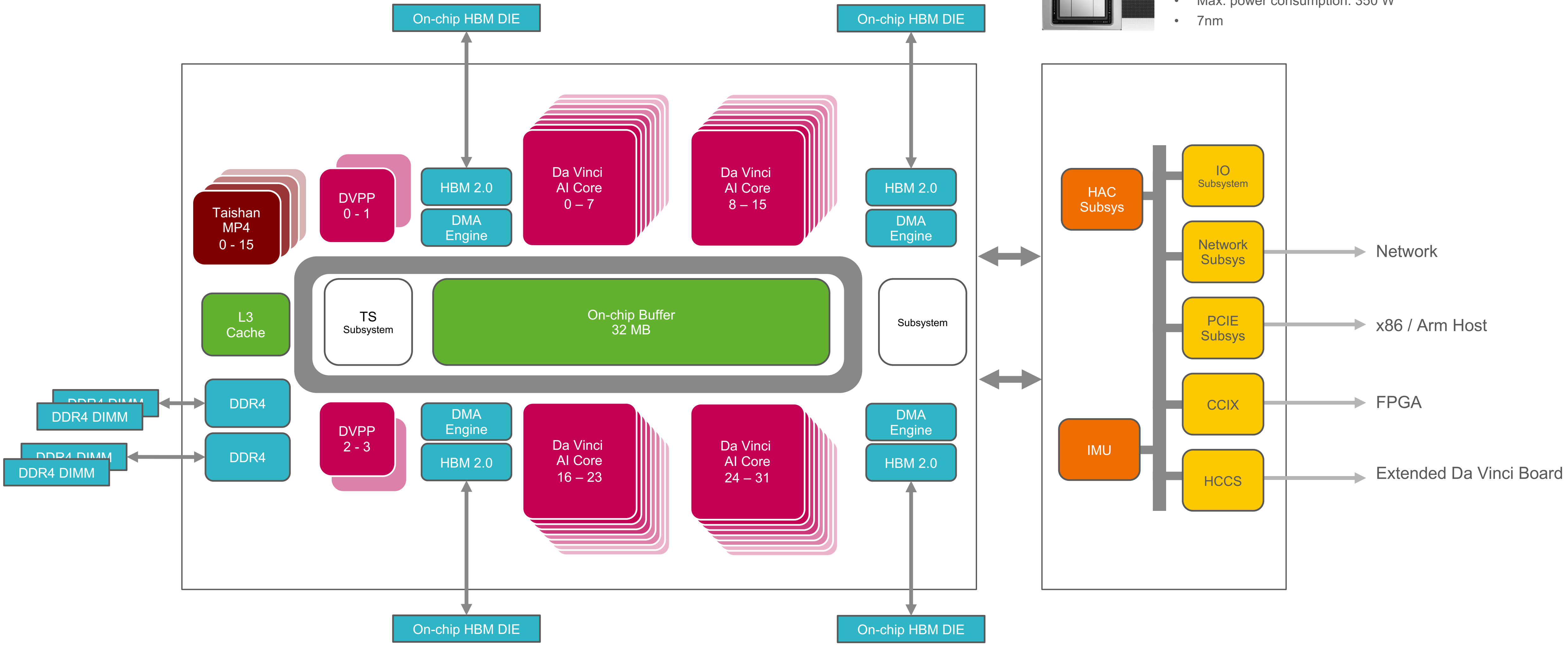


- Half precision (FP16): 8 TFLOPS
- Integer precision (INT8): 16 TOPS
- 16-channel full-HD video decoder: H.264/265
- 1-channel full-HD video encoder: H.264/265
- Max. power consumption: 8 W
- 12nm

HiSilicon Ascend 910

— Highest compute density on a single chip

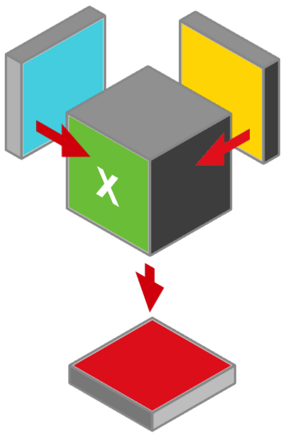
- Half precision (FP16): 256 TFLOPS
- Integer precision (INT8): 512 TOPS
- 128-channel full-HD video decoder: H.264/265
- Max. power consumption: 350 W
- 7nm



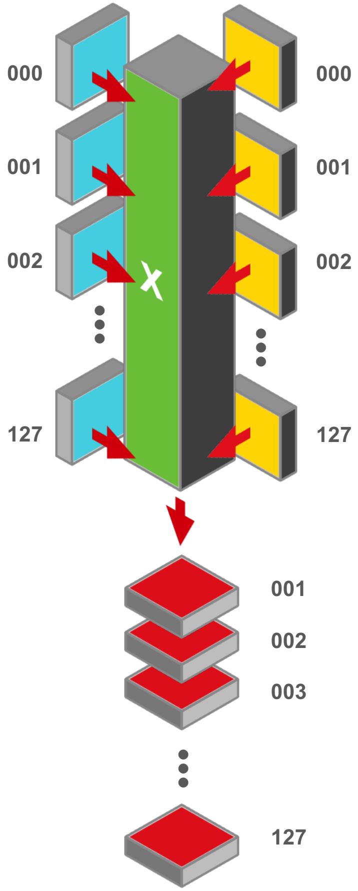
Da Vinci Architecture

Building blocks and compute intensity

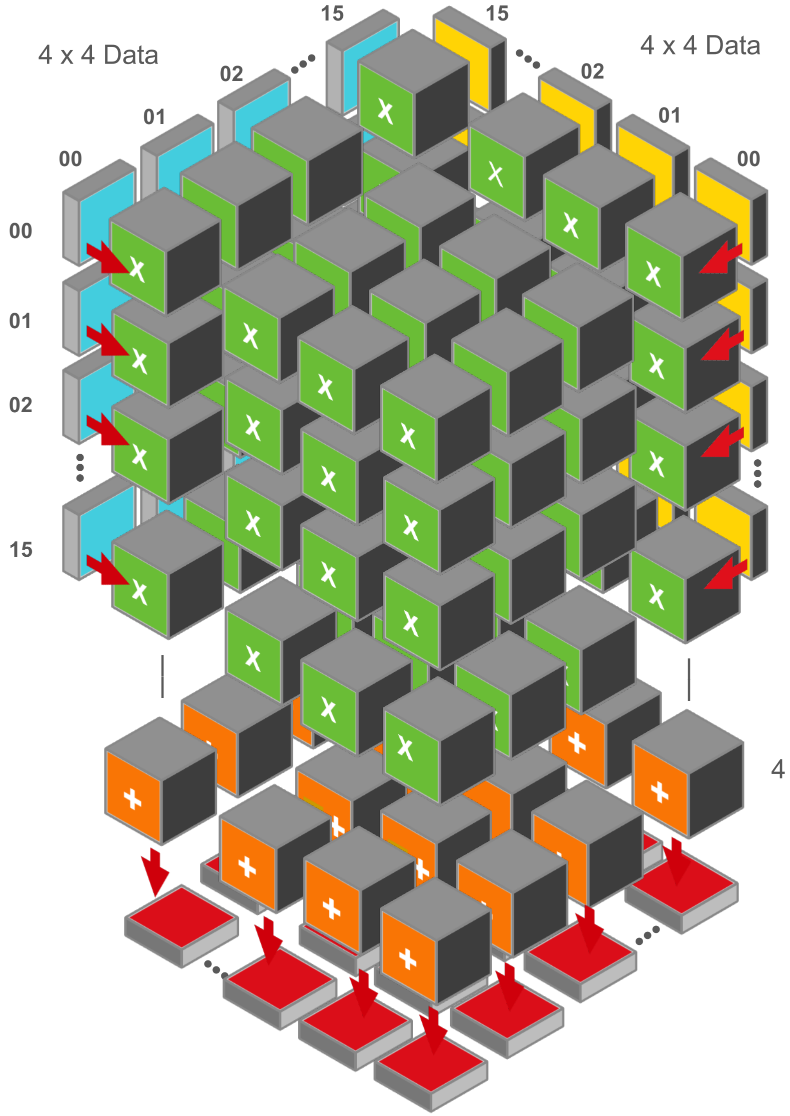
Scalar Unit
Full flexibility in computation



Vector Unit
Rich and efficient operations



Cube Unit
High intensity computation



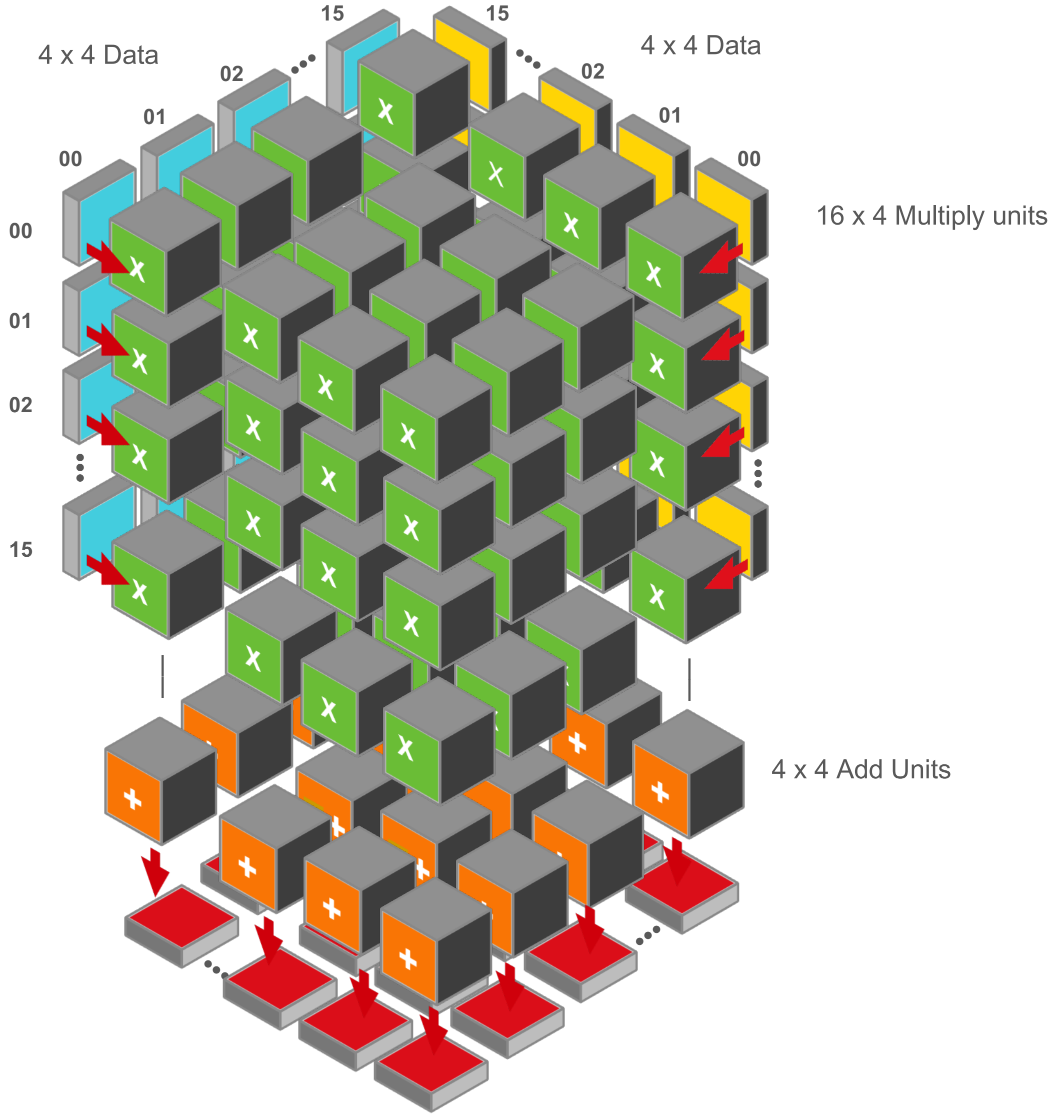
Building blocks and compute intensity

Number of multiply-accumulators (MACs)

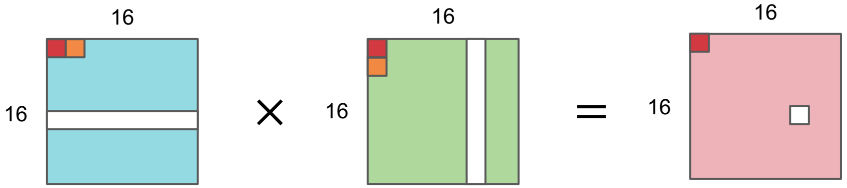
N	N^2	N^3
1	1	1
2	4	8
4	16	128
8	64	512
16	**256**	**4,096**
32	1,024	32,768
64	4,096	262,144

	GPU + Tensor core	AI core + SRAM
Area (normalised to 12 nm)	5.2 mm^2	13.2 mm^2
Compute power	1.7 FLOPS (FP16)	8 FLOPS (FP16)

Cube Unit
High intensity computation



Advantages of special compute units




```
float a[16][16], b[16][16], c[16][16];
```

Scalar:

```
for (int i=0; i<16; i++) {
    for (int j=0; j<16; j++) {
        for (int k=0; k<16; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

$Cycles = 16 * 16 * 16 * 2 = 8192$
$Data\ per\ cycle = Rd\ 2;\ \ Wr\ 1$

Vector:

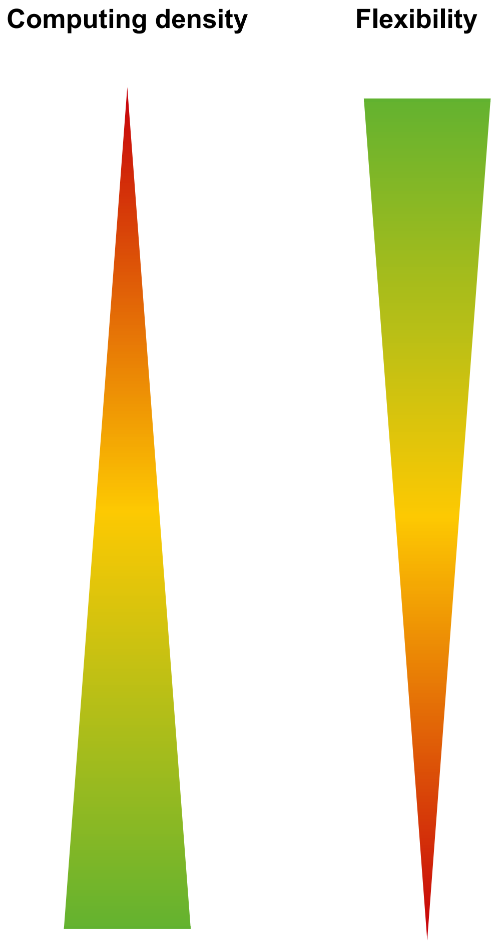
```
for (int i=0; i<16; i++) {
    for (int j=0; j<16; j++) {
        c[i][j] = a[i][:] *+ b[:][j];
    }
}
```

$Cycles = 16 * 16 = 256$
$Data\ per\ cycle = Rd\ 2 * 16;\ \ Wr\ 16$

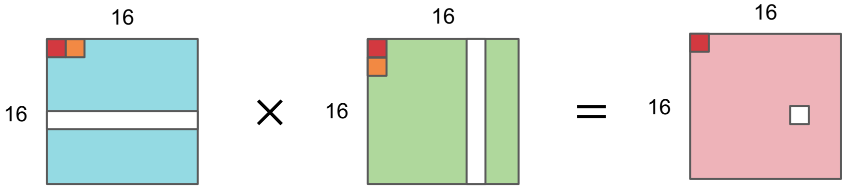
Cube:

```
c[:][:] = a[:][:] X b[:][:]
```

$Cycles = 1$
$Data\ per\ cycle = Rd\ 2 * 16 * 16;\ \ Wr$ 16*16



Advantages of special compute units




```
float a[16][16], b[16][16], c[16][16];
```

Scalar:

```
for (int i=0; i<16; i++) {
    for (int j=0; j<16; j++) {
        for (int k=0; k<16; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

$Cycles = 16 * 16 * 16 * 2 = 8192$
$Data\ per\ cycle = Rd\ 2;\ \ Wr\ 1$

Vector:

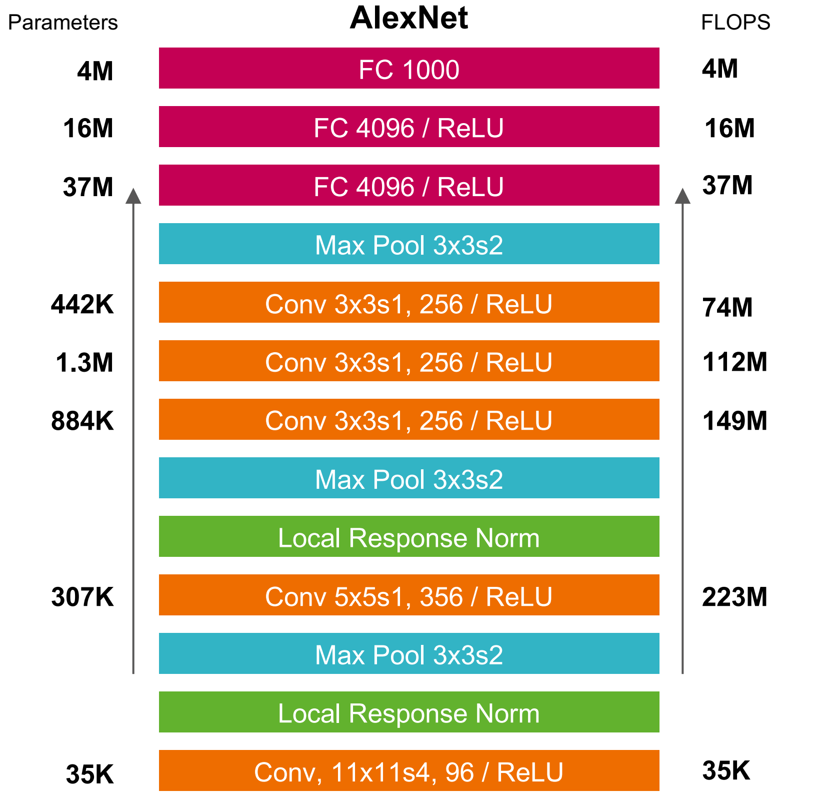
```
for (int i=0; i<16; i++) {
    for (int j=0; j<16; j++) {
        c[i][j] = a[i][:] *+ b[:][j];
    }
}
```

$Cycles = 16 * 16 = 256$
$Data\ per\ cycle = Rd\ 2 * 16;\ \ Wr\ 16$

Cube:

```
c[:][:] = a[:][:] X b[:][:]
```

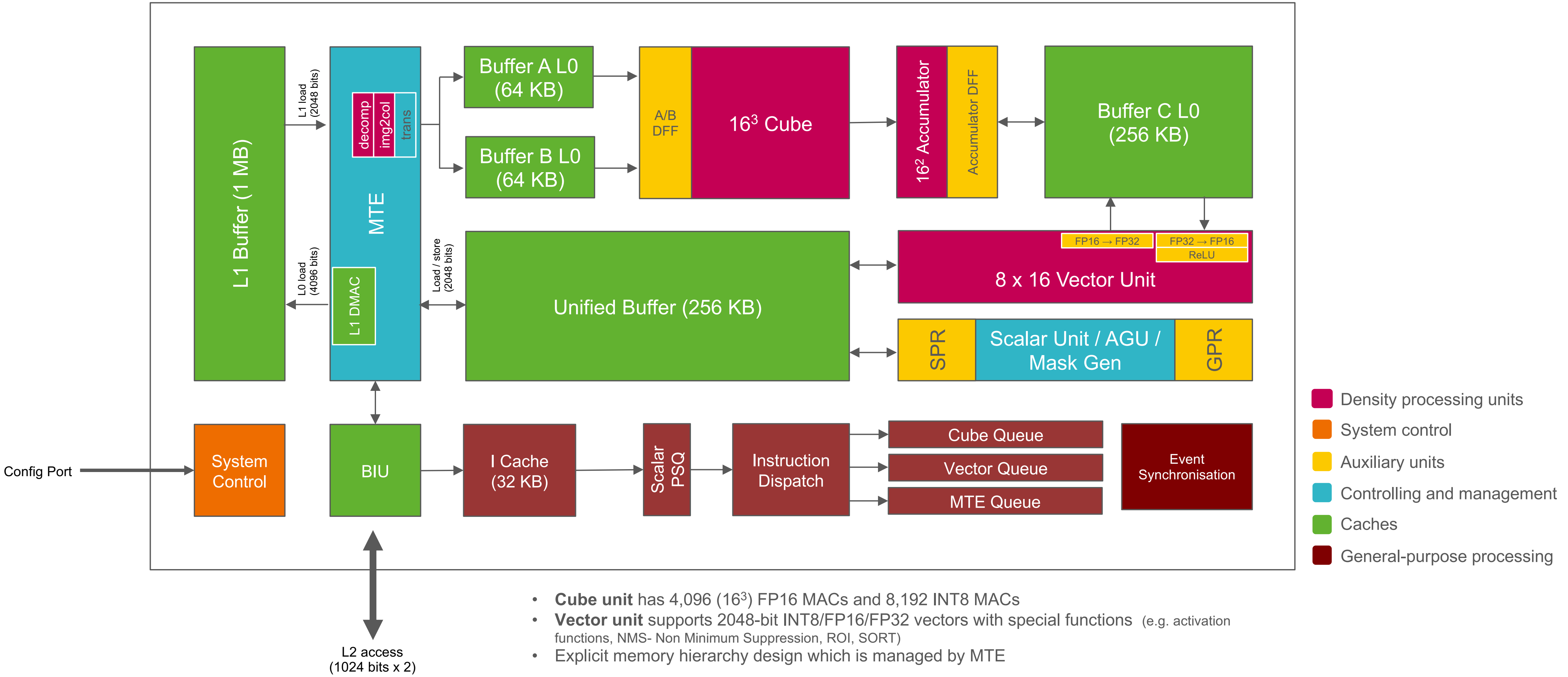
$Cycles = 1$
$Data\ per\ cycle = Rd\ 2 * 16 * 16;\ \ Wr\ 16*16$



Number of parameters and floating point operations per second (FLOPS) for each layer of the AlexNet artificial intelligence model.

99% of the computations are matrix-matrix multiplications	Typical CNN networks		
	AlexNet	**VGG16**	**Inception-v3**
Model memory (MB)	> 200	> 500	90-100
Parameter count (Million)	60	138	23.2
Computation amount (Million)	720	15300	5000

Da Vinci core architecture



- **Cube unit** has 4,096 (16^3) FP16 MACs and 8,192 INT8 MACs
- **Vector unit** supports 2048-bit INT8/FP16/FP32 vectors with special functions (e.g. activation functions, NMS- Non Minimum Suppression, ROI, SORT)
- Explicit memory hierarchy design which is managed by MTE

Micro-architectural configurations

Core scale	Cube Operations/cycle	Vector Operations/cycle	L0 Bus width	L1 Bus width	L2 → Memory Bandwidth
Max	8192	256		A: 8192 B: 2048	Ascend 310: $\frac{192\ GB/s}{2\ cores}$ Ascend 910: $\frac{3\ TB/s}{32\ cores}$
Lite	4096	128		A: 8192 B: 2048	38.4 GB/s
Tiny	512	32		A: 2048 B: 512	N/A
Performance baseline		Minimise vector limitation	Matches with execution units; eliminates bottleneck	Ensure this no a limitation	Limited by network-on-a-chip; avoids additional limitations

End-to-end life cycle

Implementation of end-to-end lifecycle in AI projects [Alake, 2020], [Sato et al., 2019]

- Problem statement
- Ideal problem solution
- Understanding and insight into the problem
- Technical requirements

- Data structure and source
- Solution form
- Model architecture
- Algorithm research
- Hardware requirements

- Data gathering (diverse, unbiased and abundant)

- Data reformatting
- Data cleaning
- Data normalization
- Data augmentation

- Usage of pre-trained models?
- Fine-tuning pre-trained models

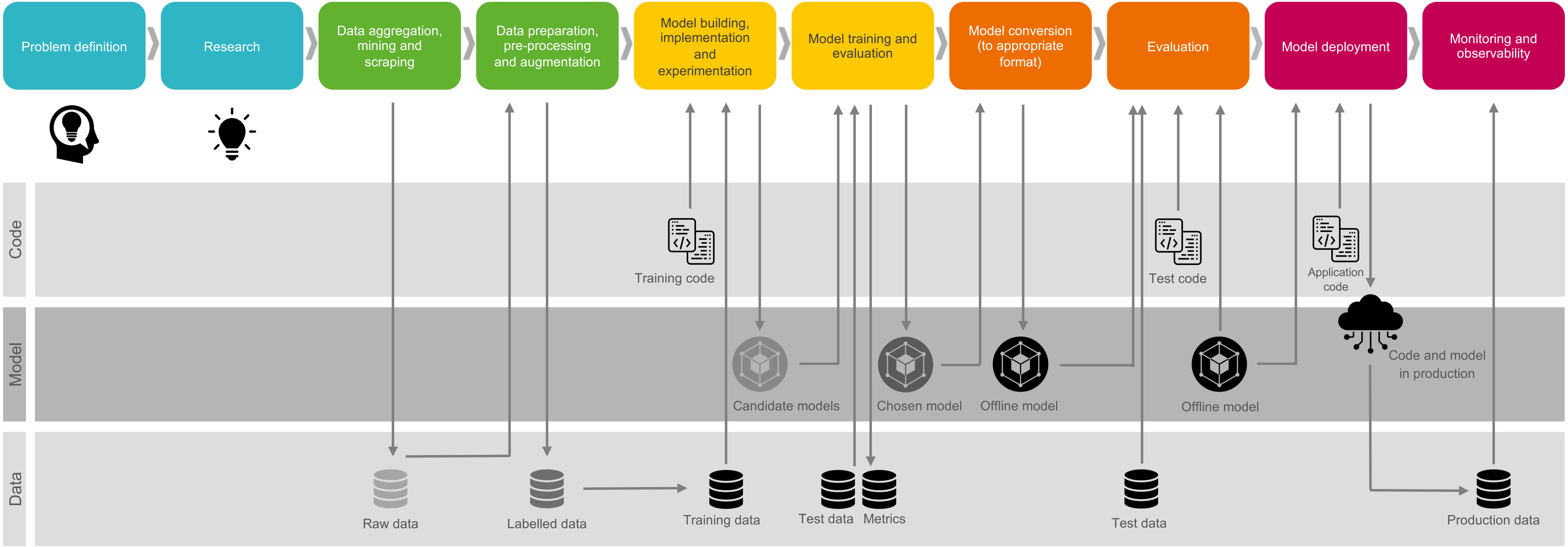
- Training accuracy
- Validation accuracy
- Training loss
- Validation loss
- Underfitting or overfitting?

- Refine and optimise model
- Model conversion
- Mobile-optimised model

- Confusion matrix (error matrix)
- Precision-recall

- UI interface to access model functionalities
- Continuous integration pipeline that enables model redeployment

- Model performance monitoring system



The challenges to AI implementations in projects

- Problem statement
- Ideal problem solution
- Understanding and insight into the problem
- Technical requirements

- Data structure and source
- Solution form
- Model architecture
- Algorithm research
- Hardware requirements

- Data gathering (diverse, unbiased and abundant)

- Data reformatting
- Data cleaning
- Data normalization
- Data augmentation

- Usage of pre-trained models?
- Fine-tuning pre-trained models

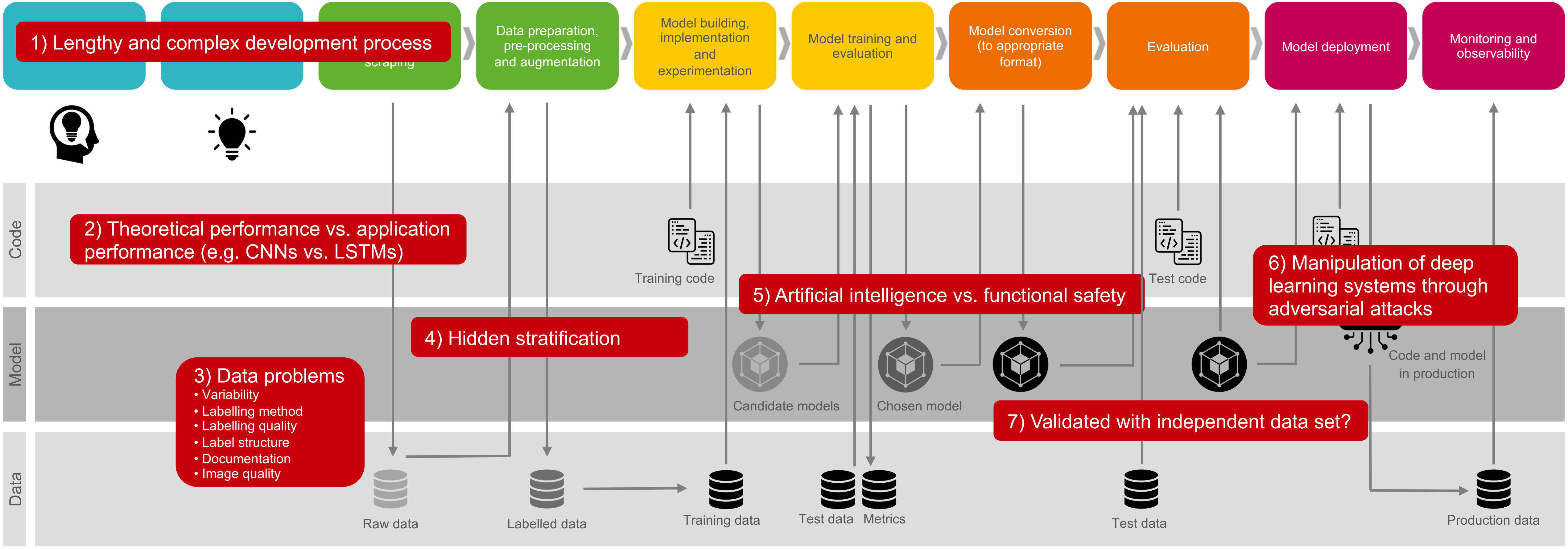
- Training accuracy
- Validation accuracy
- Training loss
- Validation loss
- Underfitting or overfitting?

- Refine and optimise model
- Model conversion
- Mobile-optimised model

- Confusion matrix (error matrix)
- Precision-recall

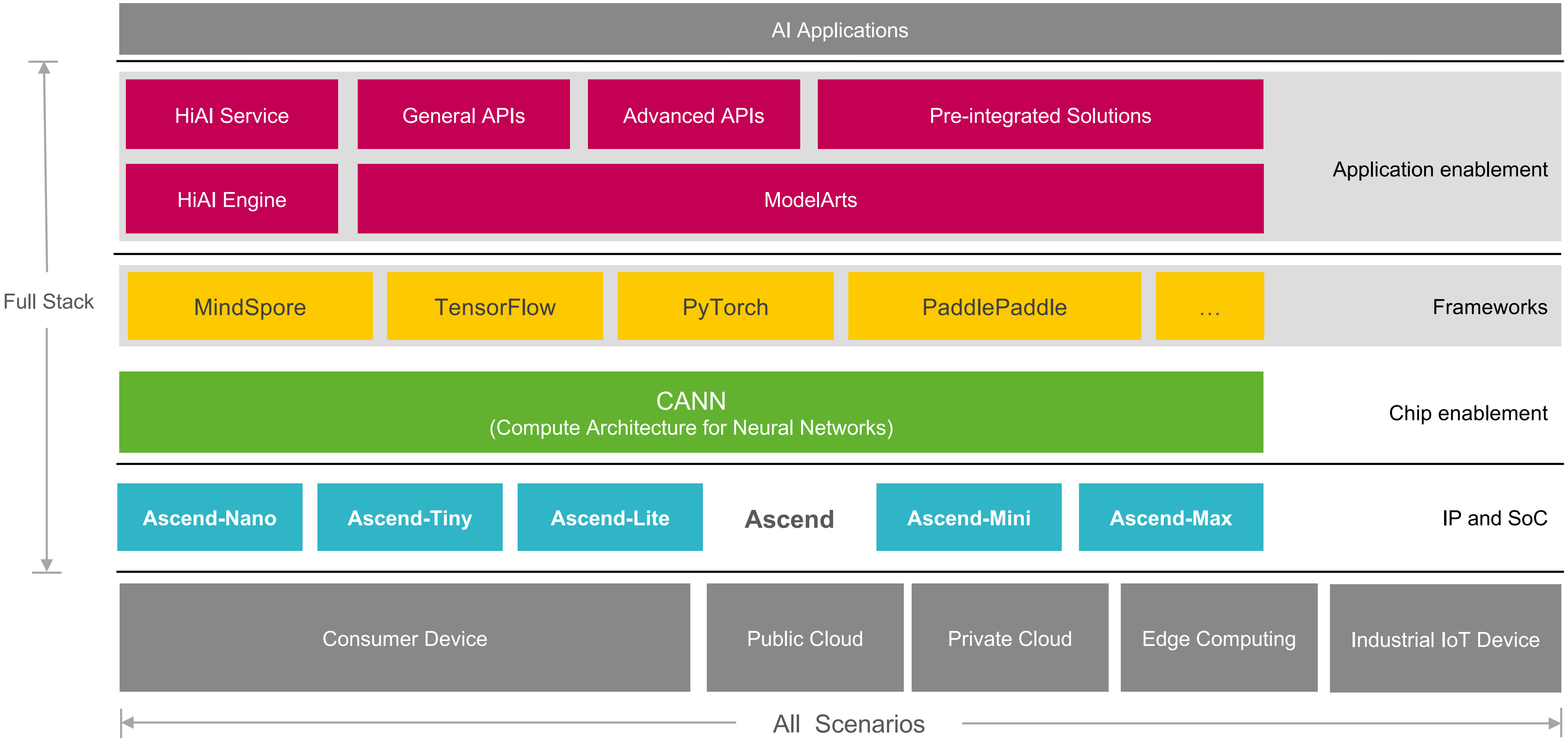
- UI interface to access model functionalities
- Continuous integration pipeline that enables model redeployment

- Model performance monitoring system



Software stack

Ascend AI software stack



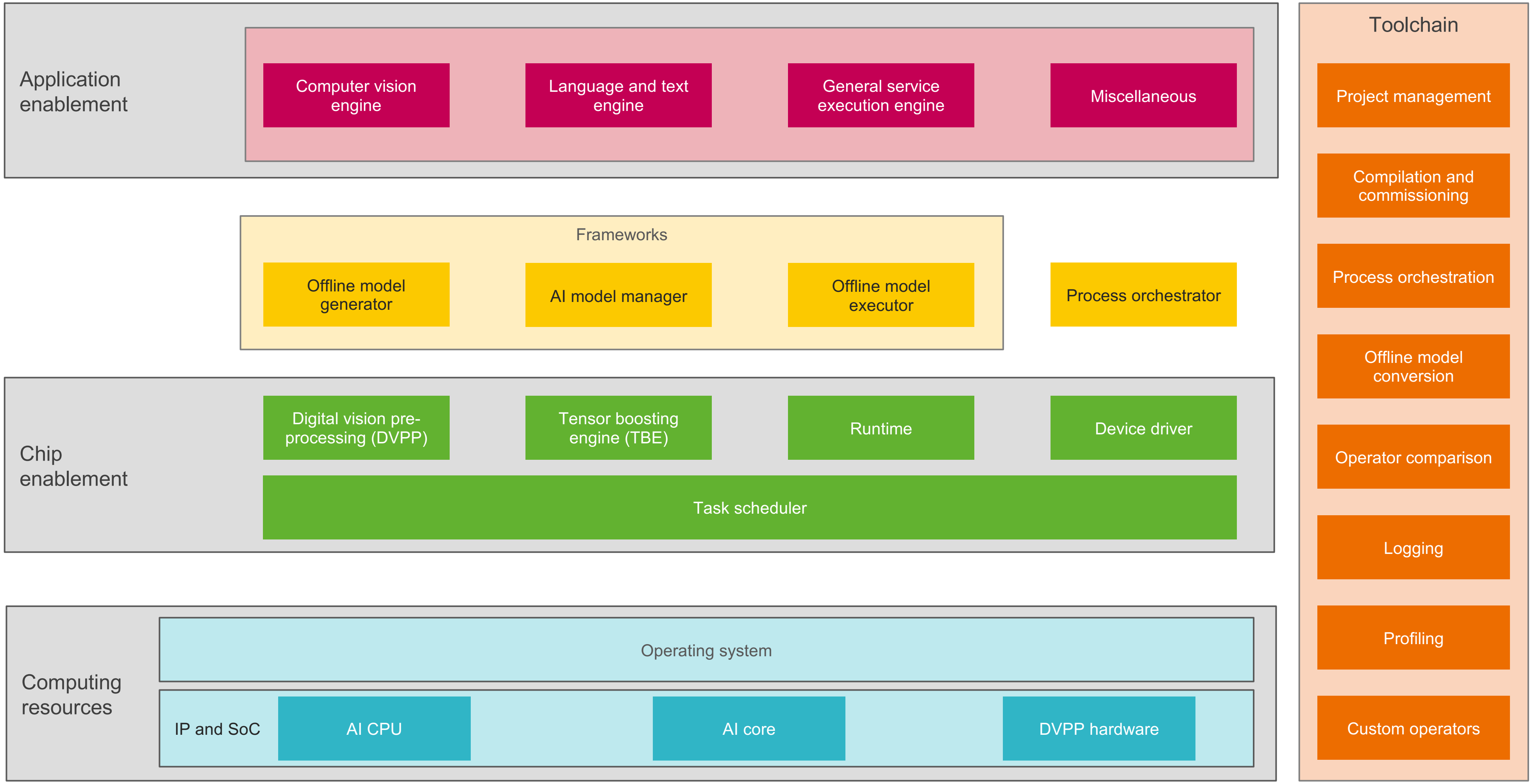
Application enablement:
Full-pipeline services (ModelArts), hierarchical APIs, and pre-integrated solutions

MindSpore:
Unified training and inference framework for device, edge, and cloud (both standalone and cooperative)

CANN:
SoC operator library and highly automated development toolkit

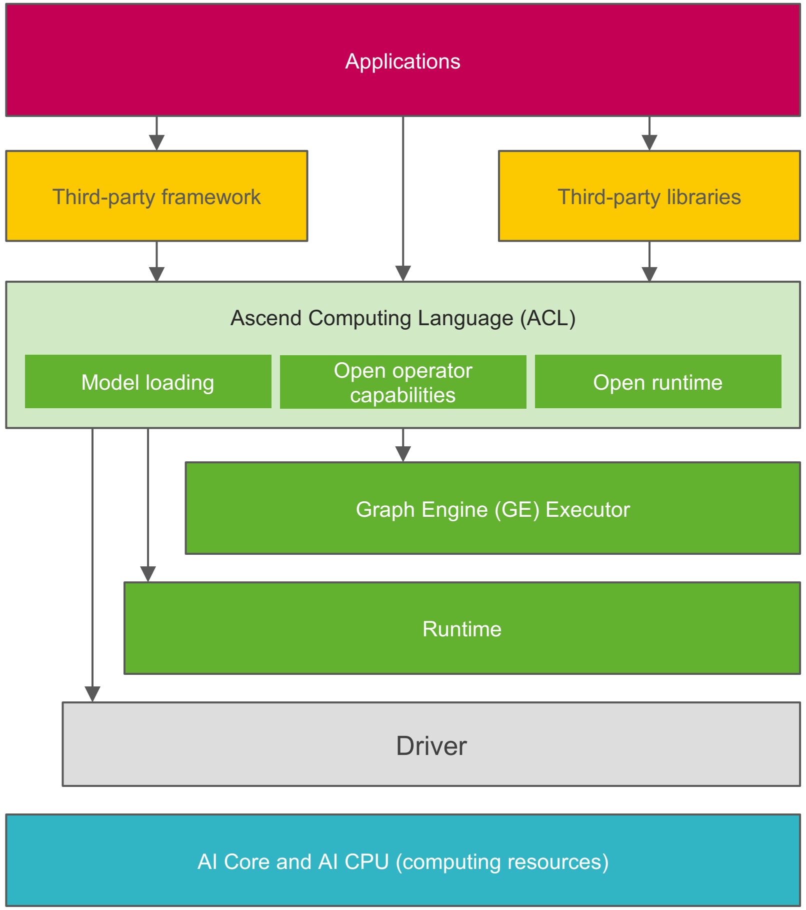
Ascend:
AI IP and SoC series based on unified scalable architecture

Logical architecture of the Ascend AI software stack



- The **application enablement** layer provides different processing algorithms for specific application fields.
- The **frameworks** are providing offline model generation and execution capabilities for the Ascend AI processor.
- **The chip enablement** layer bridges between the offline models and the Ascend AI processor.
- The **computing resource** layer is responsible for executing specific computing tasks allocated from its upper layers.

Chip enablement layer and Ascend Computing Language (ACL)



- Programming Interface for Ascend AI Processor
- C++ and Python APIs
  - Runtime API (Resource Management):
    - Device management
    - Context management
    - Streams management
    - Memory management
  - Model and Operator API:
    - Model loading and execution, operator loading and execution (Graph Engine)
  - Media data processing (DVPP AIPP)

Software flow for model conversion and deployment

Problem definition
- Problem statement
- Ideal problem solution
- Understanding and insight into the problem
- Technical requirements

Research
- Data structure and source
- Solution form
- Model architecture
- Algorithm research
- Hardware requirements

Data aggregation, mining and scraping
- Data gathering (diverse, unbiased and abundant)

Data preparation, pre-processing and augmentation
- Data reformatting
- Data cleaning
- Data normalization
- Data augmentation

Model building, implementation and experimentation
- Usage of pre-trained models?
- Fine-tuning pre-trained models

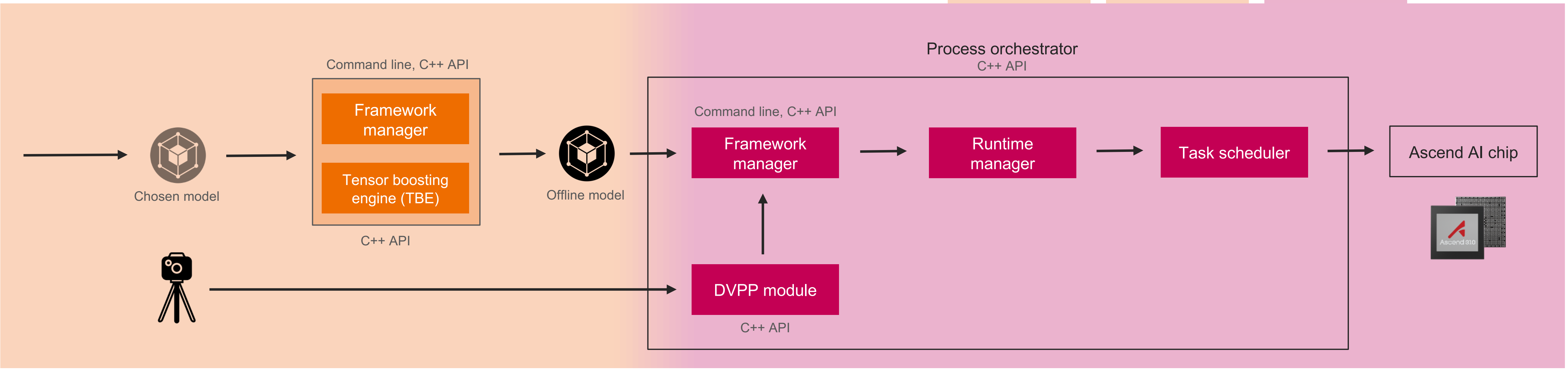
Model training and evaluation
- Training accuracy
- Validation accuracy
- Training loss
- Validation loss
- Underfitting or overfitting?

Model conversion (to appropriate format)
- Refine and optimise model
- Model conversion
- Mobile-optimised model

Evaluation
- Confusion matrix (error matrix)
- Precision-recall

Model deployment
- UI interface to access model functionalities
- Continuous integration pipeline that enables model redeployment

Monitoring and observability
- Model performance monitoring system



Software flow for model conversion and deployment

Problem definition
- Problem statement
- Ideal problem solution
- Understanding and insight into the problem
- Technical requirements

Research
- Data structure and source
- Solution form
- Model architecture
- Algorithm research
- Hardware requirements

Data aggregation, mining and scraping
- Data gathering (diverse, unbiased and abundant)

Data preparation, pre-processing and augmentation
- Data reformatting
- Data cleaning
- Data normalization
- Data augmentation

Model building, implementation and experimentation
- Usage of pre-trained models?
- Fine-tuning pre-trained models

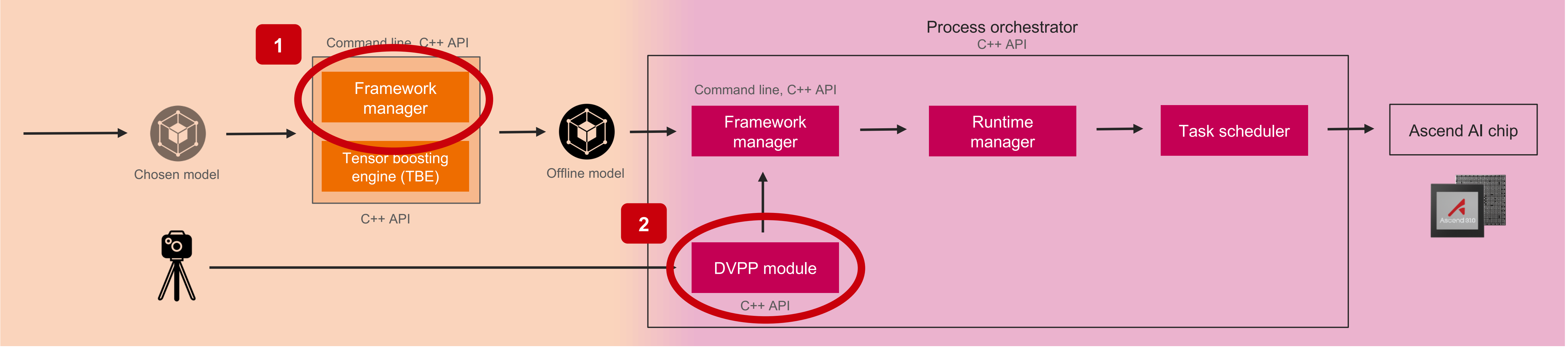
Model training and evaluation
- Training accuracy
- Validation accuracy
- Training loss
- Validation loss
- Underfitting or overfitting?

Model conversion (to appropriate format)
- Refine and optimise model
- Model conversion
- Mobile-optimised model

Evaluation
- Confusion matrix (error matrix)
- Precision-recall

Model deployment
- UI interface to access model functionalities
- Continuous integration pipeline that enables model redeployment

Monitoring and observability
- Model performance monitoring system



Framework manager

Problem definition
- Problem statement
- Ideal problem solution
- Understanding and insight into the problem
- Technical requirements

Research
- Data structure and source
- Solution form
- Model architecture
- Algorithm research
- Hardware requirements

Data aggregation, mining and scraping
- Data gathering (diverse, unbiased and abundant)

Data preparation, pre-processing and augmentation
- Data reformatting
- Data cleaning
- Data normalization
- Data augmentation

Model building, implementation and experimentation
- Usage of pre-trained models?
- Fine-tuning pre-trained models

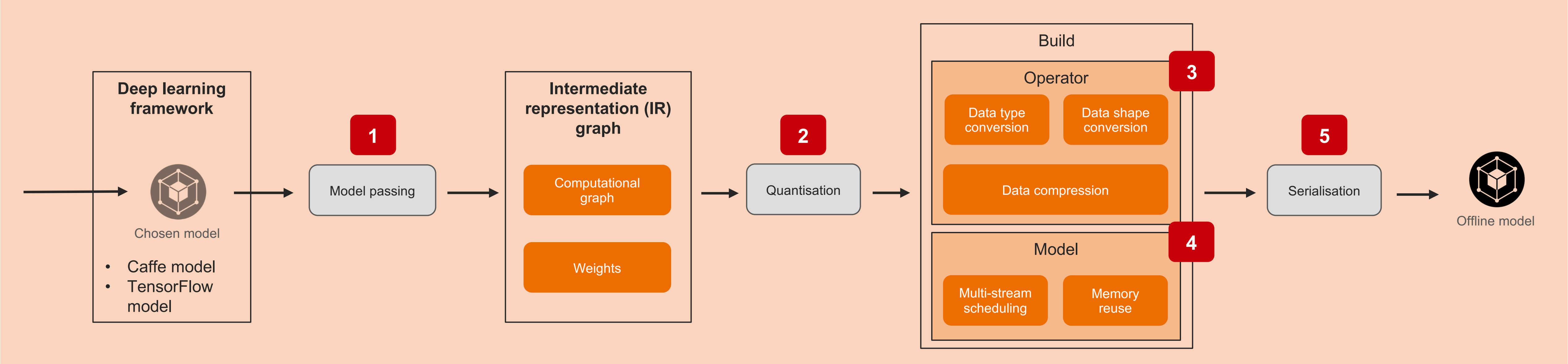
Model training and evaluation
- Training accuracy
- Validation accuracy
- Training loss
- Validation loss
- Underfitting or overfitting?

Model conversion (to appropriate format)
- Refine and optimise model
- Model conversion
- Mobile-optimised model

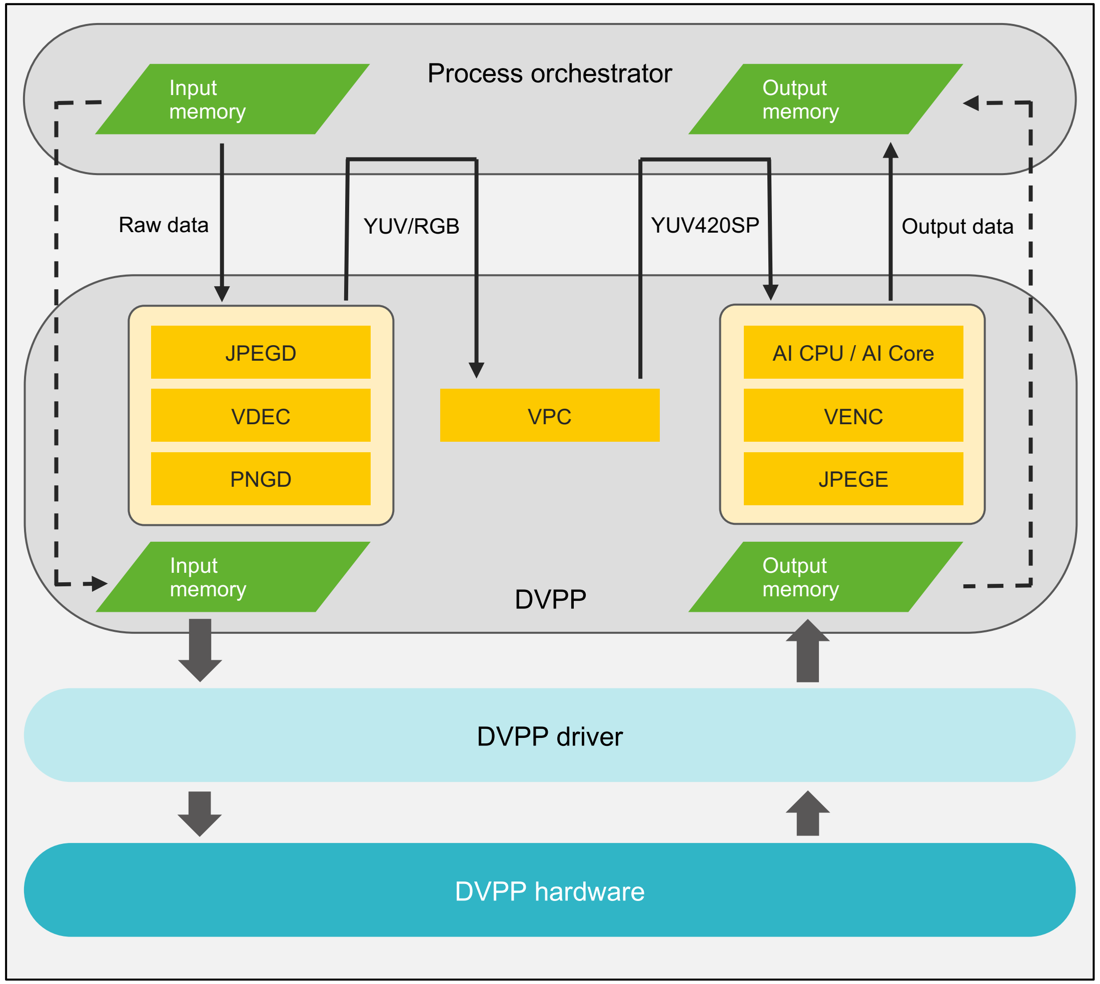
Evaluation
- Confusion matrix (error matrix)
- Precision-recall

Model deployment
- UI interface to access model functionalities
- Continuous integration pipeline that enables model redeployment

Monitoring and observability
- Model performance monitoring system

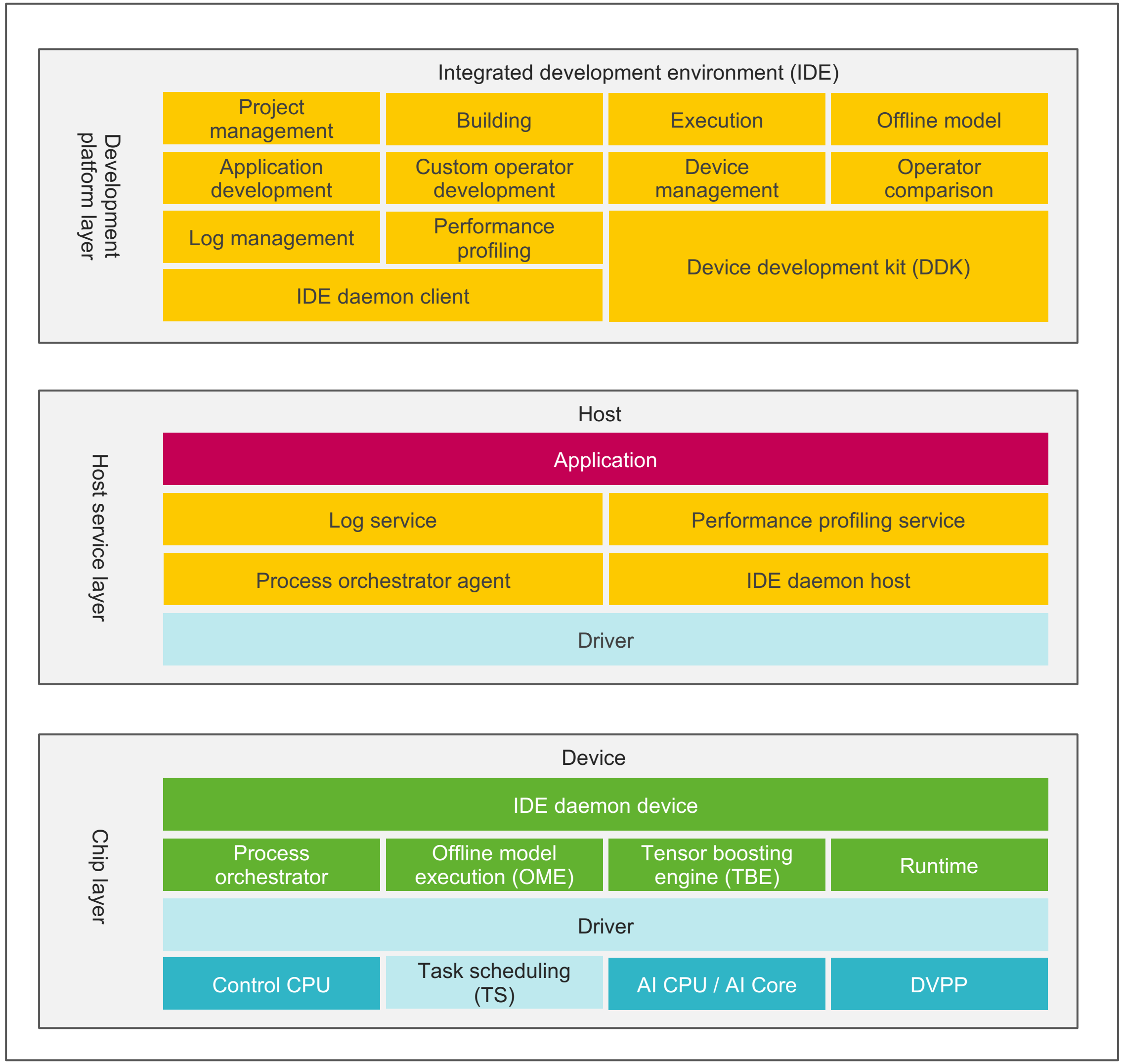


Digital vision pre-processing (DVPP)

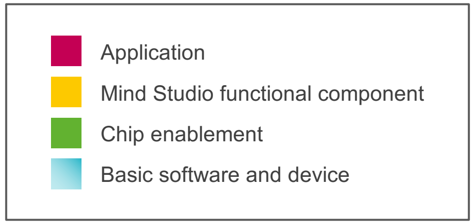


- **The DVPP provides the following six external interfaces:**
  - **Video decoder** (VDEC) decodes H.264/H.265 videos and outputs images for video pre-processing.
  - **Video encoder** (VENC) module encodes output data of DVPP or the raw input YUV data into H.264/H.265 videos for playback and display.
  - **JPEG picture decoder** (JPEGD) module decodes the JPEG images, converts their format into YUV, and pre-processes the inference input data for the neural network.
  - **JPEG picture encoder** (JPEGE) module is used to restore the format of processed data to JPEG for the post-processing of the inference output data of the neural network.
  - **PNG picture decoder** (PNGD) module needs to be called to decode the image into the RGB format before it is output to the Ascend AI processor for inference and computing.
  - **Vision pre-processing core** (VPC) module provides other image and video processing functions, such as format conversion (for example, conversion from YUV/RGB to YUV420), resizing, and cropping.

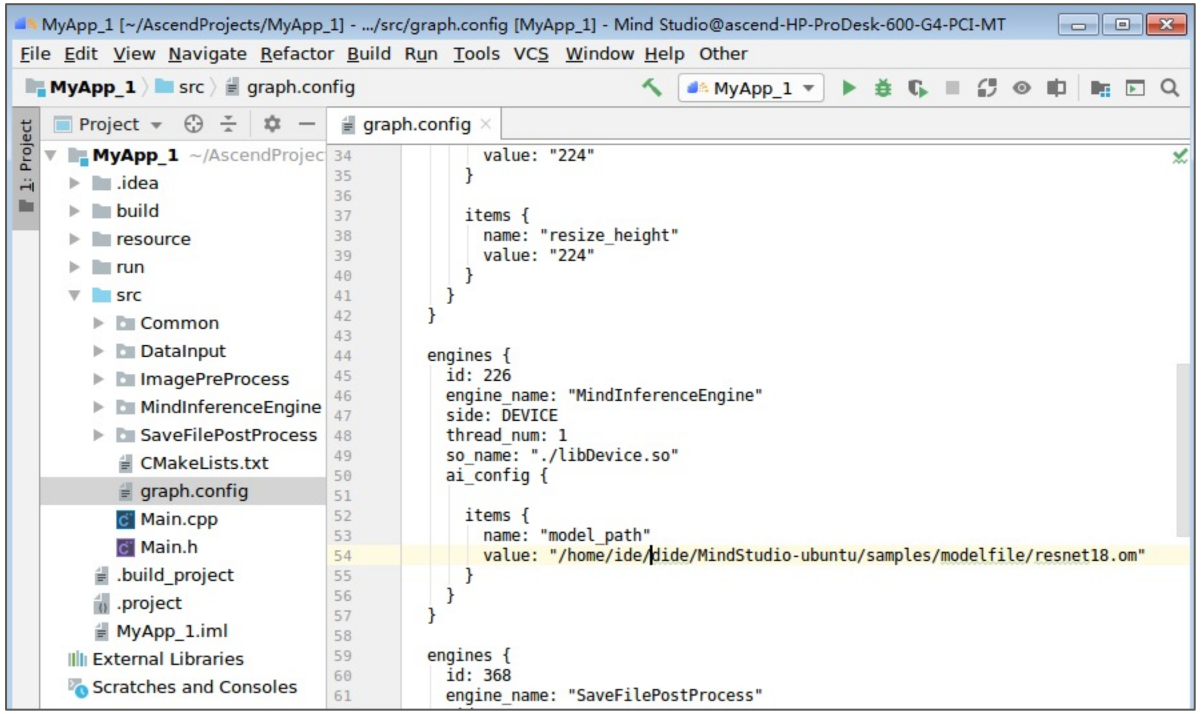
Mind Studio /1

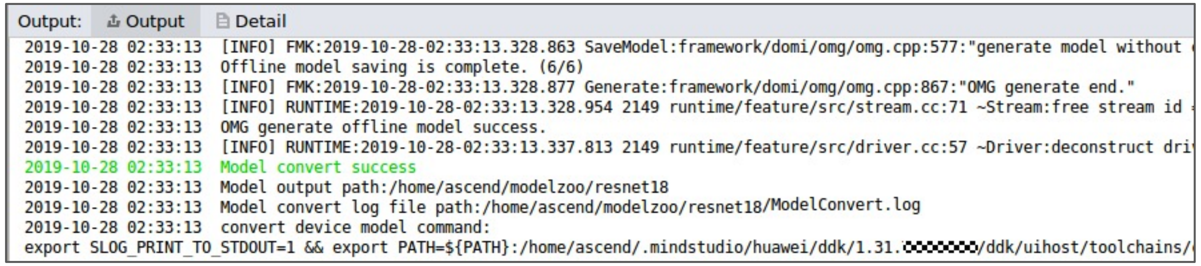


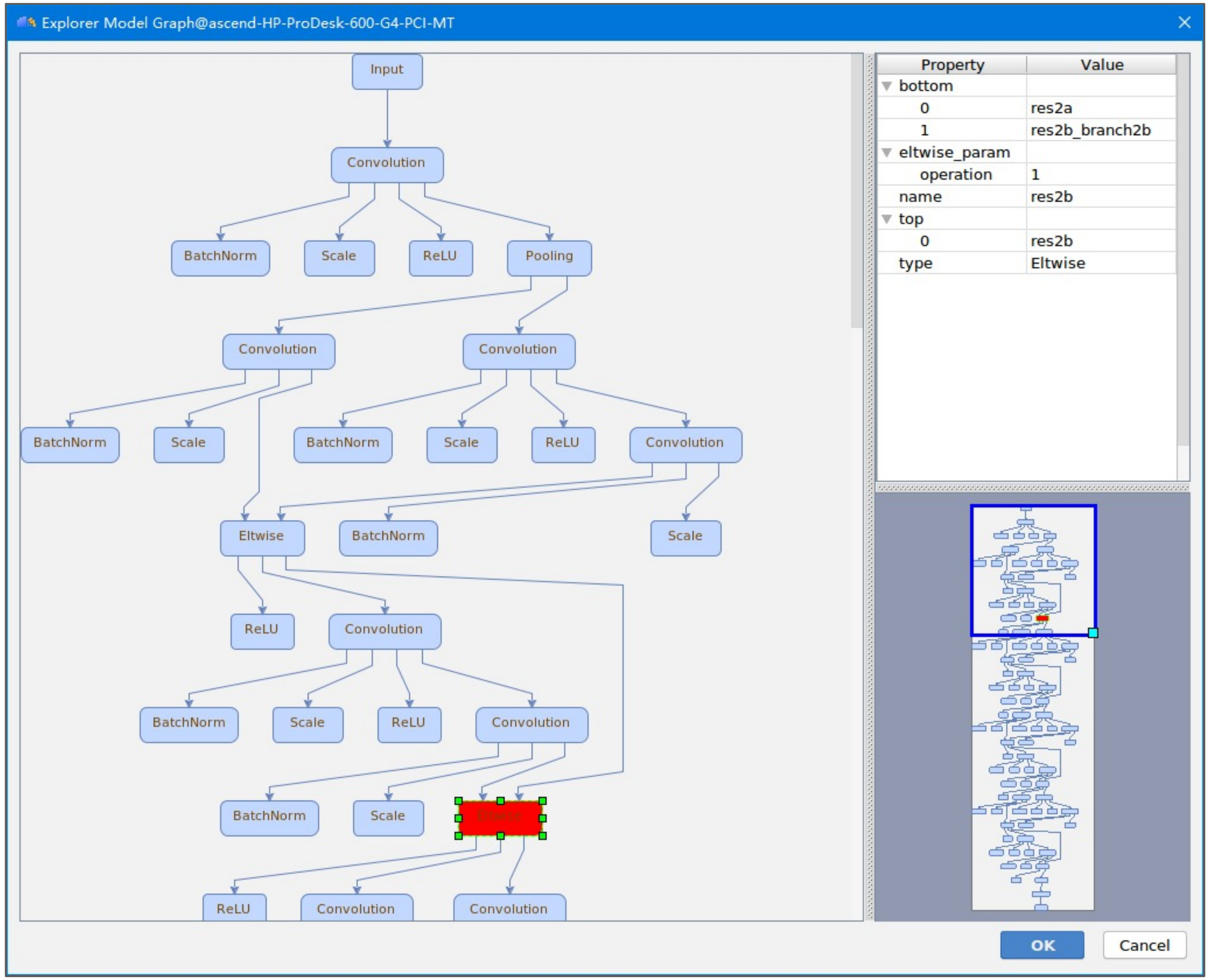
- **Mind Studio is an Intellij-based development toolchain platform.**
- **Mind Studio offers the following features:**
  - Project management
  - Development and building of operators, computing engines, and applications
  - Execution of developed operators and computing engines on Ascend AI processor
  - Debugging
  - Process orchestration
  - Custom operator development
  - Offline model conversion for converting trained third-party network models
  - Log management for system-wide log collection and analysis
  - Performance profiling that enables efficient, easy-to-use, and scalable systematic performance analysis
  - Device management for managing devices connected to the host
  - Operator comparison for comparing the execution results
  - DDK installation and management for streamlining AI algorithm development



Mind Studio /2



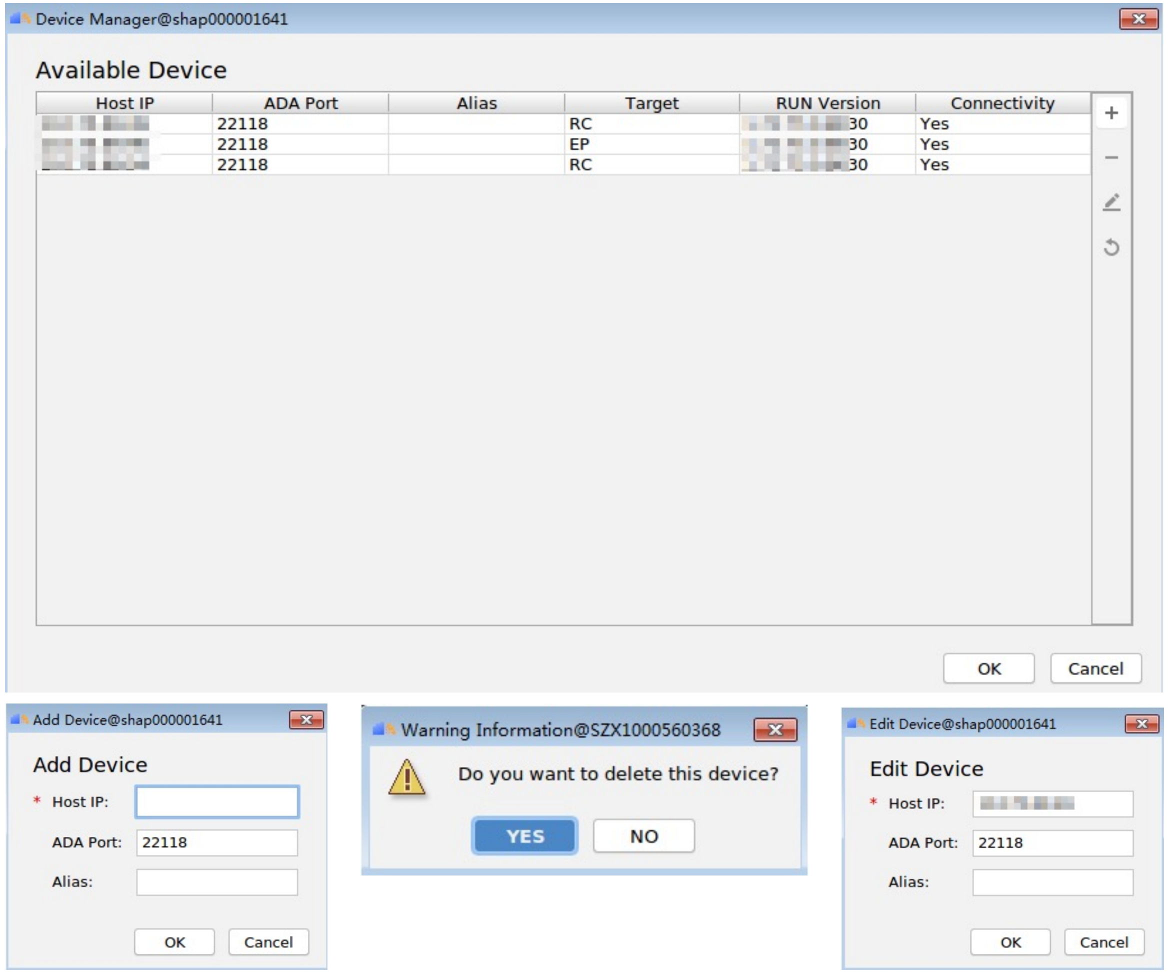




Mind Studio /3

— Device Manager

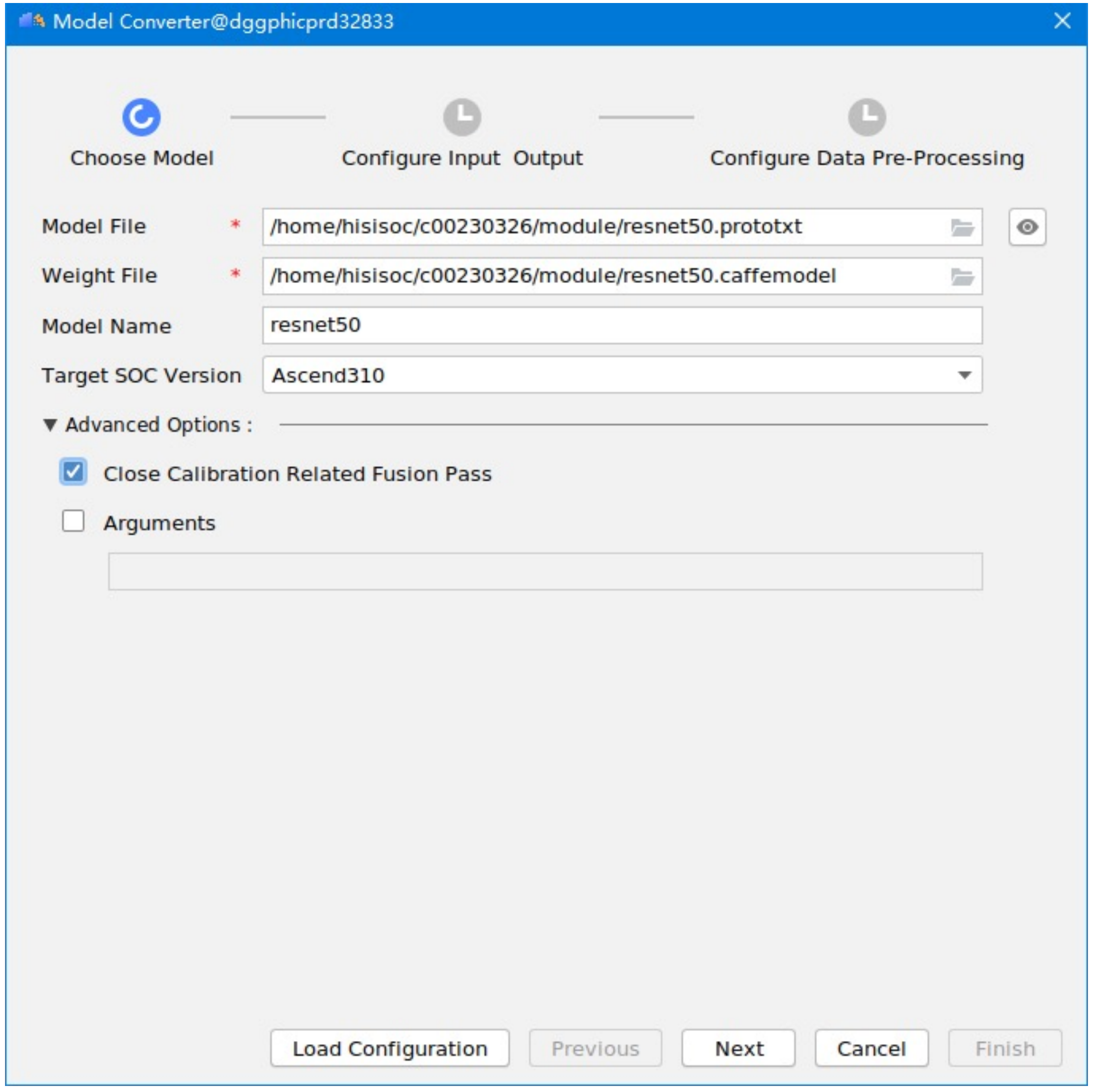
The Device Manager allows you to add, delete, and modify devices.
Choose **Tools** ▸ **Device Manager** from the main menu of Mind Studio.



Parameter or icon	Description
Host IP	Device IP address
ADA port	Port number used by the ADC to communicate with the ADA. The value range is [20000, 25000]. Ensure that the configured port is not occupied. You can run the `netstat -an | grep` *`PortNumber`* command to check whether a port is occupied. Defaults to **22118**.
Alias	Device alias. Using aliases can help manage devices expediently when multiple devices are connected.
Target	Device type **EP**: ASIC form, such as Atlas 200/300/500 **RC**: Atlas 200 DK
Run version	Version of the software package
Connectivity	Status of the connection between Mind Studio and the device: **YES**: connected **NO**: disconnected
+	Adds a device. After a device is added, you can click this icon to add more devices.
−	Deletes a device. You can select a device and click this icon to delete it.
	Edits a device. Select a device to be edited and click this icon to modify the **Host IP**, **ADA Port**, and **Alias** properties.
	Checks the device connection status. After modifying the device information, you can click this icon to refresh the device connection status, software version number, and device type.

Mind Studio /4

— Model Converter



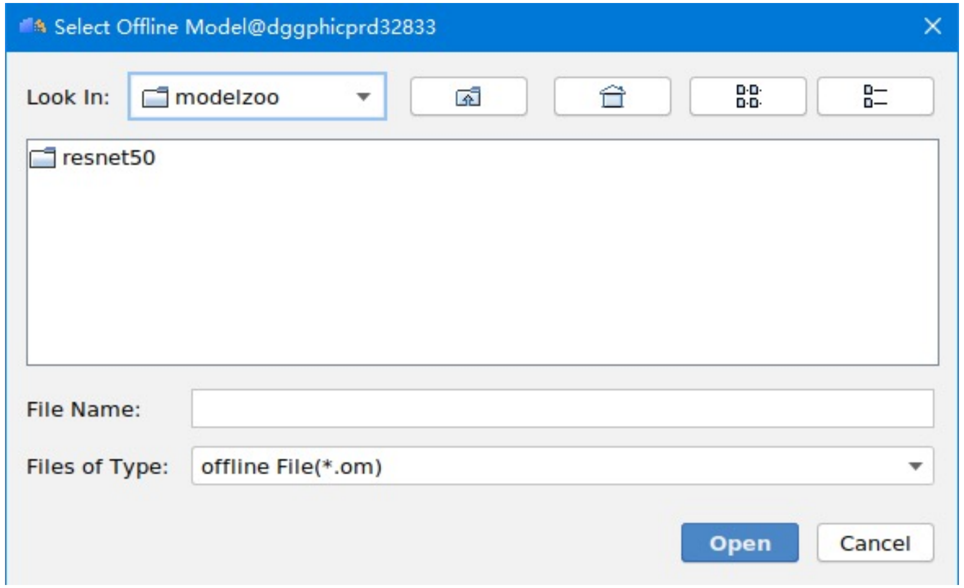
Trained models under frameworks such as Caffe and TensorFlow can be converted into offline models compatible with the Ascend AI processor by using the Ascend Tensor Compiler (ATC). During offline model conversion, you can enable operator scheduling optimization, weight data re-orchestration, and memory usage optimization, thereby preprocessing your models without depending on the device.

Mind Studio /5

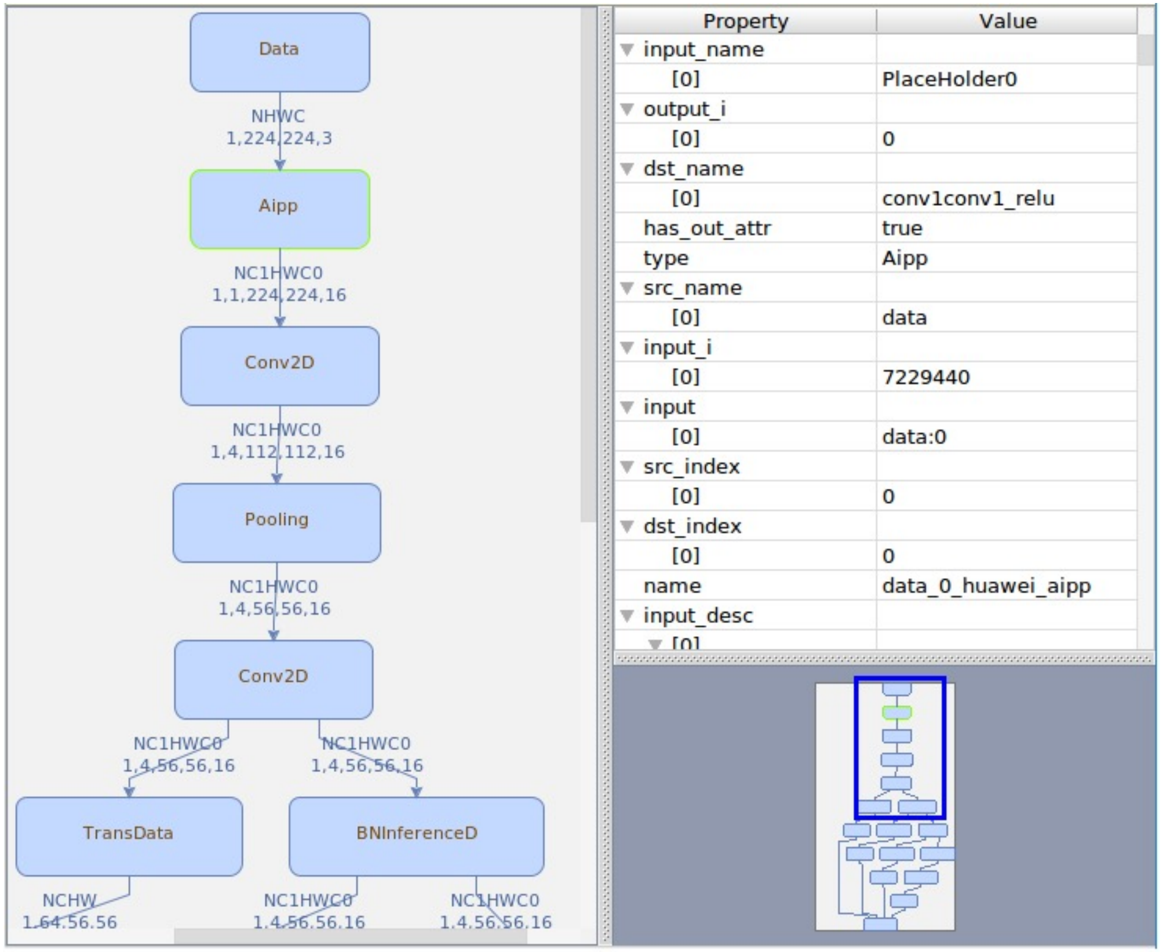
— Model Visualizer

The .om model file of a successfully converted model can be visualized in Mind Studio, so that you can view the network topology including all operators in the model.

On the menu bar, choose **Tools ▸ Model Visualizer**.



Choose **resnet50 ▸ device**, select the converted `resnet50.om` model file, and click **Open**.



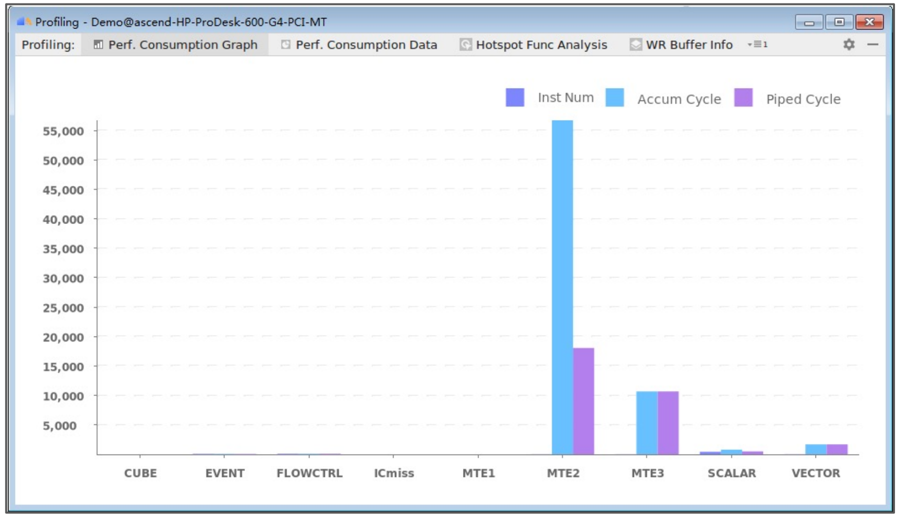
Mind Studio /6

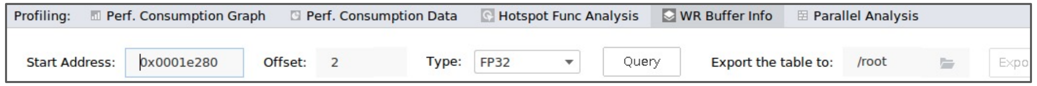
— Profiler

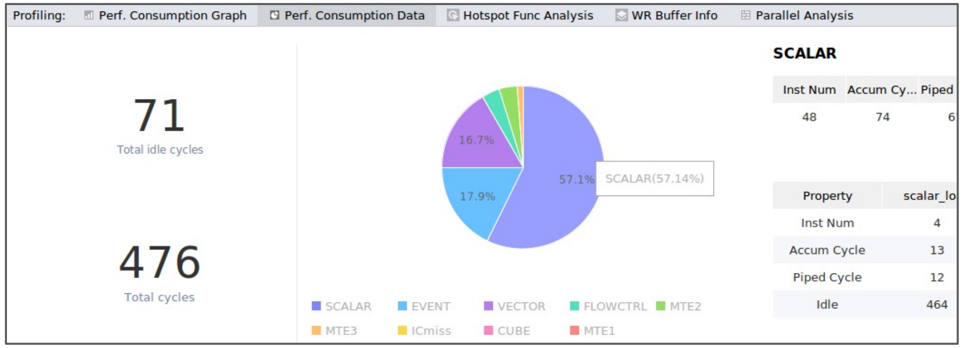
For a single-operator simulation project, set **Target** to **Simulator_Performance** and run the test cases. After profiling is successfully executed, the profiling data generated during simulation is displayed on the console in the lower part of the IDE.

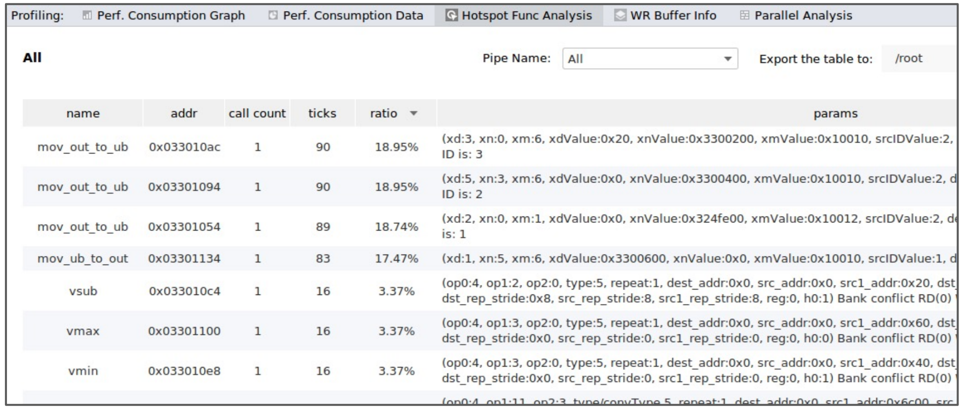
Right-click the operator project name and choose **View Profiling Result** from the shortcut menu to view the profiling result, covering the following matrices:

- Perf. Consumption Graph
- Perf. Consumption Data
- Hotspot Function Analysis
- WR BufferInfo
- Parallel Analysis







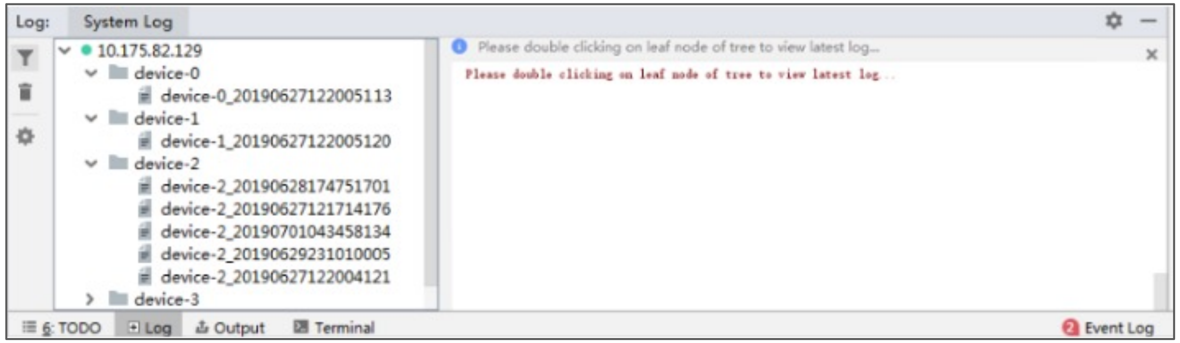


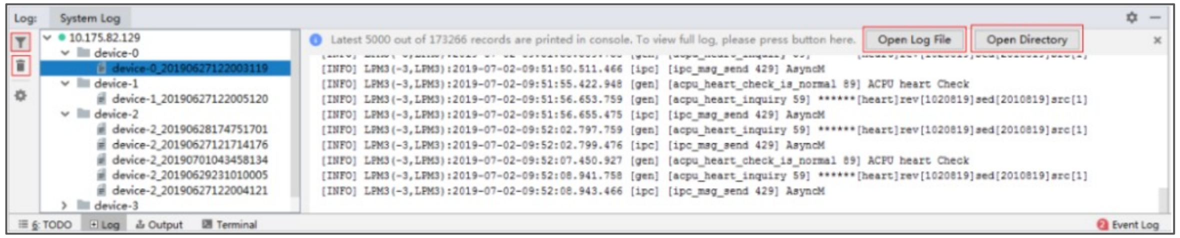
Mind Studio /7

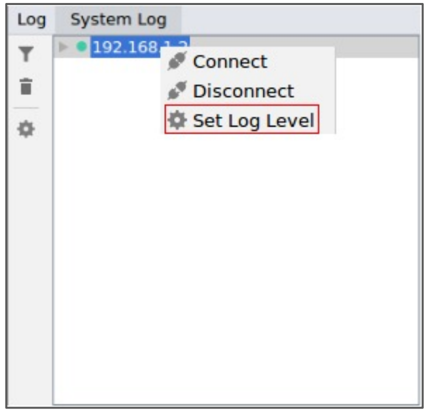
— Log Manager

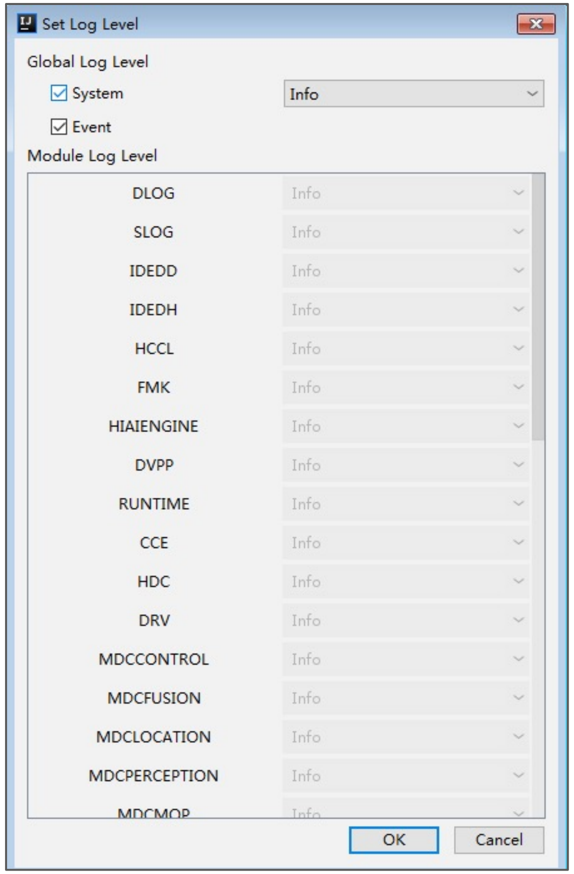
Mind Studio provides a system-wide log collection and analysis solution for the Ascend AI Processor, improving the efficiency of locating algorithm problems at runtime. Mind Studio also provides a unified log format and a GUI for visualized analysis of cross-platform logs and runtime diagnosis, facilitating the use of the log analysis system.

Log management: Click the **+Log** tab at the bottom of the Mind Studio window.









Model Zoo (excerpt)

Image classification

- AlexNet
- DenseNet
- DenseNet-121
- EfficientNet
- GoogLeNet
- Inception V4
- MobileNet v2
- ResNet-101
- ResNet-50
- ResNeXt-50
- SquezzeNet
- VGG16
- VGG19
- …

Object detection and classification

- FasterRCNN
- MobileNet v2
- OpenPose
- Yolo v3
- …

NLP and machine translation

- BERT
- BERT-Base
- NMT / GNMTv2
- TinyBERT
- Transformer
- …

Speech recognition

- Jasper
- WaveGlow
- WaveNet
- Tacotron
- …

Recommendation

- DeepFM
- NCF
- Wide & Deep
- …

Text recognition and classification

- CNN & CTC
- CRNN
- CTPN
- DeepText
- EAST/AdvancedEast
- PSENet
- TextCNN
- XLNET
- …

Domain	Models	Domain	Models	Domain	Models	Domain	Models
3D Reconstruction	6	Language Modelling	55	Recommendation Systems	25	Speech Synthesis	3
Click-Through Rate Prediction	6	Machine Translation	55	Scene Text Detection	32	Text Classification	35
Object Recognition	9	Medical Image Segmentation	18	Semantic Segmentation	82	Video Super-Resolution	13
Image Classification	172	Object Detection	88	Sentiment Analysis	73	Optical Character Recognition	3
Image Super-Resolution	58	Pose Estimation	41	Speech Enhancement	12		
Instance Segmentation	34	Question Answering	66	Speech Recognition	32		

Gain more practical experiences

Atlas 200 DK developer board

Right isometric view

Lower left isometric view

Open inside view

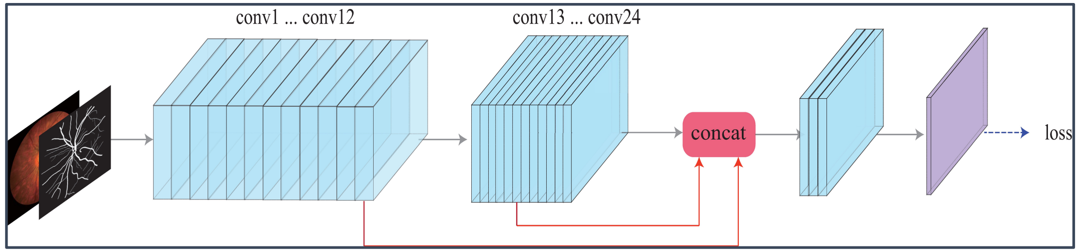
AI Compute Power	• Up to 8 TFLOPS FP16 (16 TOPS INT8) 3 options: 16 TOPS, 8 TOPS, and 4 TOPS
Memory	• LPDDR4x, 8 GB, and 3,200 Mbit/s
Storage	• 1 Micro-SD (TF) card slot, supporting SD 3.0 and a maximum rate of SDR52
Network Port	• 10/100/1000Mbps Ethernet RJ45 port
USB Port	• 1 USB 3.0 Type-C port, which is used only as a slave device and compatible with USB 2.0
Other Ports	• 1 x 40-pin I/O connector • 2 x onboard microphones

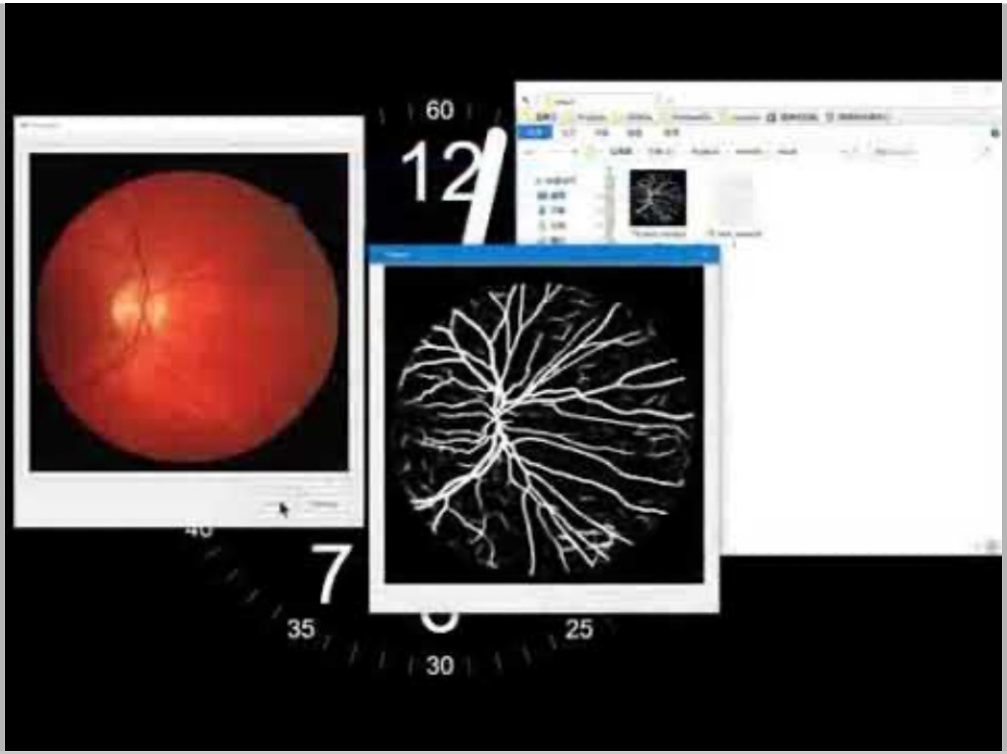
Camera	• 2 x 15-pin Raspberry Pi Camera connectors, supporting the Raspberry Pi v1.3 and v2.1 camera modules
Power Supply	• 5V to 28V DC. 12V 3A adapter is configured by default
Dimensions (H x W x D)	• 32.9 mm x 137.8 mm x 93.0 mm
Power Consumption	• 20W
Weight	• 234g

Retinal blood vessel segmentation in the eyeground

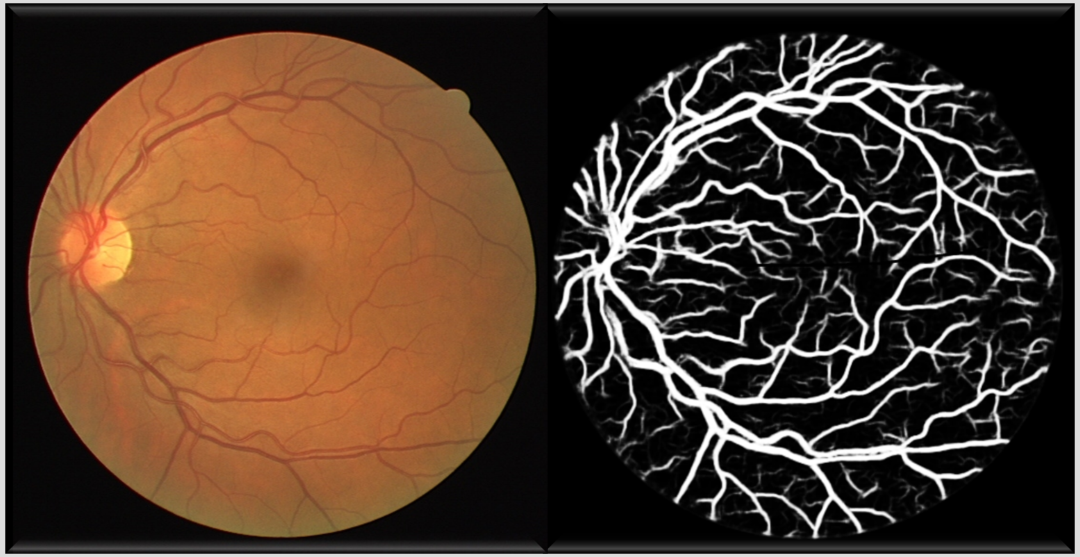
- The fundus retinal blood vessel segmentation application was developed for the Atlas 200 DK inference system, in partnership with the Nankai University, led by Professor Li Tao of Intelligent Computing System Research Office .
- This project makes full use of the neural network computing power of the Atlas 200 DK system to segment the fundus vessels in real-time.
- The total inference time of **20 pictures is 761.8 milliseconds**, and the average inference time of one image is 38 milliseconds.

An overview of the vascular segmentation model





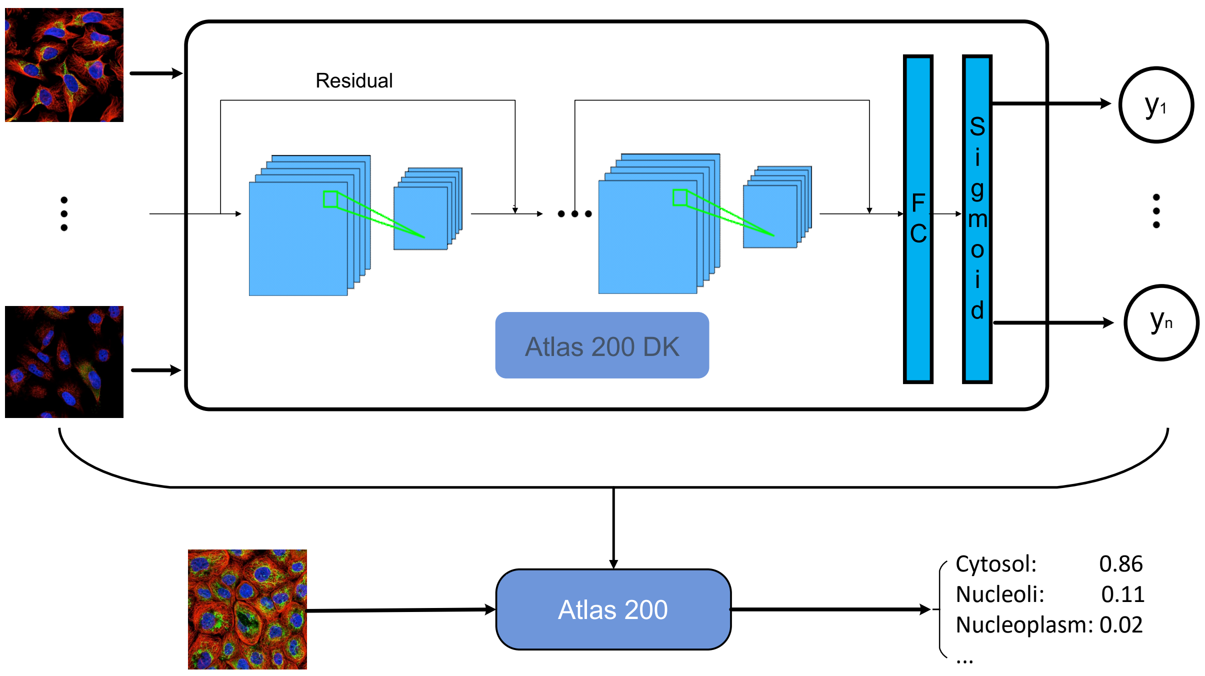
Before and after



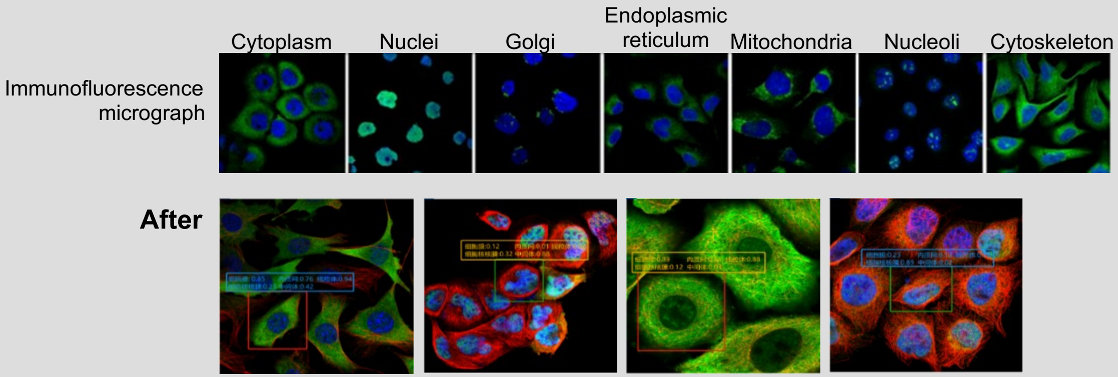
Prediction of protein subcellular localization

- Use deep learning tools to accurately identify the organelles where proteins are located in human protein fluorescence micrographs
- The trained model was executed on the Atlas 200 DK developer kit, and use Atlas 200 DK
- The model analyses unlabelled protein fluorescence and predicts the location of sub-cells with pictures
- Protein subcellular localisation prediction targets the microscopic fluorescence images of proteins in cancer tissues and other tissues to identify the localisation of proteins; to find location markers related to cancer

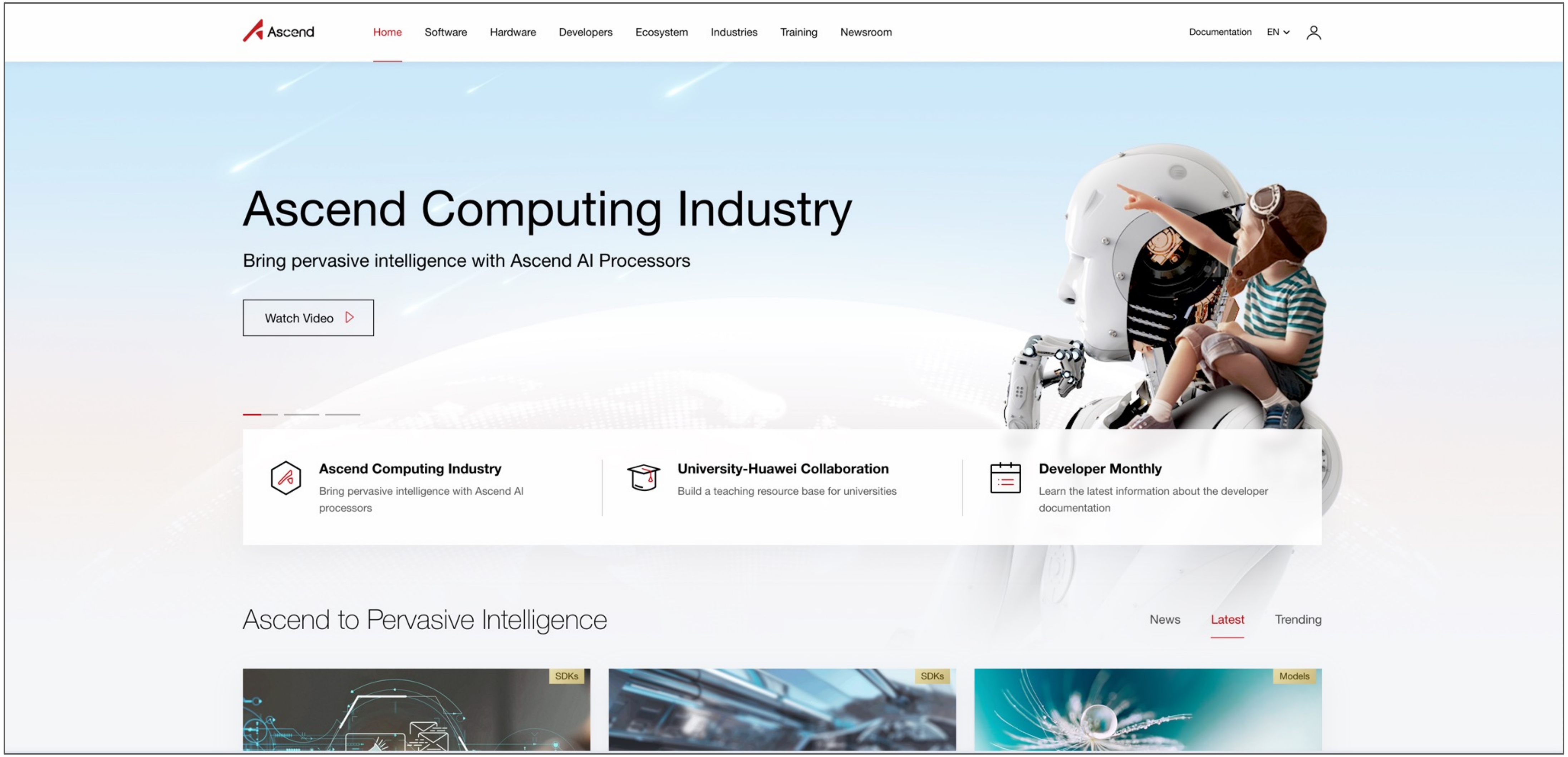
An overview of the artificial intelligence model



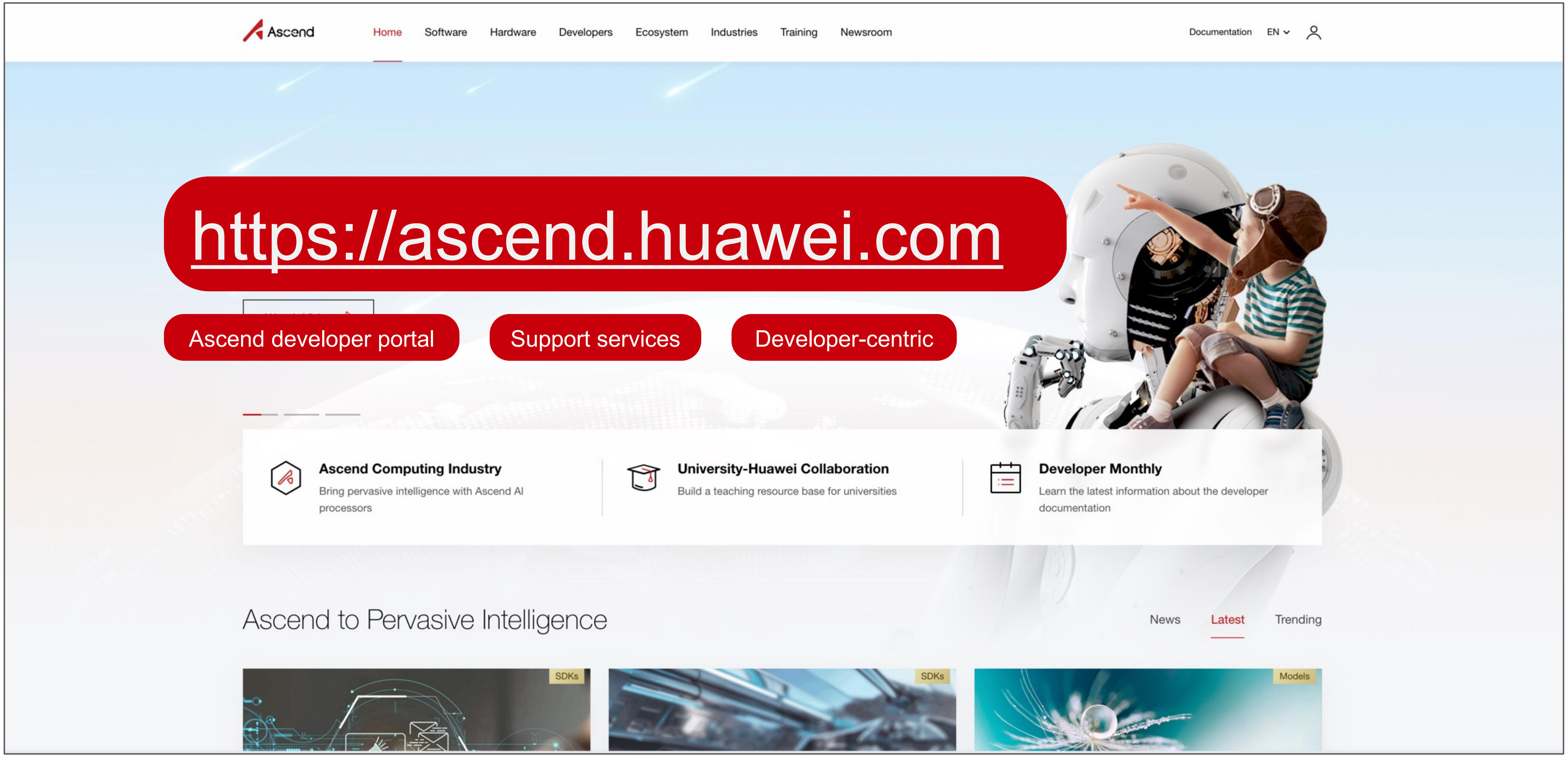
Video



Ascend developer community



Ascend developer community



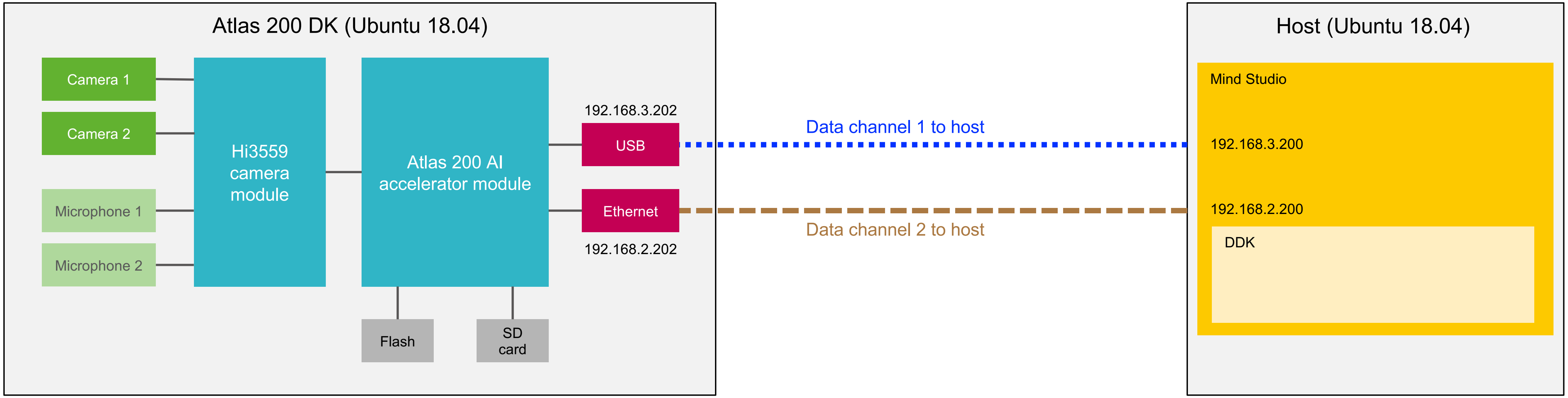
Getting started with Atlas 200 DK developer board

1. **Preparing the Ubuntu-based development environment**
   - Install Python, xterm, Firefox, fonts, numpy, OpenJDK, etc.
   - Modify `.bashrc` file
2. **Setting up the Atlas 200 DK hardware environment**
   - Removing the upper case and install the camera
3. **Create and write SD card image**
   - Download and verify software packages
   - Write image to the SD card
4. **Boot and connect to the Atlas 200 DK developer board**
5. **Install third-party packages**
6. **Create the first project: Colourful Image Colourisation**

Preparing the Ubuntu-based development environment

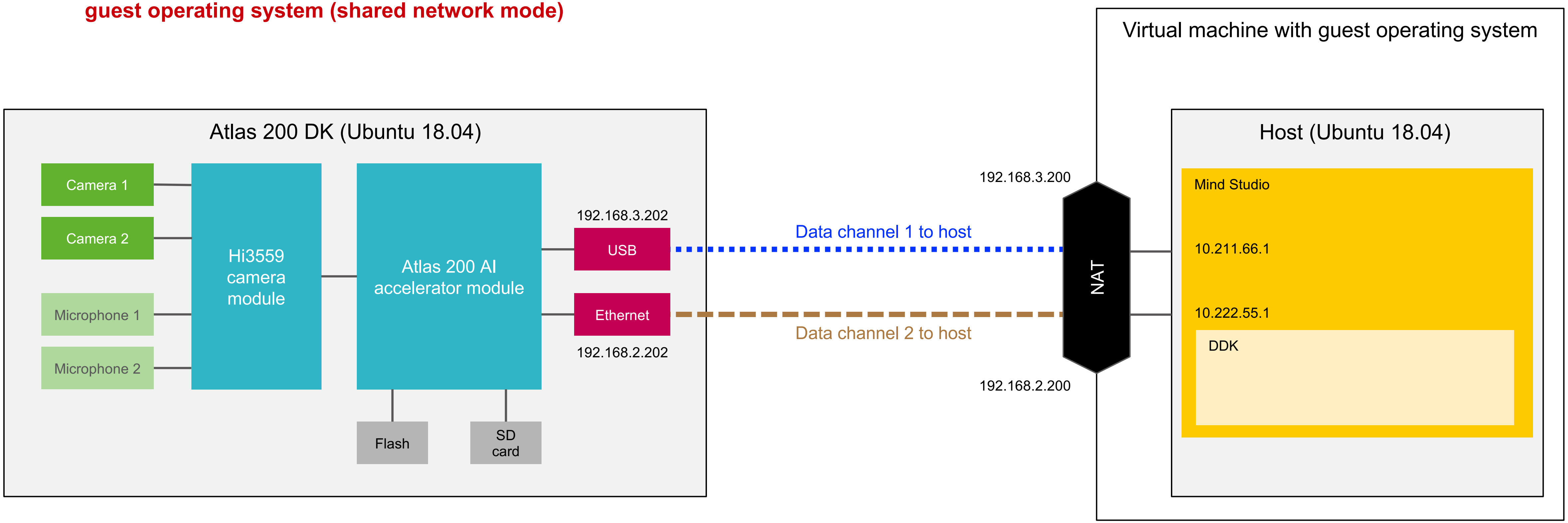
Environment deployment /1

— Direct USB or Ethernet connection



Environment deployment /2

— Direct USB or Ethernet connection and virtual machine with guest operating system (shared network mode)

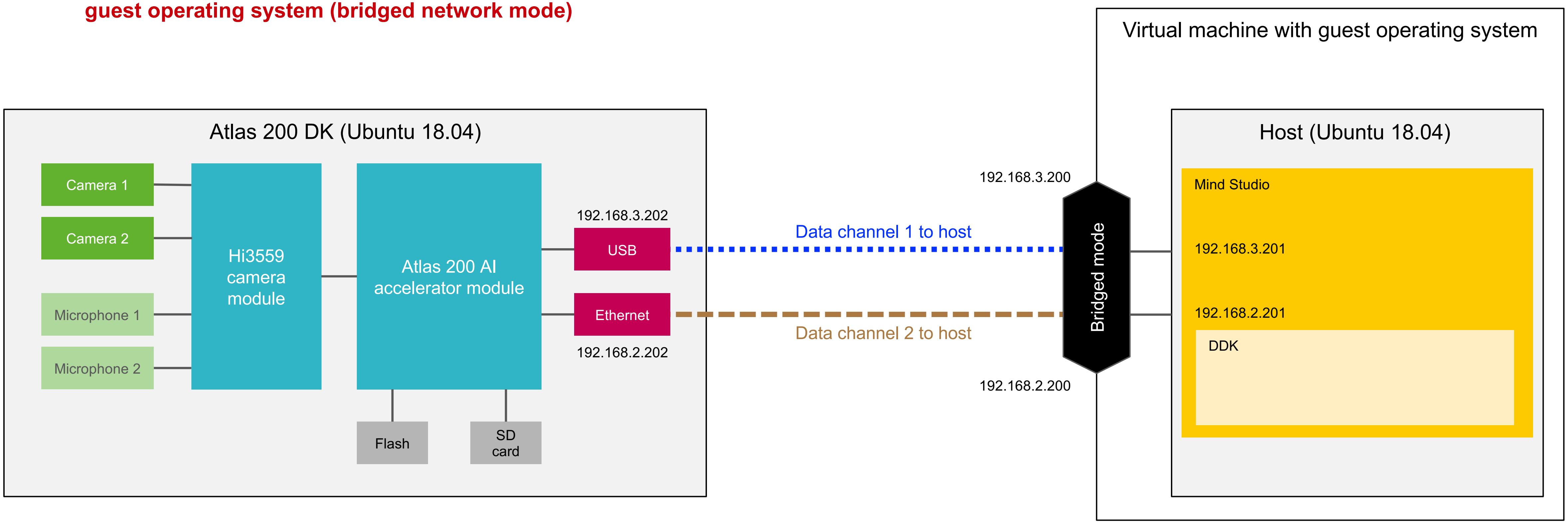


- **Virtual machine requirements**
  - Main memory ≧ 4 GB
  - Bridged network with default adapter
  - Hard disk ≧ 5 GB

- **Shared network** (NAT) is the default network mode for virtual machines.
  - The hardware virtualisation software creates a separate *virtual subnet* with its own virtual DHCP server running.
  - A virtual machine belongs to that *virtual subnet* with its own IP range.
  - A virtual machine is not visible in the real subnet the host system belongs to.
  - A virtual machine use full internet access.

Environment deployment /3

— Direct USB or Ethernet connection and virtual machine with guest operating system (bridged network mode)



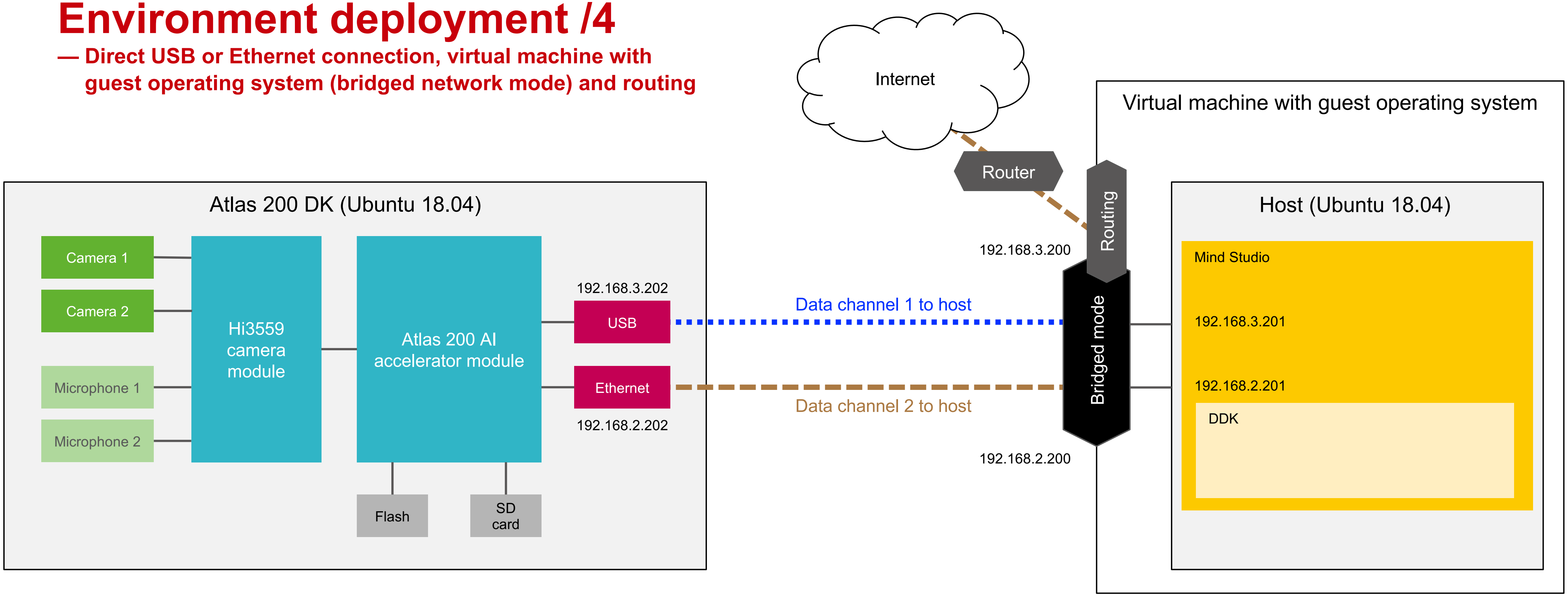
- **Virtual machine requirements**
  - Main memory ≧ 4 GB
  - Bridged network with default adapter
  - Hard disk ≧ 5 GB

- **Bridget network** is the recommended network mode and uses a virtualised network interface card with direct access to Internet.
  - A virtual machine appears as a separate computer that belongs to the same subnet as the host system.
  - A DHCP server (e.g. your router) provides a virtual machine with an IP address within the same IP range as other computer in the same subnet.
  - A virtual machine can ping and see all computers in the subnet.
  - Other computers can ping and see the virtual machine.

- **Shared network** (NAT) is the default network mode for virtual machines.
  - The hardware virtualisation software creates a separate *virtual subnet* with its own virtual DHCP server running.
  - A virtual machine belongs to that *virtual subnet* with its own IP range.
  - A virtual machine is not visible in the real subnet the host system belongs to.
  - A virtual machine use full internet access.

Environment deployment /4

— Direct USB or Ethernet connection, virtual machine with guest operating system (bridged network mode) and routing



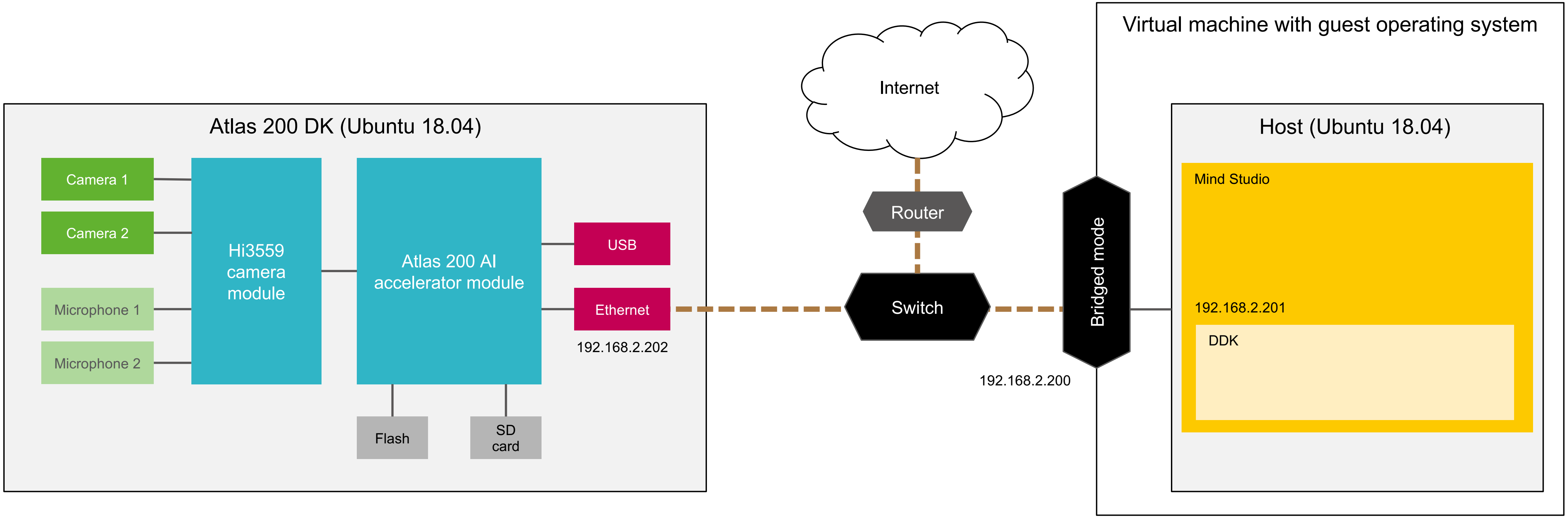
- **Virtual machine requirements**
  - Main memory ≧ 4 GB
  - Bridged network with default adapter
  - Hard disk ≧ 5 GB

- **Bridget network** is the recommended network mode and uses a virtualised network interface card with direct access to Internet.
  - A virtual machine appears as a separate computer that belongs to the same subnet as the host system.
  - A DHCP server (e.g. your router) provides a virtual machine with an IP address within the same IP range as other computer in the same subnet.
  - A virtual machine can ping and see all computers in the subnet.
  - Other computers can ping and see the virtual machine.

- **Shared network** (NAT) is the default network mode for virtual machines.
  - The hardware virtualisation software creates a separate *virtual subnet* with its own virtual DHCP server running.
  - A virtual machine belongs to that *virtual subnet* with its own IP range.
  - A virtual machine is not visible in the real subnet the host system belongs to.
  - A virtual machine use full internet access.

Environment deployment /5

— Ethernet connection, virtual machine with guest operating and Internet access (recommended)



- **Virtual machine requirements**
  - Main memory ≧ 4 GB
  - Bridged network with default adapter
  - Hard disk ≧ 5 GB

- **Bridget network** is the recommended network mode and uses a virtualised network interface card with direct access to Internet.
  - A virtual machine appears as a separate computer that belongs to the same subnet as the host system.
  - A DHCP server (e.g. your router) provides a virtual machine with an IP address within the same IP range as other computer in the same subnet.
  - A virtual machine can ping and see all computers in the subnet.
  - Other computers can ping and see the virtual machine.

- **Shared network** (NAT) is the default network mode for virtual machines.
  - The hardware virtualisation software creates a separate *virtual subnet* with its own virtual DHCP server running.
  - A virtual machine belongs to that *virtual subnet* with its own IP range.
  - A virtual machine is not visible in the real subnet the host system belongs to.
  - A virtual machine use full internet access.

About version 1.73.0.0

- **Atlas 200 DK developer board comes with three versions, including 1.3.0.0, 1.32.0.0, and 1.73.0.0. 1.73.0.0 is the latest version based on our new software architecture, facilitating developers to read, understand, and develop their artificial intelligence applications.**
- **This version allows multiple installation methods.**
  - Development environment: Install the toolkit packages for model conversion, source code creation and build.
  - Operating environment: Install device driver, OPP, and ACL library to run and built the source code or applications.

  Please note:

  - *If you install the development and operating environment on the same machine, both environments are co-deployed. If the installation is on different machines, the environments are separately deployed.*
  - *To use Mind Studio (only supported on Ubuntu for x86 architecture), use the separate deployment mode.*
- **The following describes the installation of the development and the operating environment via the separate deployment mode. The development environment is Ubuntu 18.04 (x86) in a virtual machine setup or dual-boot (it is recommended to have a clean operating system installation). The operating environment is Ubuntu 18.04 (ARM64) on the Atlas 200 DK developer board.**

Hardware and software requirements

- **Atlas 200 DK developer board, including network cable, power supply and SD card**
- **Ubuntu 18.04 in a virtual machine or dual-boot (preferably a newly created environment). It is recommended to use a virtual machine with more that 4 GB of memory.**
- **Two network ports that can be used to access the Internet (one for the virtual machine and the other for the developer board)**

Install environment dependencies /1

You will do the installation as a regular user. Please ensure that this regular user and the root user exist in the current environment.

1) Configure user permissions

Grant `sudo` permissions to a regular user for the toolkit installation.

Switch to the root user:

```
sudo bash
```

Grant the write permission on the `sudoers` file and open the file:

```
chmod u+w /etc/sudoers
vi /etc/sudoers
```

Add the following content in the marked box below **# User privilege specification**, as shown in the following figure:

Remove the write permission on the `/etc/sudoers` file:

```
chmod u-w /etc/sudoers
```

Install environment dependencies /2

2) **Install the related system dependencies and system components, which the toolkit packages requires**

```
sudo apt-get install -y gcc make cmake unzip zlib1g zlib1g-dev libsqlite3-dev openssl libssl-dev \
libffi-dev pciutils net-tools g++-5-aarch64-linux-gnu
```

3) **Compile and install Python**

Go to the home directory as regular user:

```
cd $HOME
```

Download the Python 3.7.5 source code package and decompress it:

```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
tar -zxvf Python-3.7.5.tgz
```

Go to the decompressed folder and run the following configuration, build, and installation commands:

```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-shared
make -j8
sudo make install
```

Install environment dependencies /3

Run the following commands to copy the .so files to the lib directory of the operating system and create Python soft links:

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

Install the Python dependency packages:

```
pip3.7.5 install attrs psutil decorator numpy protobuf==3.11.3 scipy sympy cffi grpcio \
grpcio-tools requests --user
```

Modify the PATH environment variable:

```
vim ~/.bashrc
```

Append the following line to the file:

```
export PATH=/usr/local/python3.7.5/bin/:$PATH
```

Run the following command for the environment variable to take effect:

```
source ~/.bashrc
```

Install the toolkit packages

Download and install the two toolkit packages which required by the development environment.

Possible download links are http://shrnk.cc/hbe41 or https://www.huaweicloud.com/ascend/resource/Software

The required packages are:

- `Ascend-Toolkit-20.0.RC1-arm64-linux_gcc7.3.0.run`
- `Ascend-Toolkit-20.0.RC1-x86_64-linux_gcc7.3.0.run`

Save the package in the `$HOME/Ascend` directory of your regular user in the development environment.

Install the toolkit packages:

```
chmod +x Ascend-Toolkit*.run
```

Install the toolkit packages:

```
cd $HOME/Ascend
./Ascend-Toolkit-20.0.RC1-arm64-linux_gcc7.3.0.run --install
./Ascend-Toolkit-20.0.RC1-x86_64-linux_gcc7.3.0.run --install
```

Install the media module device driver

You need to install header and library files if you use an external camera to collect source data for the development of an artificial intelligence application.

Possible download links are http://shrnk.cc/hbe41 or https://www.huaweicloud.com/ascend/resource/Software

The required package is:

- `Ascend310-driver-1.73.5.1.b050-ubuntu18.04.aarch64-minirc.tar.gz`

Save the package in the `$HOME/Ascend` directory of your regular user in the development environment.

Install the device driver:

```
chmod +x Ascend-Toolkit*.run
```

Install the toolkit packages:

```
cd $HOME/Ascend
tar -zxvf Ascend310-driver-1.73.5.1.b050-ubuntu18.04.aarch64-minirc.tar.gz
```

```
ascend@ubuntu:~/Ascend$ ll
total 80288
drwxrwxr-x  4 ascend ascend     4096 Jul 27 05:19 ./
drwxr-xr-x 38 ascend ascend     4096 Jul 28 04:53 ../
-rwxrw-rw-  1 ascend ascend 82198281 Jul 15 04:41 Ascend310-driver-1.73.5.1.b050
-ubuntu18.04.aarch64-minirc.tar.gz*
drwxrwxr-x  3 ascend ascend     4096 Jul 22 18:31 ascend-toolkit/
drwxr-xr-x  4 ascend ascend     4096 Jun 15 08:30 driver/
ascend@ubuntu:~/Ascend$
```

Install Mind Studio /1

Download the Mind Studio version 2.3.3, as shown in the following figure.

Possible download links are http://shrnk.cc/hbe41 or https://www.huaweicloud.com/ascend/resources/Tools/0

The required package is:

- `mindstudio.tar.gz`

Save the package in the $HOME directory of your regular user in the development environment.

```
cd $HOME
```

Install required dependencies:

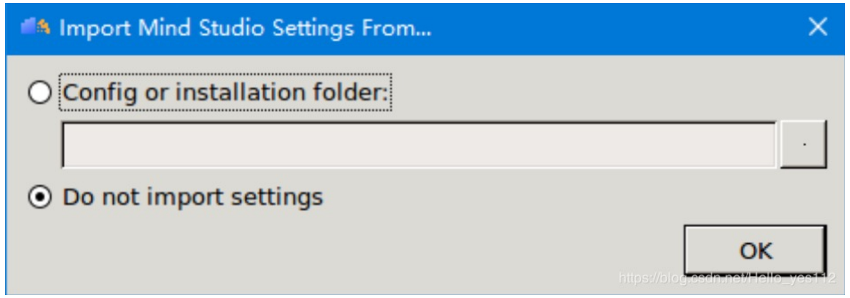
```
sudo apt-get -y install xterm openjdk-8-jdk fonts-wqy-zenhei fonts-wqy-microhei fonts-arphic-ukai \
fonts-arphic-uming
sudo /usr/local/python3.7.5/bin/pip3 install --user coverage gnureadline pylint matplotlib PyQt5==5.14.0
sudo apt install libcanberra-gtk-module libcanberra-gtk3-module
```

Decompress the file and run Mind Studio:

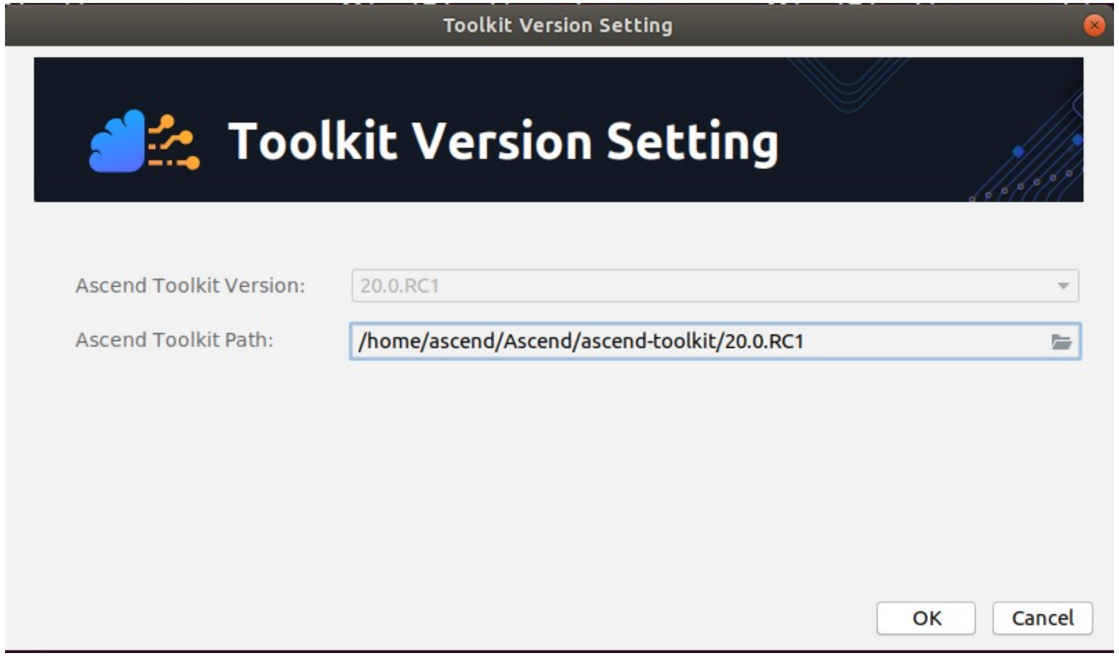
```
tar -zxvf mindstudio.tar.gz
cd MindStudio-ubuntu/bin
./Mindstudio.sh
```

Install Mind Studio /2

In the following dialog box, select **Do not import settings**.



Select the Toolkit path (`/home/ascend/Ascend/ascend-toolkit/20.0.RC1` is used as an example):



Create and write SD card image

Setting up the operating environment /1

For the set up of the operating environment, you need to write the runtime code and system programs to the SD card. Afterwards, you can insert the SD card in the Atlas 200 DK developer board.

1) Download and install the software packages which which are required to write the SD card.

Possible download links are http://shrnk.cc/hbe41 or https://www.huaweicloud.com/ascend/resource/Software

http://cdimage.ubuntu.com/ubuntu/releases/18.04/release/

The required packages are:

- `Ascend310-driver-1.73.5.1.b050-ubuntu18.04.aarch64-minirc.tar.gz`
- `Ascend310-aicpu_kernels-1.73.5.1.b050-minirc.tar.gz`
- `Ascend-acllib-1.73.5.1.b050-ubuntu18.04.aarch64-minirc.run`
- `ubuntu-18.04.5-server-arm64.iso`

Setting up the operating environment /2

2) Download the card making script

Run the following command the `$HOME` directory of your development environment for your regular user. This will download the program from the **ascend-tools** repository:

```
git clone https://gitlab.schihei.de/schihei/ascend-tools.git
```

or

```
git clone https://gitee.com/ascend/tools.git ascend-tools
```

Go to the card making directory.

```
cd $HOME/ascend-tools/makesd/for_1.7x.0.0/
```

Copy the downloaded files into this directory and verify with ls if all files are available.

```
~/ascend-tools/makesd/for_1.7x.0.0$ ls
Ascend310-aicpu_kernels-1.73.5.1.b050-minirc.tar.gz                  make_sd_card.py      README.md
Ascend310-driver-1.73.5.1.b050-ubuntu18.04.aarch64-minirc.tar.gz     make_ubuntu_sd.sh    sd_card_making_log
Ascend-acllib-1.73.5.1.b050-ubuntu18.04.aarch64-minirc.run           README_EN.md         ubuntu-18.04.5-server-arm64.iso
```

3) Install required Python packages

```
pip3 install pyyaml
```

4) Install system dependencies

```
sudo apt-get install qemu-user-static binfmt-support python3-yaml gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
```

Setting up the operating environment /3

5) Configure static IP address for Atlas 200 DK developer board

Open the file `make_sd_card.py` and change the `NETWORK_CARD_DEFAULT_IP` variable according to your network configuration. This variable will set the static IP address, which the developer board will use.

```
vim make_sd_card.py
```

```
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
# ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.
#   =======================================================================
#
import os
import platform
import signal
import subprocess
import time
import yaml
import sys


NETWORK_CARD_DEFAULT_IP="192.168.2.205"
USB_CARD_DEFAULT_IP="192.168.1.2"
                                                        38,1          4%
```

<u>Note:</u> It is possible to add later a nameserver entry or can configure to use dynamic IP addresses via DHCP. To do this please edit `/etc/netplan/01-netcfg.yaml` and execute `netplan apply`.

Setting up the operating environment /4

6) Connect the card reader and make a bootable SD card

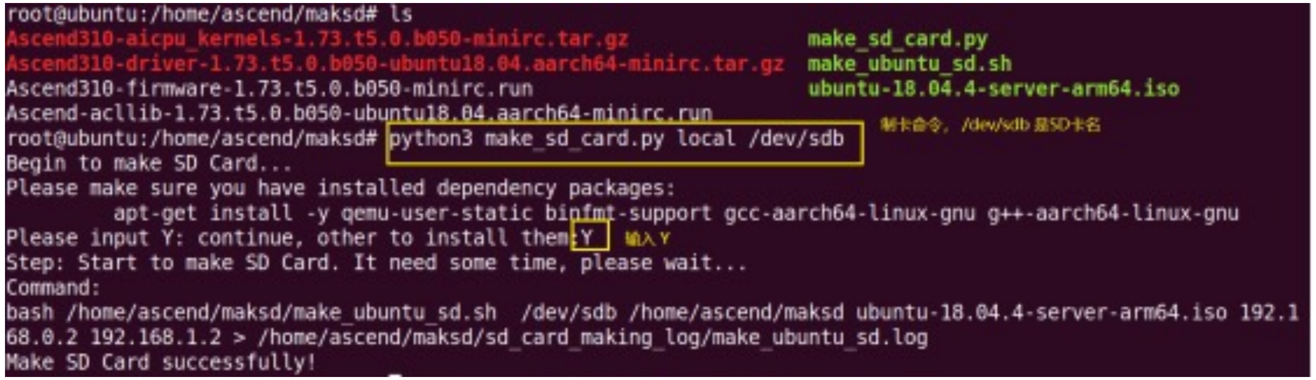
Switch to the root user and prepare for card making:

`sudo bash`

Execute the script to prepare for card making:

`python3 make_sd_card.py local /dev/sdb`

Note: `/dev/sdb` indicates the device name of the SD card. You can run the `fdisk -l` command as the root user to view the device name.



When a message is displayed, asking you whether to continue the installation, enter `Y`. Wait for about 7 minutes. The message **Make SD Card successfully!** is displayed, indicating that the SD card has been made successfully.

Boot and connect to the Atlas 200 DK developer board

Power on the Atlas 200 DK developer board

1) **Power on the Atlas 200 DK**

Insert the prepared card to the Atlas 200 DK developer board, power on the board, and connect it to the network.

2) **Login to the developer board**

Run the `ssh` command as a common user to log in to the developer board (password: **`Mind@123`**):

`ssh HwHiAiUser@<YOUR-IP-ADDRESS>` e.g. 192.168.2.205

3) **Restart ada**

As new environment variables are set, the `ada` tool in the operating environment needs to be restarted. Otherwise, the development environment cannot access the newly set environment variables in the operating environment.

Run the following commands as a common user to view the process ID of the ada tool:

```
HwHiAiUser@davinci-mini:~$ ps -ef | grep ada
HwHiAiU+  1996      1  0 08:40 ?        00:00:00 /var/ada
HwHiAiU+  2060   2047  0 08:41 pts/0    00:00:00 grep --color=auto ada
```

Since the process ID of the ada tool is 1996, run the following command to kill the ada process:

```
kill -9 1996
```

Run the following commands as a common user to restart ada:

```
HwHiAiUser@davinci-mini:/var$ cd /var/
HwHiAiUser@davinci-mini:/var$ ./ada &
```

Install third-party packages

Installation of additional packages /1

1) **Install FFmpeg**

Switch to the root user (password: **Mind@123**):

```
su root
```

Install the related system software packages. As a root user, install the following software packages:

```
apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff5-dev git cmake
```

Exit the root user and switch to a regular user:

```
exit
```

Create a folder to store the built files:

```
mkdir -p /home/HwHiAiUser/ascend_ddk/arm
```

Download FFmpeg source code to the `$HOME` directory of your regular user, decompress it, go to the directory, compile and install it:

```
cd $HOME
wget http://www.ffmpeg.org/releases/ffmpeg-4.1.3.tar.gz
tar -zxvf ffmpeg-4.1.3.tar.gz
cd ffmpeg-4.1.3
./configure --enable-shared --enable-pic --enable-static --disable-yasm --prefix=/home/HwHiAiUser/ascend_ddk/arm
make -j8
make install
```

Installation of additional packages /2

Switch to the root user and add the FFmpeg library:

```
su root
echo "/home/HwHiAiUser/ascend_ddk/arm/lib" > /etc/ld.so.conf.d/ffmpeg.conf
ldconfig
```

Add the binaries to the PATH environment variable:

```
echo "export PATH=$PATH:/home/HwHiAiUser/ascend_ddk/arm/bin" >> /etc/profile
source /etc/profile
```

Copy all files in the `./lib/pkgconfig` directory in the FFmpeg installation directory to the related directory of the operating system. The installation of OpenCV depends on these files:

```
cp /home/HwHiAiUser/ascend_ddk/arm/lib/pkgconfig/* /usr/share/pkgconfig
```

Exit the root user and switch to a regular user:

```
exit
```

Installation of additional packages /3

2) Install OpenCV

Switch to the root user (password: **Mind@123**):

```
su root
```

Install the related system software packages. As a root user, install the following software packages:

```
apt install python-dev python3-dev
```

Run the following commands to go to the $HOME directory of a regular user, download OpenCV, and create a build directory:

```
cd $HOME
git clone -b 4.3.0 https://gitee.com/mirrors/opencv.git
cd opencv
mkdir build
cd build
```

Build OpenCV and install OpenCV:

```
cmake ../ -DBUILD_SHARED_LIBS=ON -DBUILD_TESTS=OFF -DCMAKE_BUILD_TYPE=RELEASE \
-DCMAKE_INSTALL_PREFIX=/home/HwHiAiUser/ascend_ddk/arm
make -j8
make install
```

Installation of additional packages /4

Modify the `LD_LIBRARY_PATH` environment variable.

Open the `.bashrc` configuration file as a regular user:

```
vi ~/.bashrc
```

Append the following line to the file:

```
export LD_LIBRARY_PATH=/home/HwHiAiUser/Ascend/acllib/lib64:/home/HwHiAiUser/ascend_ddk/arm/lib
```

Restart ada.

Installation of additional packages /5

3) Synchronize the FFmpeg and OpenCV libraries with the development environment

Note: The development environment refers to the local environment where Mind Studio is installed. The regular user `ascend` is used as an example.

As a regular user in the development environment, import the third-party libraries from the operating environment to the development environment for the build process. Run the following command to create a directory for storing the third-party libraries:

```
mkdir $HOME/ascend_ddk
```

Copy the libraries to the development environment:

```
scp -r HwHiAiUser@192.168.1.2:/home/HwHiAiUser/ascend_ddk/arm $HOME/ascend_ddk/
```

Switch to the root user and copy the libraries required for the build process from the operating environment to a directory in the development environment:

```
su root
cd /usr/lib/aarch64-linux-gnu
scp -r HwHiAiUser@192.168.1.2:/lib/aarch64-linux-gnu/* ./
scp -r HwHiAiUser@192.168.1.2:/usr/lib/aarch64-linux-gnu/* ./
```

Switch to a regular user:

```
exit
```

Installation of additional packages /6

4) Install Python 3 in the operating environment

Configure user permissions:

```
su root
```

Grant the write permission on the `sudoers` file and open the file:

```
chmod u+w /etc/sudoers
vi /etc/sudoers
```

Add the following content below **# User privilege specification** in the `sudoers` file:

```
# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
HwHiAiUser ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

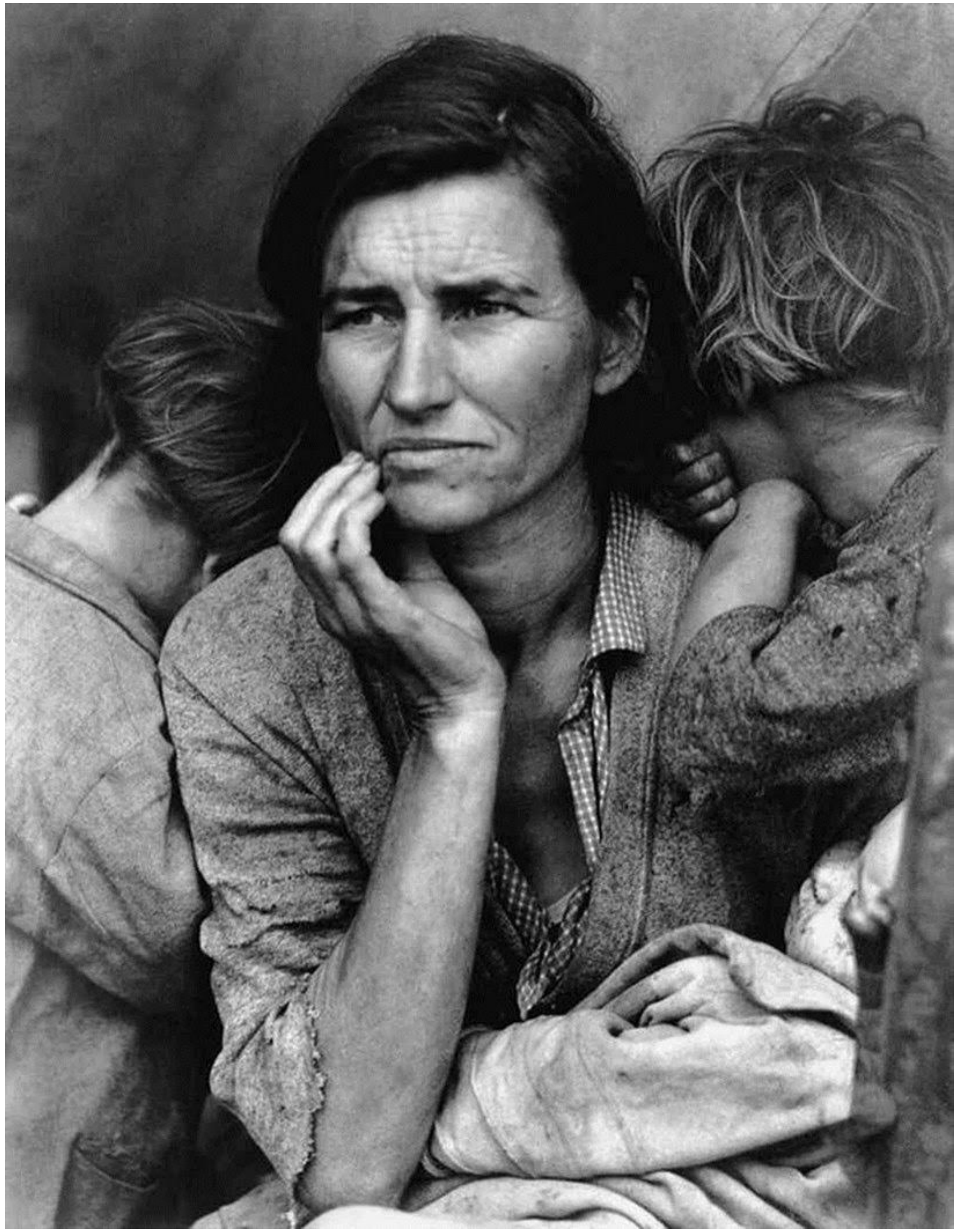
Install Python packages:

```
sudo apt-get install python3-setuptools
sudo python3 -m easy_install install pip
sudo apt-get install libtiff5-dev libjpeg8-dev zlib1g-dev libfreetype6-dev liblcms2-dev libwebp-dev tcl8.6-dev tk8.6-dev python-tk
pip3 install pillow --user
pip3 install Cython
pip3 install numpy --user
```

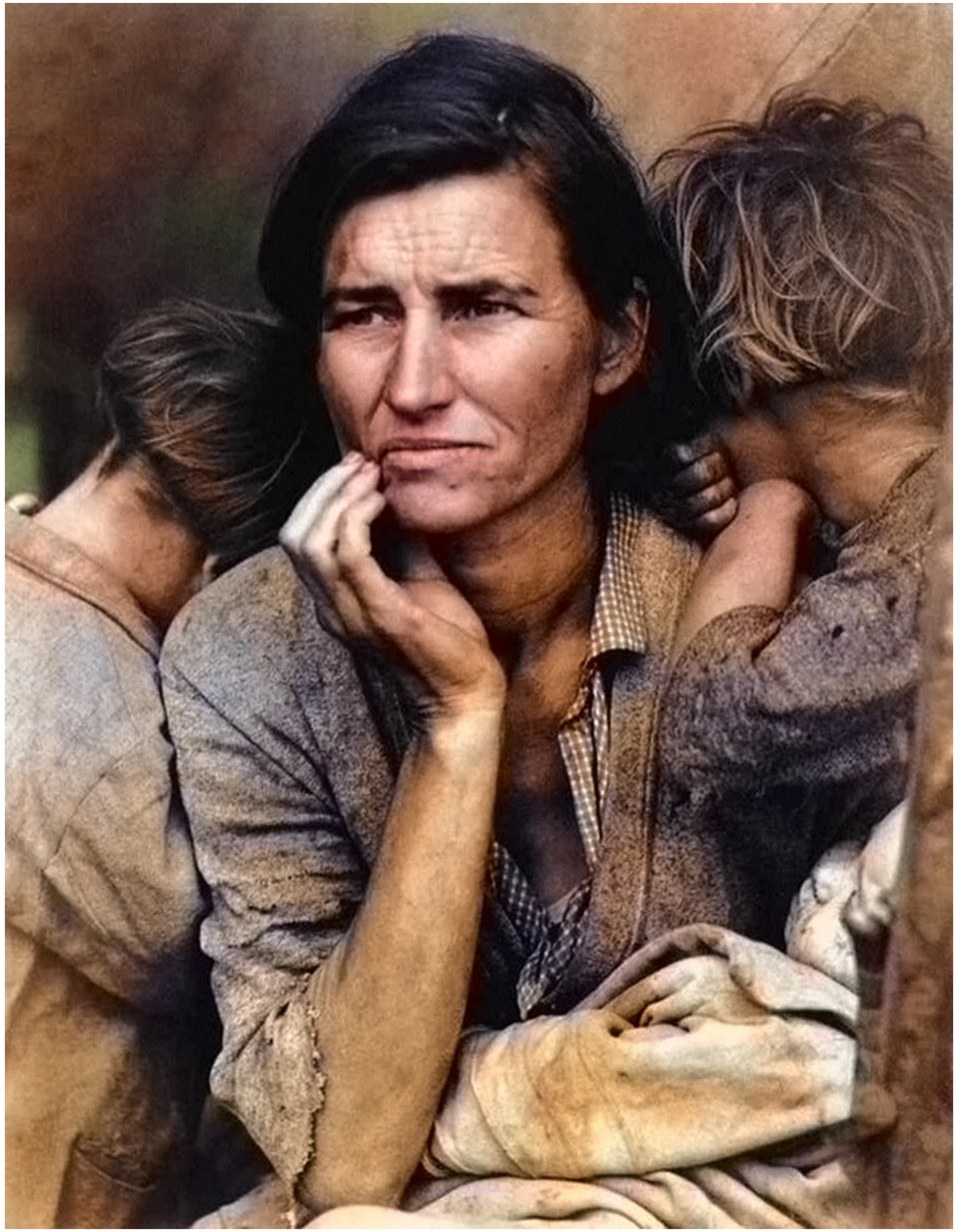
Colourful Image Colourisation [Zhang et al., 2016]

Source: Ansel Adams, Yosemite Valley Bridge

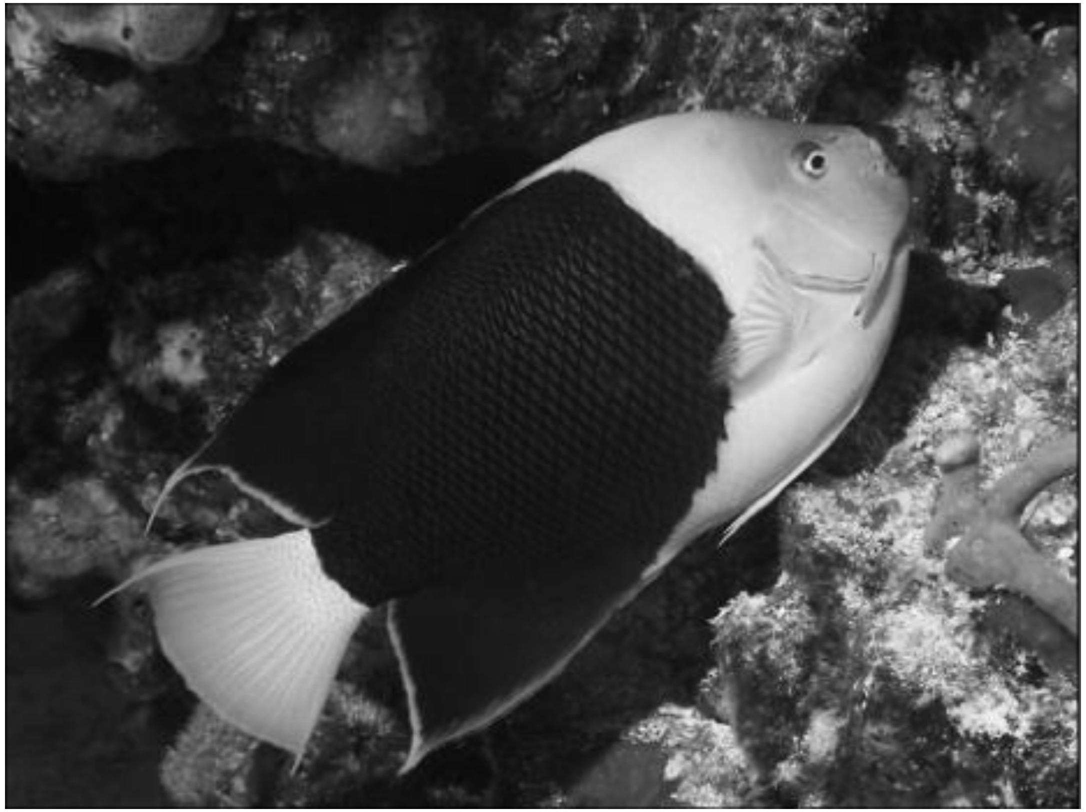
Source: Ansel Adams, Yosemite Valley Bridge



Source: Dorothea Lange, Migrant Mother, 1936



Source: Dorothea Lange, Migrant Mother, 1936



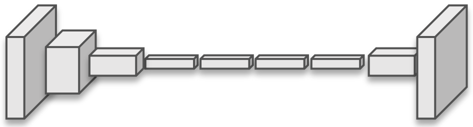
$\mathcal{F}$

Grayscale image: *L* channel

$X \in \mathbb{R}^{H \times W \times 1}$

Color information: *ab* channels

$\hat{Y} \in \mathbb{R}^{H \times W \times 2}$

L →  → ***ab***

[Zhang et al., 2016]

$\mathcal{F}$

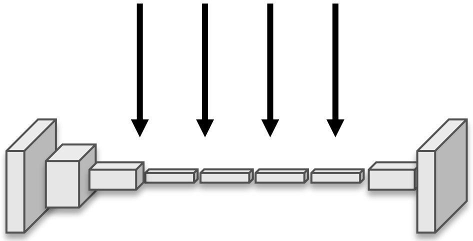
Semantics? Higher-level abstraction?

Grayscale image: *L* cha

$\mathrm{X} \in \mathbb{R}^{H \times W \times 1}$

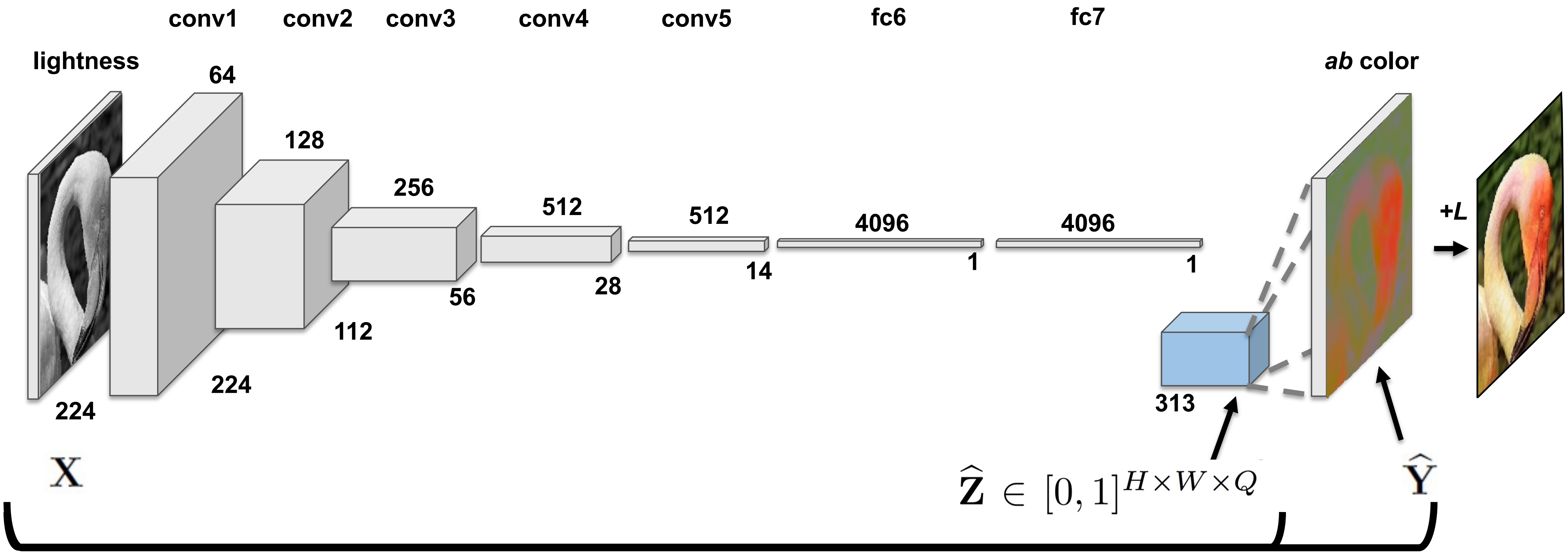
oncatenate (L, ab)

$(\mathrm{X}, \hat{\mathrm{Y}})$

L →  → ***ab*** ← "Free" supervisory signal

[Zhang et al., 2016]

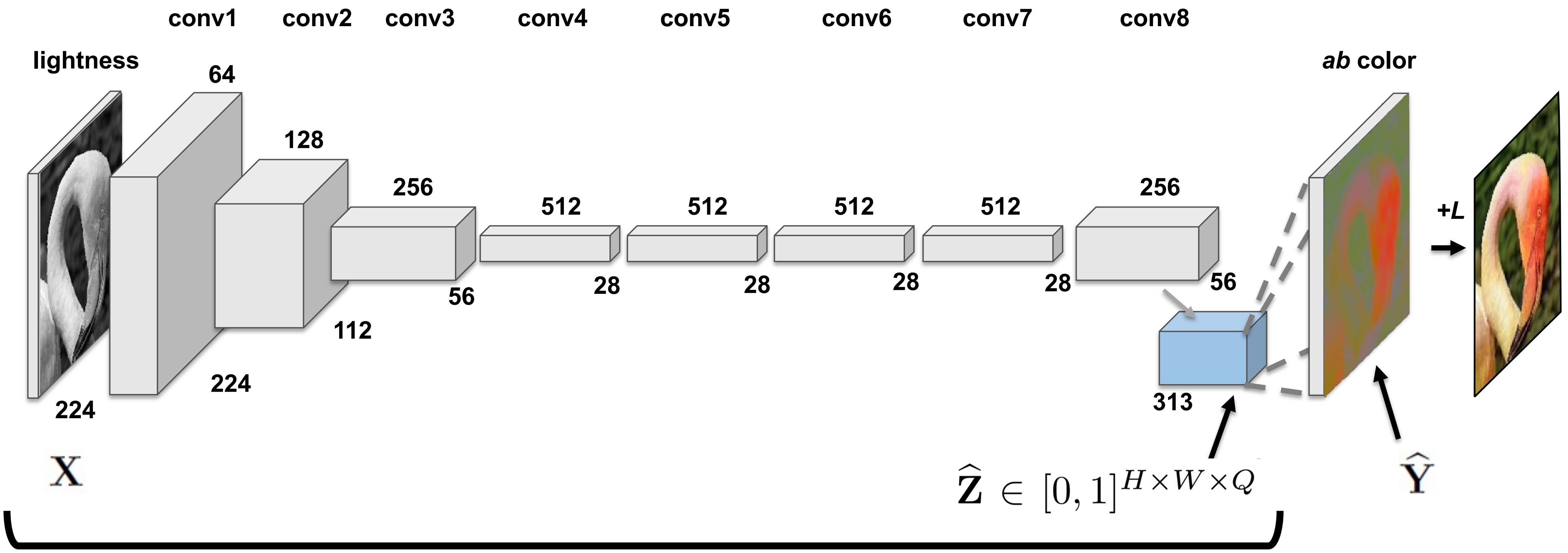
Network architecture [Zhang et al., 2016]



$$\hat{Y} = \mathcal{F}(X)$$

$$X \in \mathbb{R}^{H \times W \times 1}$$

Network architecture [Zhang et al., 2016]



$$\hat{Z} = \mathcal{G}(X)$$

$$\hat{Y} = \mathcal{H}(\hat{Z})$$

Model inference architecture

1. Convert Caffe model to offline model
2. ACL Init → select device → create contexts / streams
3. Load model → allocate input/output buffers + data descriptors
4. Input data pre-processing
5. Transfer input data to device
6. Model inference `aclmdlExecute()`
7. Copy back output data to the host
8. Output data post-processing
9. Cleanup and ACL Finalize

ATC (step 1) — ACL (steps 2–3, 5–7, 9) — OpenCV (steps 4, 8)

Model conversion

1. Convert Caffe model to offline model

Model conversion using Ascend Tensor Compiler (ATC)

```
atc --model=colorization.prototxt    \
--weight=colorization.caffemodel     \
--framework=0 --output=colorization \
--soc_version=Ascend310
```

ATC ACL OpenCV

Inference implementation with ACL /1

2. ACL Init → select device → create contexts / streams

```
aclInit("./acl.json");

aclrtSetDevice(deviceId);

aclrtContext context;
aclrtCreateContext(&context, deviceId);

aclrtStream stream;
aclrtCreateStream(&stream);
```

ATC ACL OpenCV

Inference implementation with ACL /2

3. Load model → allocate input/output buffers + data descriptors

```
// Load model
aclmdlQuerySize(modelPath, &modelMemSize_, &modelWeightSize_);

aclrtMalloc(&modelMemPtr_, modelMemSize_, ACL_MEM_MALLOC_HUGE_FIRST);

aclrtMalloc(&modelWeightPtr_, modelWeightSize_, ACL_MEM_MALLOC_HUGE_FIRST);
aclmdlLoadFromFileWithMem(modelPath, &modelId_, modelMemPtr_,
                          modelMemSize_, modelWeightPtr_, modelWeightSize_);

// Create model descriptor
modelDesc_ = aclmdlCreateDesc();
aclmdlGetDesc(modelDesc_, modelId_);

// Create input
input_ = aclmdlCreateDataset();
aclrtMalloc(&inputDataBuffer, (size_t)(inputDataSize_), ACL_MEM_MALLOC_HUGE_FIRST);
aclDataBuffer *inputData = aclCreateDataBuffer(inputDataBuffer, bufferSize);
aclmdlAddDatasetBuffer(input_, inputData);

// Create output
output_ = aclmdlCreateDataset();
size_t outputSize = aclmdlGetNumOutputs(modelDesc_);
for (size_t i = 0; i < outputSize; ++i)
{
    size_t buffer_size = aclmdlGetOutputSizeByIndex(modelDesc_, i);

    void *outputBuffer = nullptr;
    aclrtMalloc(&outputBuffer, buffer_size, ACL_MEM_MALLOC_NORMAL_ONLY);

    aclDataBuffer *outputData = aclCreateDataBuffer(outputBuffer, buffer_size);
    aclmdlAddDatasetBuffer(output_, outputData);
}
```

ATC ACL OpenCV

Inference implementation with ACL /3

- Image reading
- Resizing

→

- Normalization
- CSC
- BGR2Lab conversion

→

- Extract L channel
- Mean subtraction

4. Input data pre-processing

```
cv::Mat mat = cv::imread(imageFile, CV_LOAD_IMAGE_COLOR);
//resize
cv::Mat reiszeMat;
cv::resize(mat, reiszeMat, cv::Size(224, 224));

// deal image
reiszeMat.convertTo(reiszeMat, CV_32FC3);
reiszeMat = 1.0 * reiszeMat / 255;
cv::cvtColor(reiszeMat, reiszeMat, CV_BGR2Lab);

// pull out L channel and subtract 50 for mean-centering
std::vector<cv::Mat> channels;
cv::split(reiszeMat, channels);
cv::Mat reiszeMatL = channels[0] - 50;
```

ATC ACL OpenCV

Inference implementation with ACL /4

5. Transfer input data to device
6. Model inference `aclmdlExecute()`
7. Copy back output data to the host

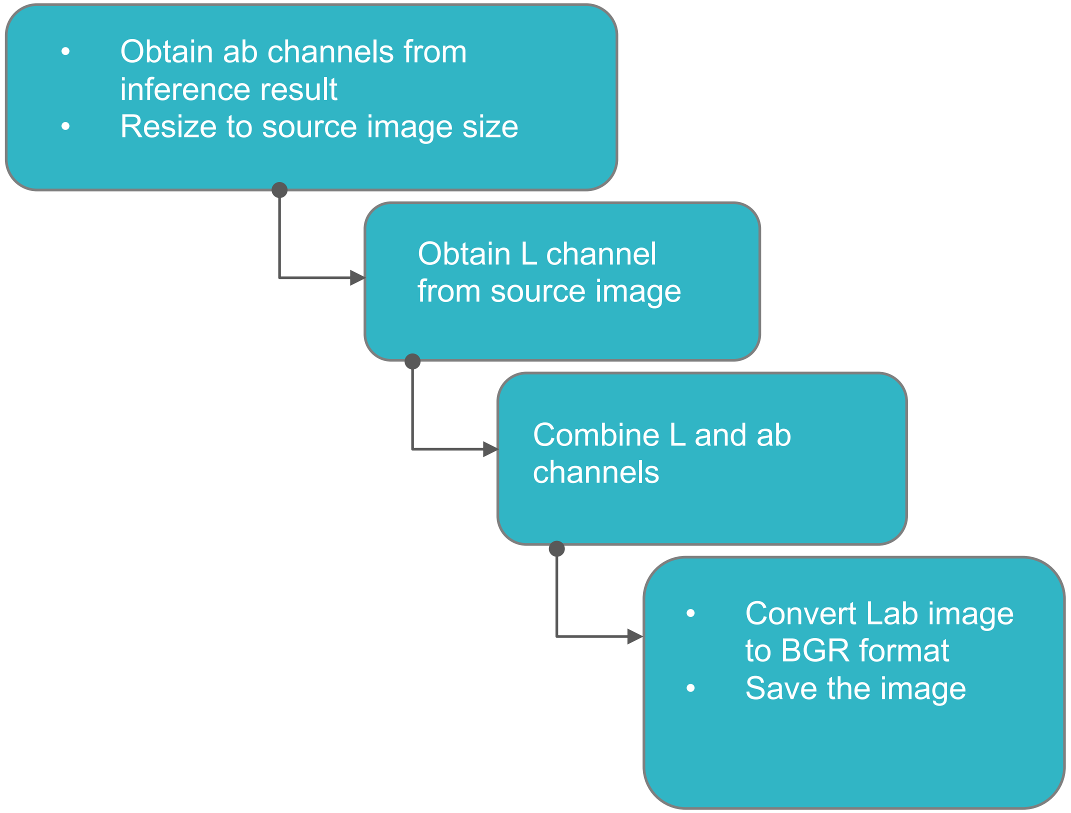
```
// Copy-in inputs
aclrtMemcpy(inputBuf_, inputDataSize_,
            reiszeMatL.ptr<uint8_t>(), inputDataSize_,
            ACL_MEMCPY_HOST_TO_DEVICE);

// Model inference
aclmdlExecute(modelId_, input_, output_);

// Copy-out results
aclDataBuffer *dataBuffer = aclmdlGetDatasetBuffer(output_, 0);
void *dataBufferDev = aclGetDataBufferAddr(dataBuffer);
size_t bufferSize = aclGetDataBufferSize(dataBuffer);
aclrtMemcpy(hostPtr, bufferSize,
            dataBufferDev, bufferSize,
            ACL_MEMCPY_DEVICE_TO_HOST);
```

ATC ACL OpenCV

Inference implementation with ACL /6



8. Output data post-processing

```
uint32_t size = bufferSize / sizeof(float);

// get a channel and b channel result data
cv::Mat mat_a(56, 56, CV_32FC1, const_cast<float *>((float *)hostPtr));
cv::Mat mat_b(56, 56, CV_32FC1, const_cast<float *>((float *)hostPtr + size / 2));

// pull out L channel in original image
cv::Mat mat = cv::imread(imageFile, CV_LOAD_IMAGE_COLOR);
mat.convertTo(mat, CV_32FC3);
mat = 1.0 * mat / 255;
cv::cvtColor(mat, mat, CV_BGR2Lab);
std::vector<cv::Mat> channels;
cv::split(mat, channels);

// resize to match size of original image L
int r = mat.rows;
int c = mat.cols;
cv::Mat mat_a_up(r, c, CV_32FC1);
cv::Mat mat_b_up(r, c, CV_32FC1);
cv::resize(mat_a, mat_a_up, cv::Size(c, r));
cv::resize(mat_b, mat_b_up, cv::Size(c, r));

// result Lab image
cv::Mat newChannels[3] = {channels[0], mat_a_up, mat_b_up};
cv::Mat resultImage;
cv::merge(newChannels, 3, resultImage);

//convert back to rgb
cv::cvtColor(resultImage, resultImage, CV_Lab2BGR);
resultImage = resultImage * 255;
SaveImage(imageFile, resultImage);
```

ATC ACL OpenCV

Inference implementation with ACL /7

```
// Free acquired resources
aclmdlUnload(modelId_);
aclmdlDestroyDesc(modelDesc_);
modelDesc_ = nullptr;

aclrtFree(modelMemPtr_);
modelMemPtr_ = nullptr;
modelMemSize_ = 0;

aclrtFree(modelWeightPtr_);
modelWeightPtr_ = nullptr;
modelWeightSize_ = 0;

aclmdlDestroyDesc(modelDesc_);
modelDesc_ = nullptr;

for (size_t i = 0; i < aclmdlGetDatasetNumBuffers(input_); ++i)
{
    aclDataBuffer *dataBuffer = aclmdlGetDatasetBuffer(input_, i);
    aclDestroyDataBuffer(dataBuffer);
}
aclmdlDestroyDataset(input_);
input_ = nullptr;

for (size_t i = 0; i < aclmdlGetDatasetNumBuffers(output_); ++i)
{
    aclDataBuffer *dataBuffer = aclmdlGetDatasetBuffer(output_, i);
    void *data = aclGetDataBufferAddr(dataBuffer);
    aclrtFree(data);
    aclDestroyDataBuffer(dataBuffer);
}
aclmdlDestroyDataset(output_);
output_ = nullptr;

// ACL fini
aclrtResetDevice(deviceId_);
aclFinalize();
```

9. Cleanup and ACL Finalize

ATC ACL OpenCV

Download project source code

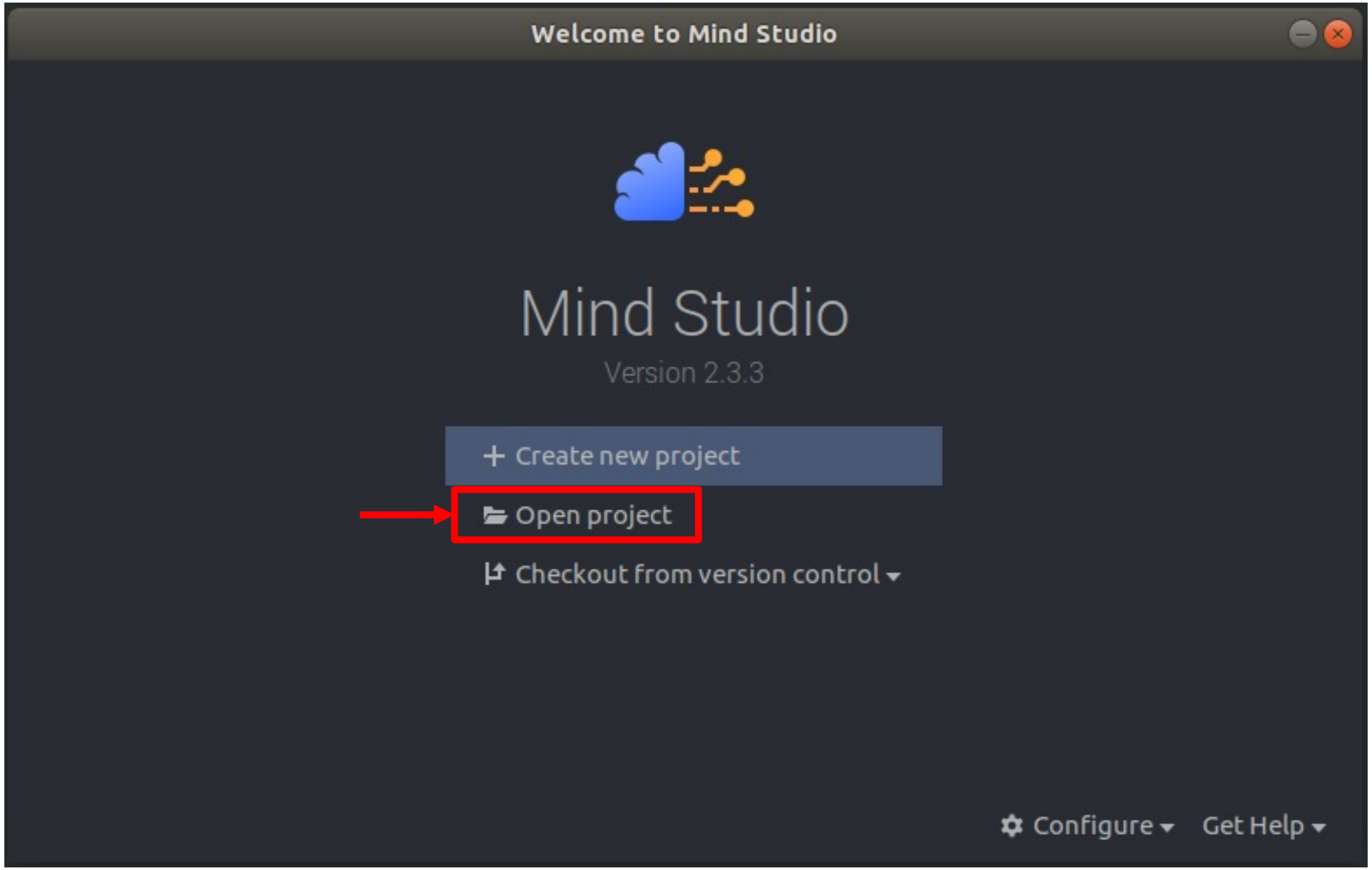
```
mkdir -p $HOME/AscendProjects
cd $HOME/AscendProjects
```

```
# Obtain the colorization project package.
git clone https://gitlab.schihei.de/schihei/sample-colorization.git
```

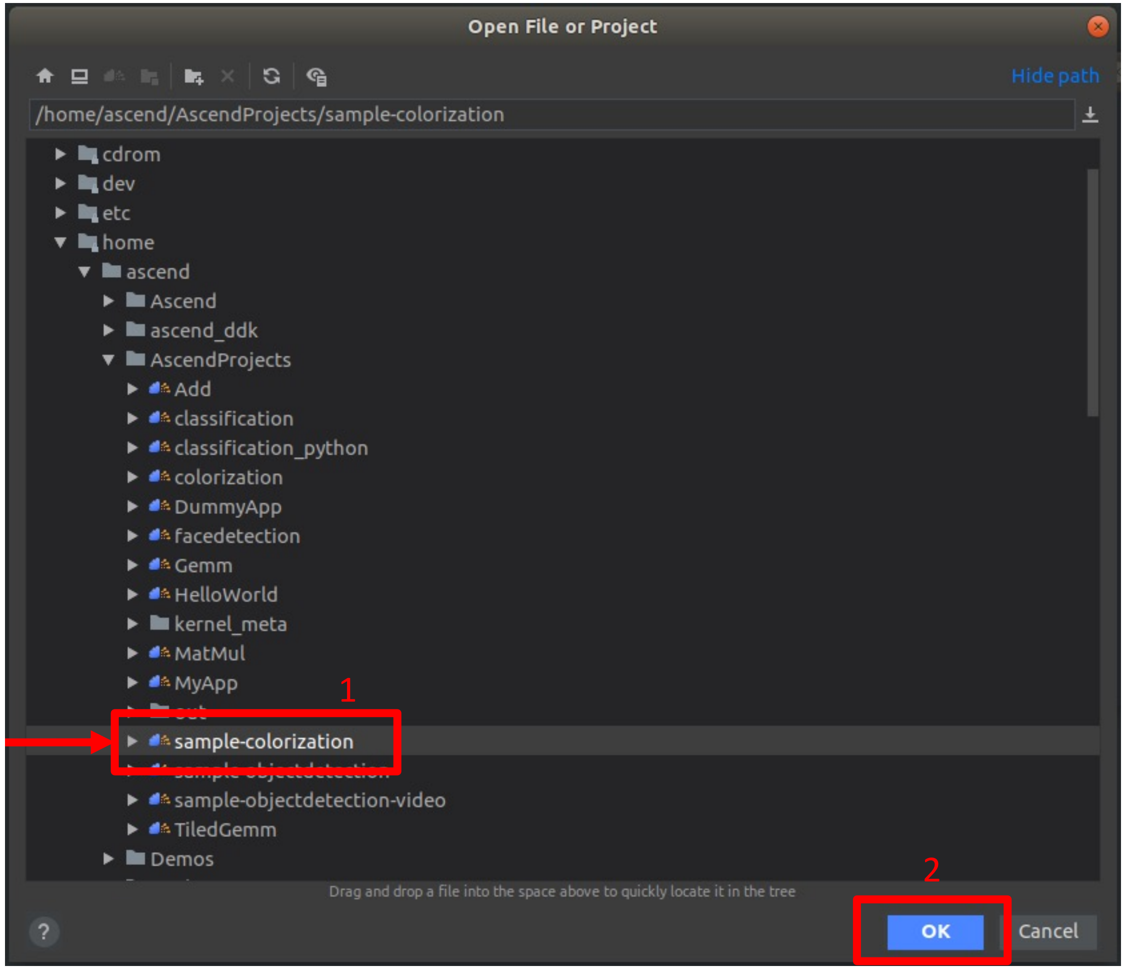
The repository includes the original Caffe model that will be converted to offline model adapted to Ascend platforms.

Loading the project /1

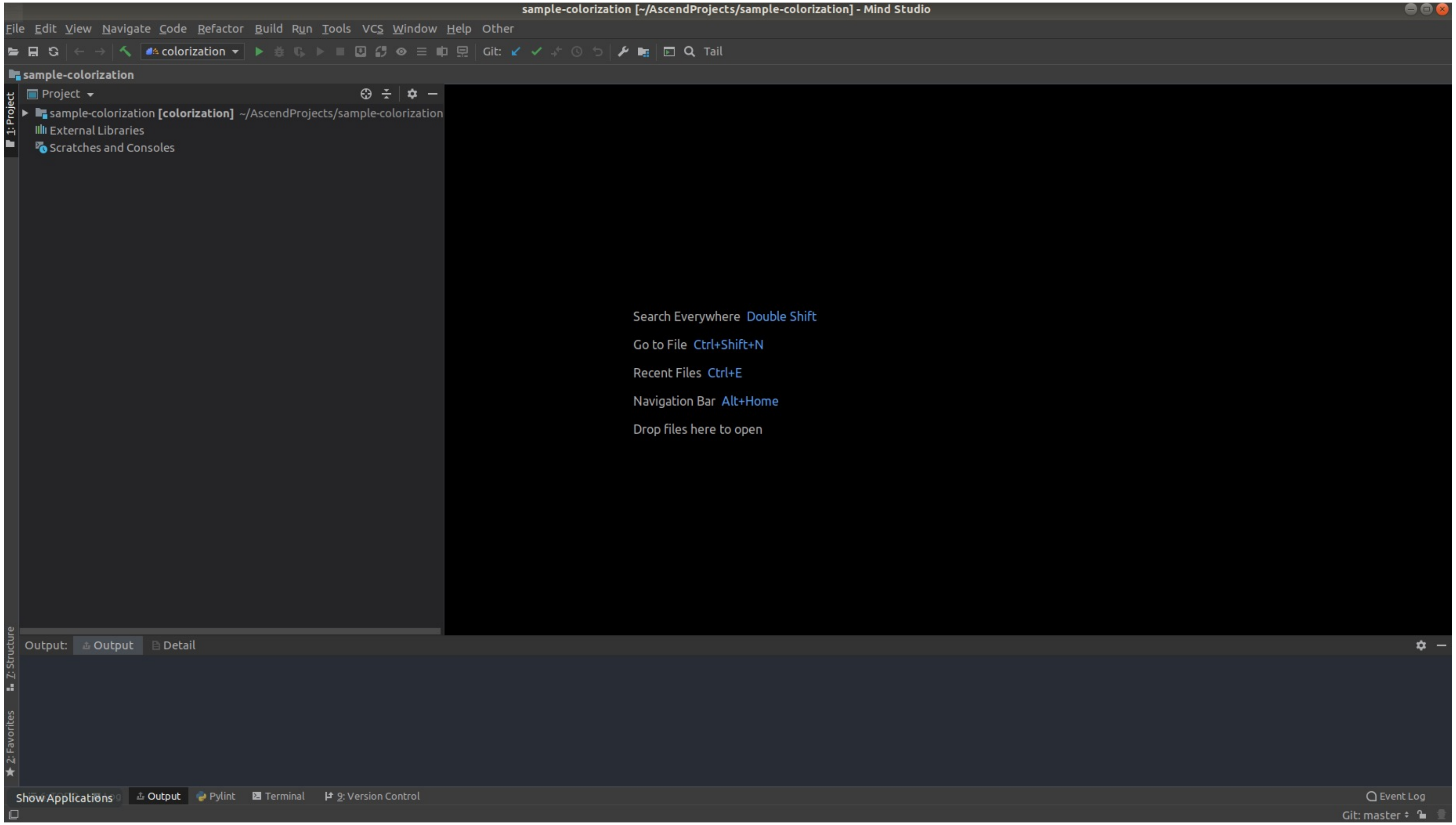
```
cd $HOME/MindStudio-ubuntu/bin && ./MindStudio.sh &
```



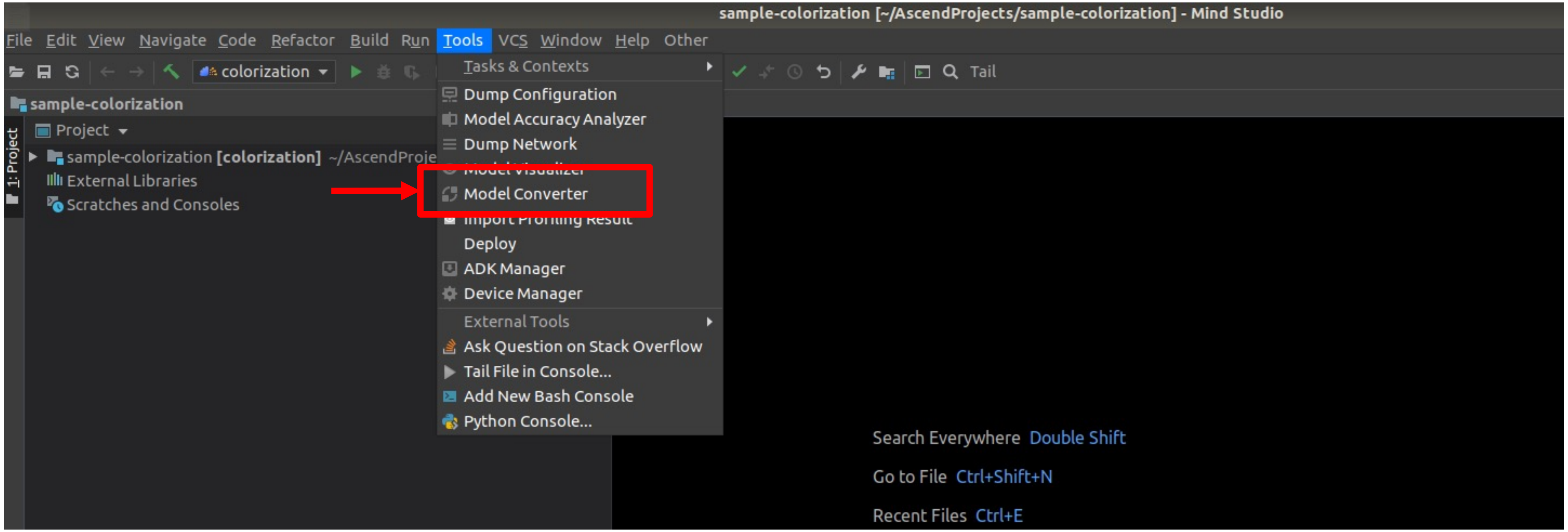
Loading the project /2



Loading the project /3

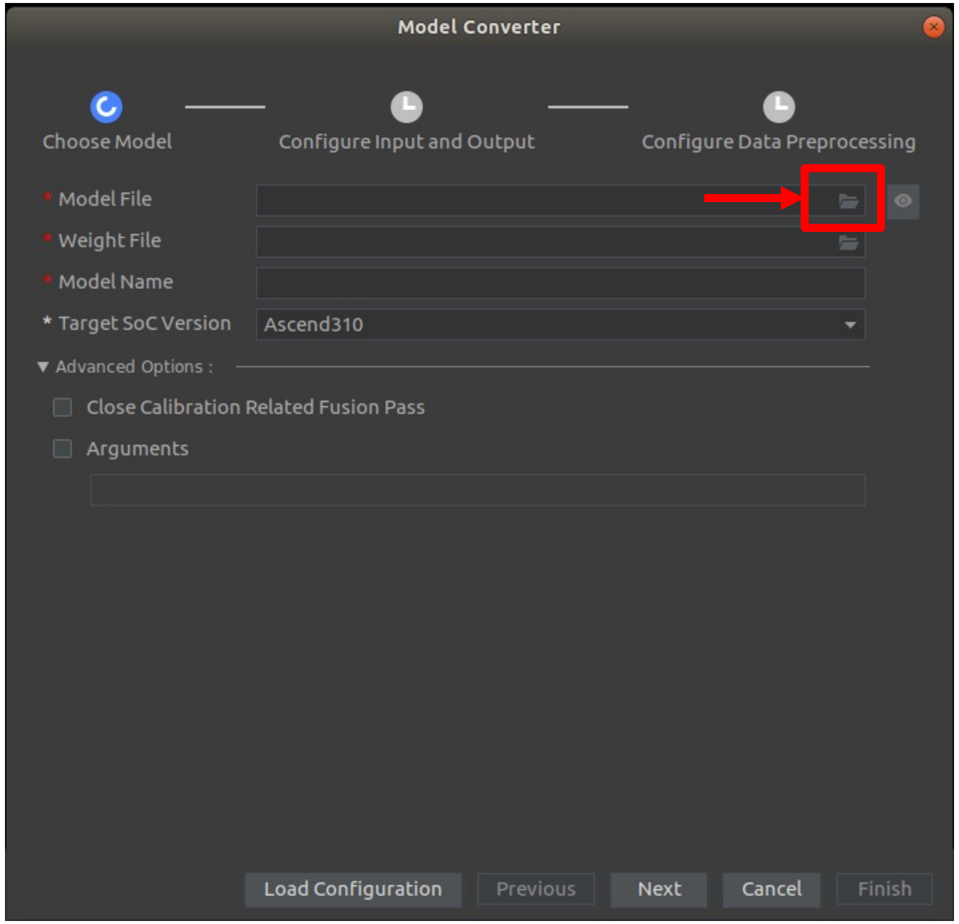


Model conversion /1



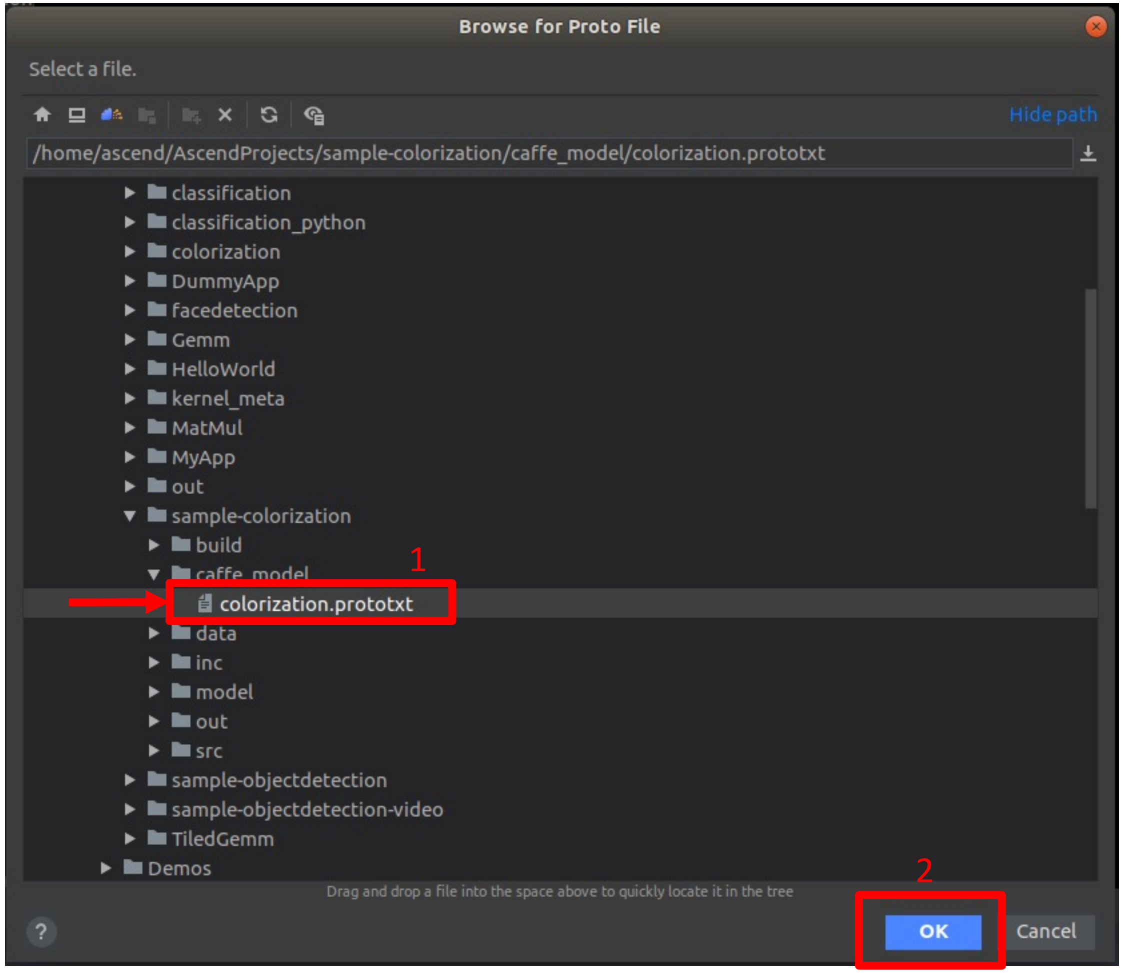
Model conversion /2

Browse for model and weight files



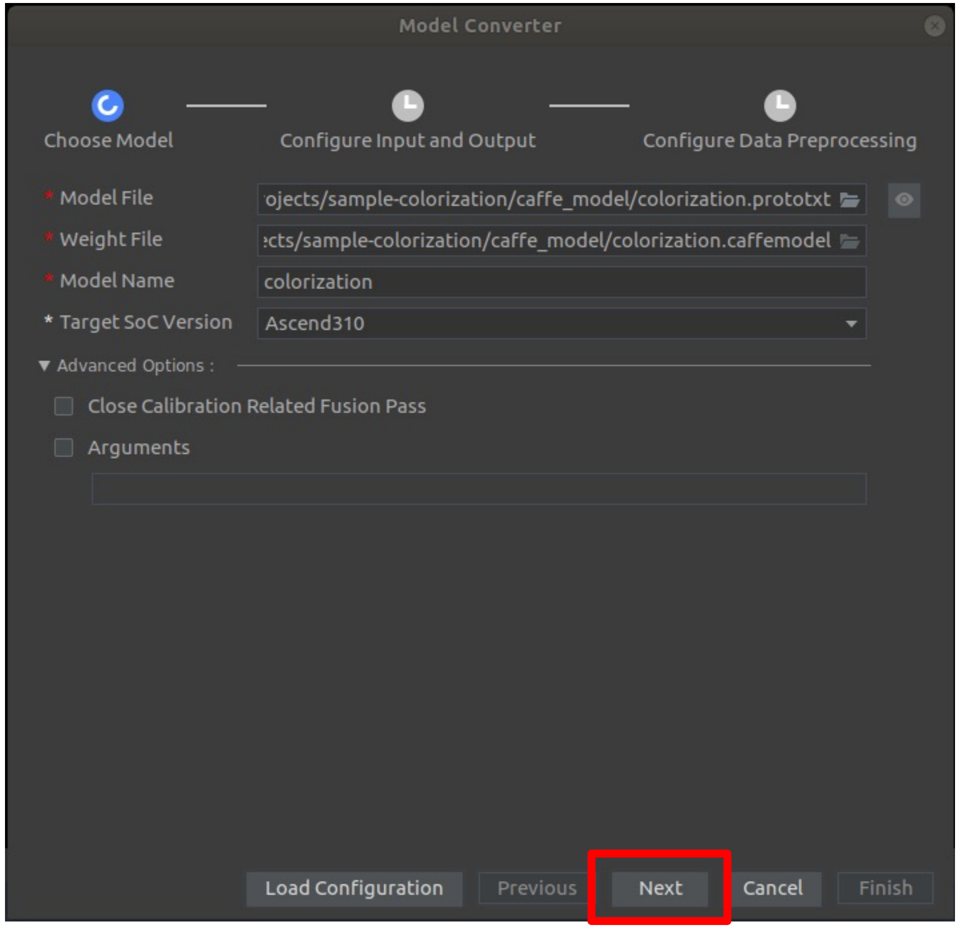
Model conversion /3

Select Caffe proto file for the colorization model



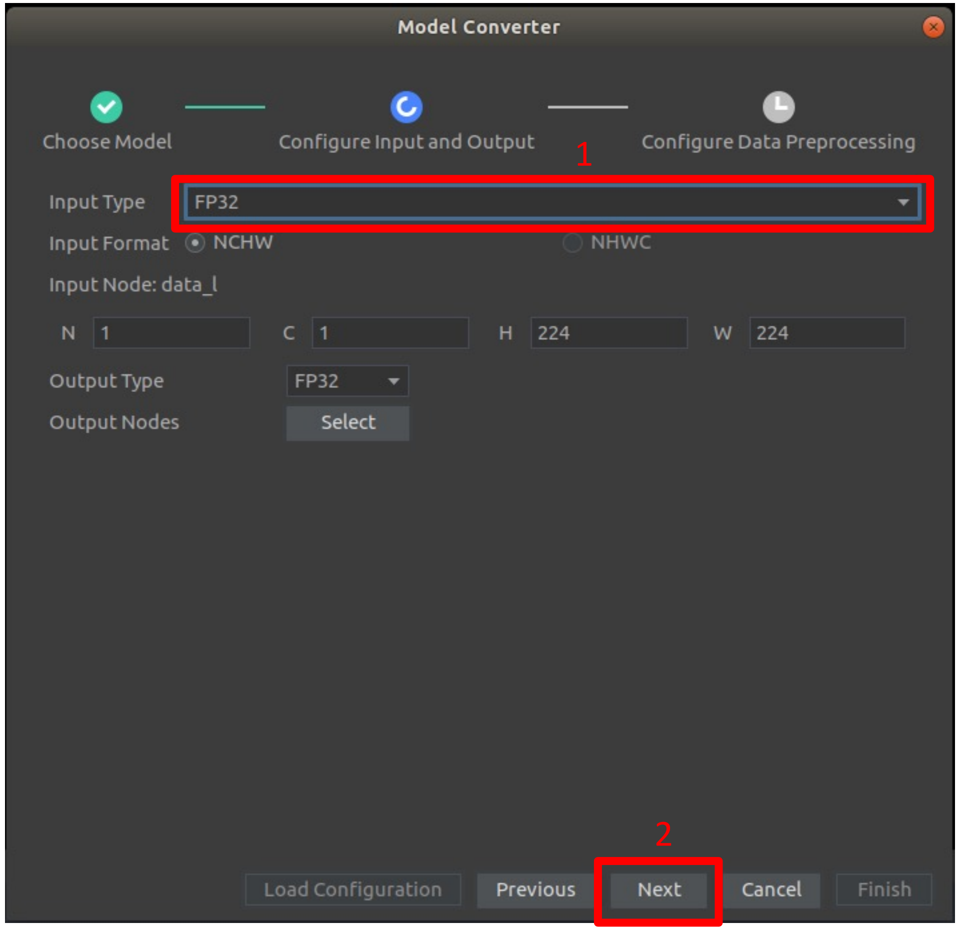
Model conversion /4

The weight file is automatically discovered



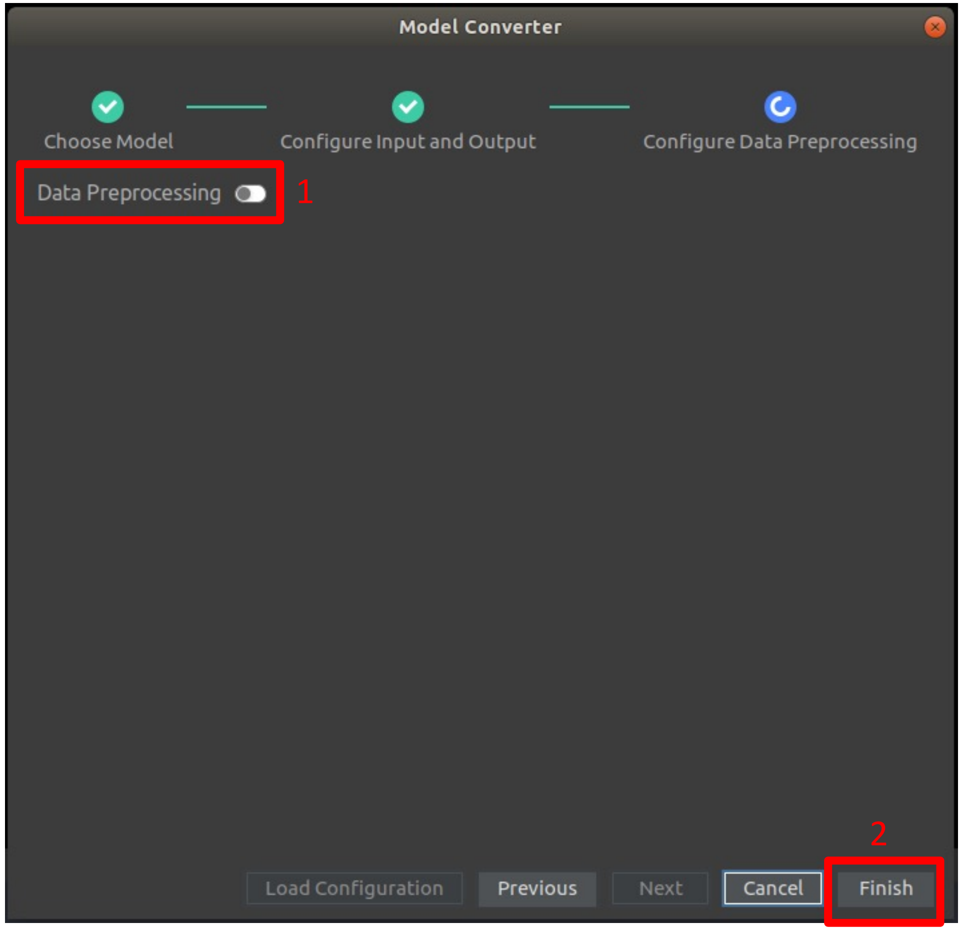
Model conversion /5

The colorization network requires FP32 type for the input and output data

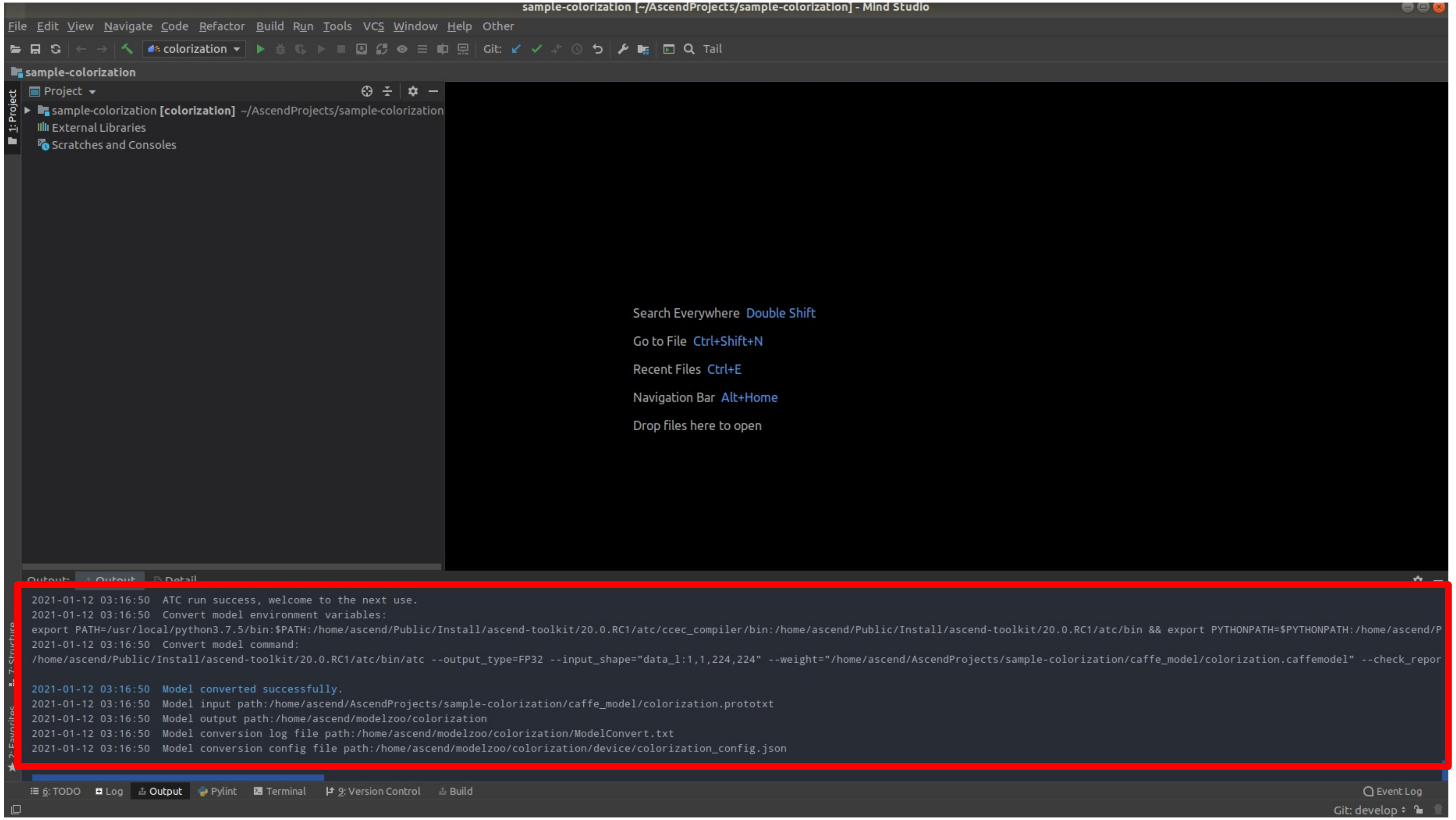


Model conversion /6

On-device data preprocessing is not used

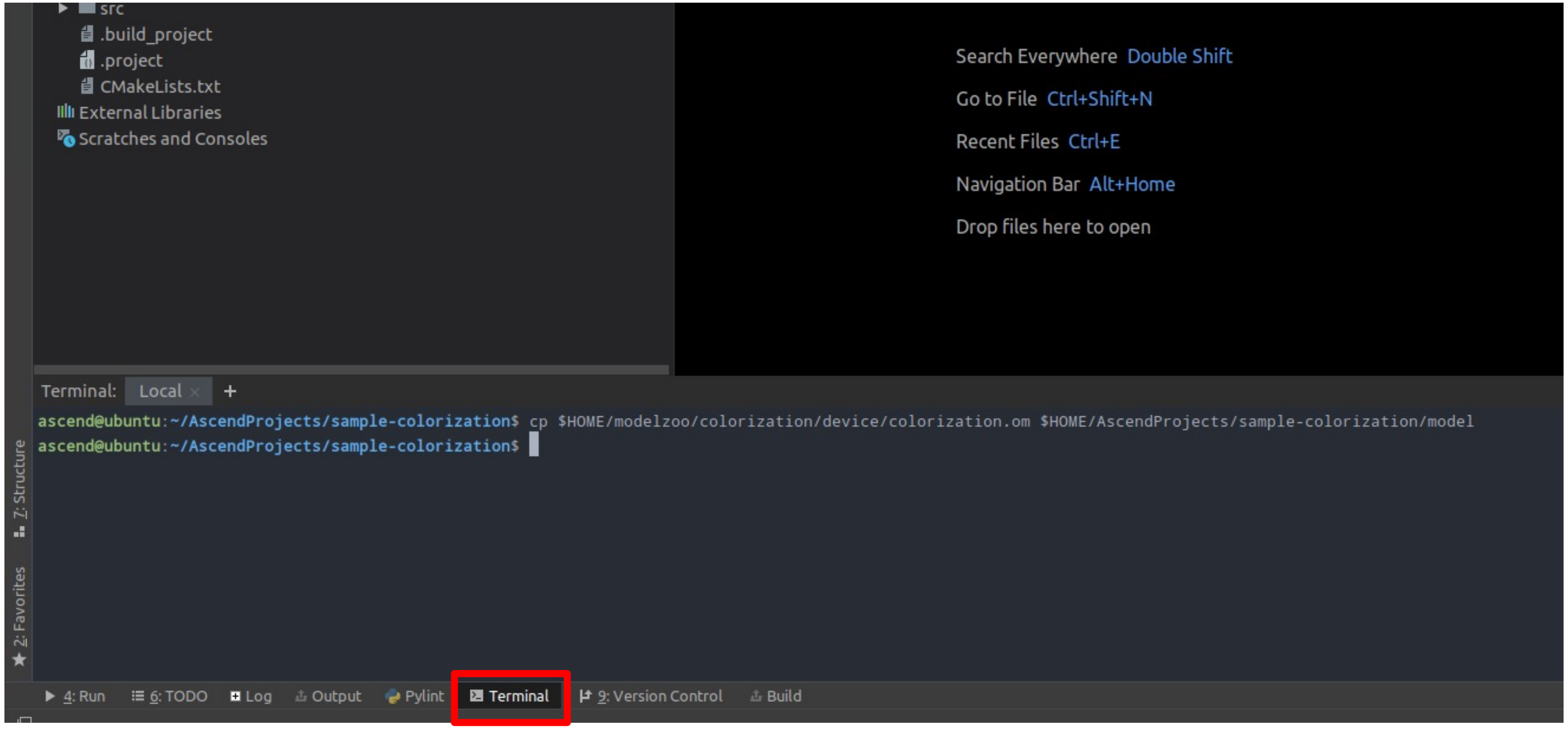


Model conversion /7

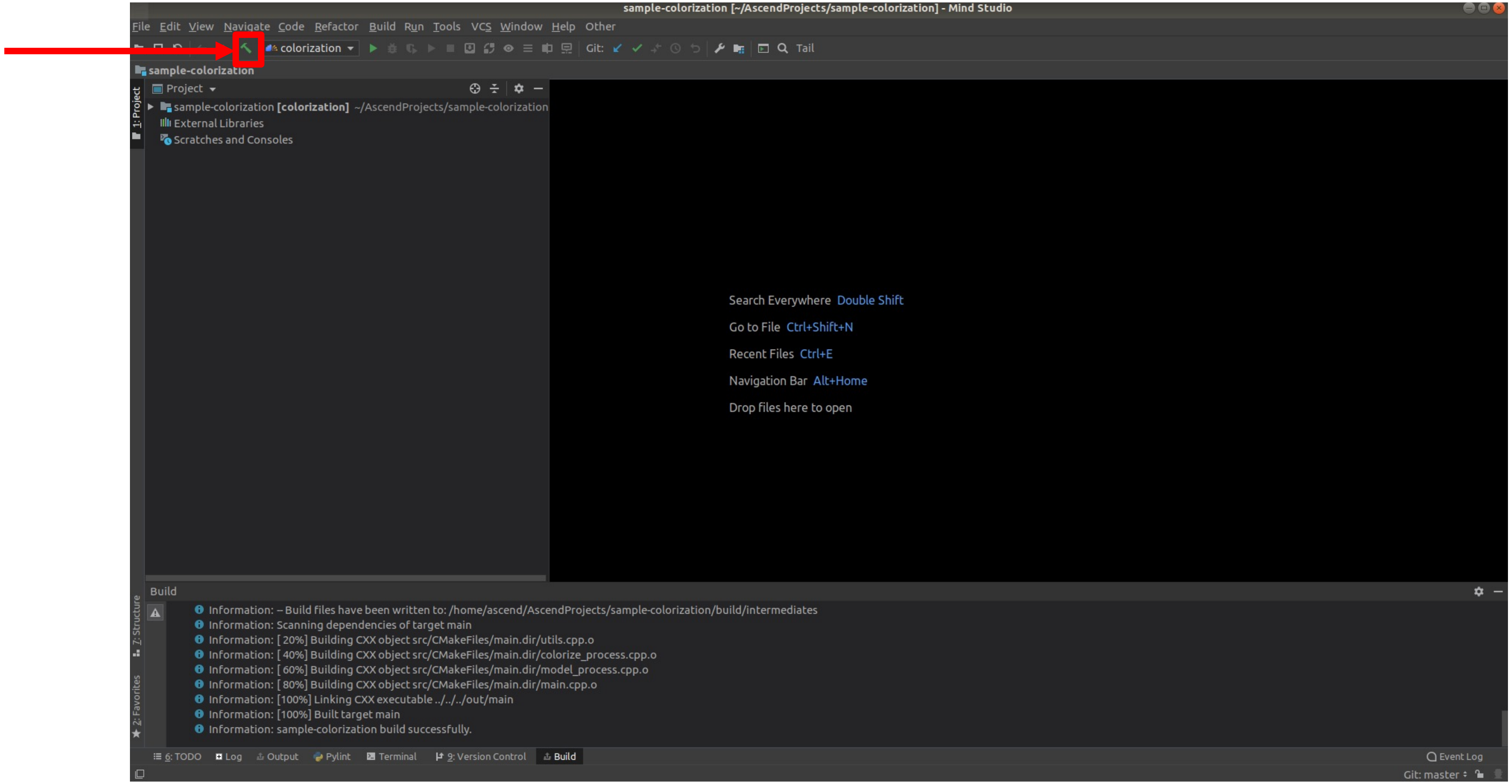


Model conversion /8

```
cp $HOME/modelzoo/colorization/device/colorization.om $HOME/AscendProjects/sample-colorization/model
```

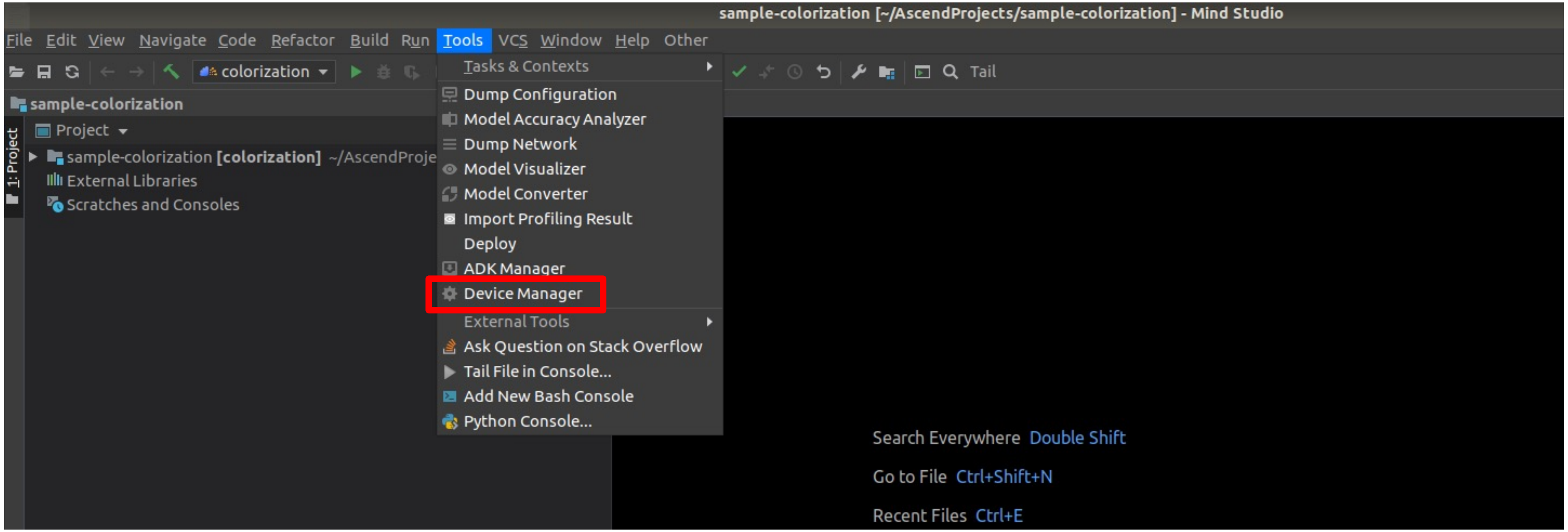


Building the project



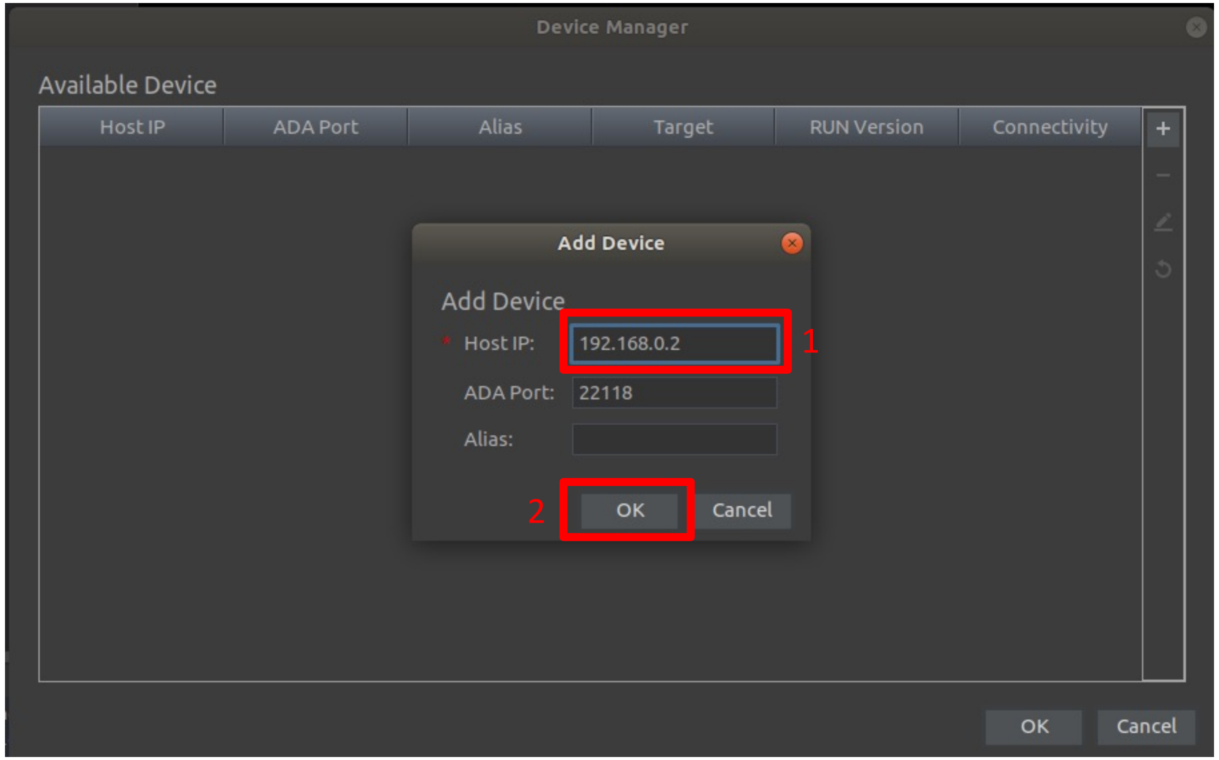
Setting the target device /1

Open the device manager and register your target device using its IP address

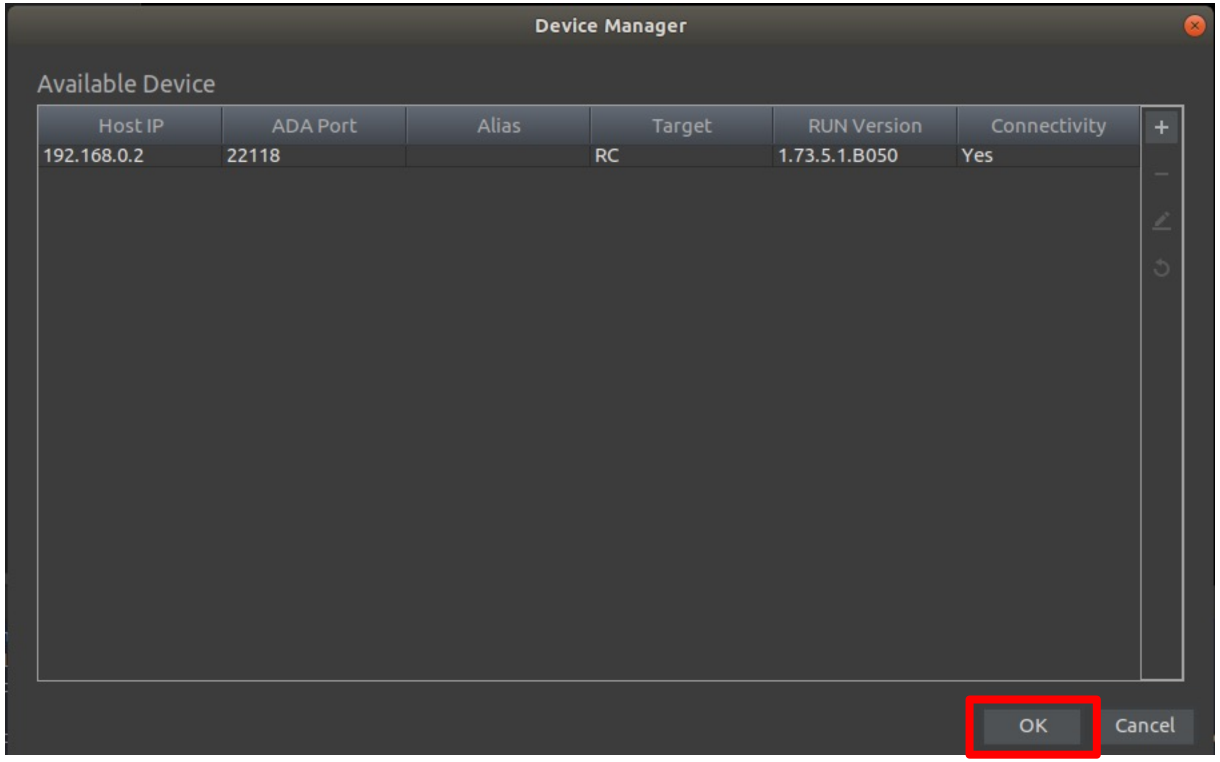


Setting the target device /2

The port 22118 must be open; otherwise MindStudio will not be able to communicate with the target device.



Setting the target device /3



Running inference /1

- **Login to the developer board using the right IP address and password as following:**

  `ssh HwHiAiUser@192.168.0.2`

  (The default password is Mind@123)

- **Restart ada**

  As new environment variables are set, the `ada` tool in the operating environment needs to be restarted. Otherwise, the development environment cannot access the newly set environment variables in the operating environment.

  Run the following commands as a common user to view the process ID of the ada tool:

```
HwHiAiUser@davinci-mini:~$ ps -ef | grep ada
HwHiAiU+  1996     1  0 08:40 ?        00:00:00 /var/ada
HwHiAiU+  2060  2047  0 08:41 pts/0    00:00:00 grep --color=auto ada
```

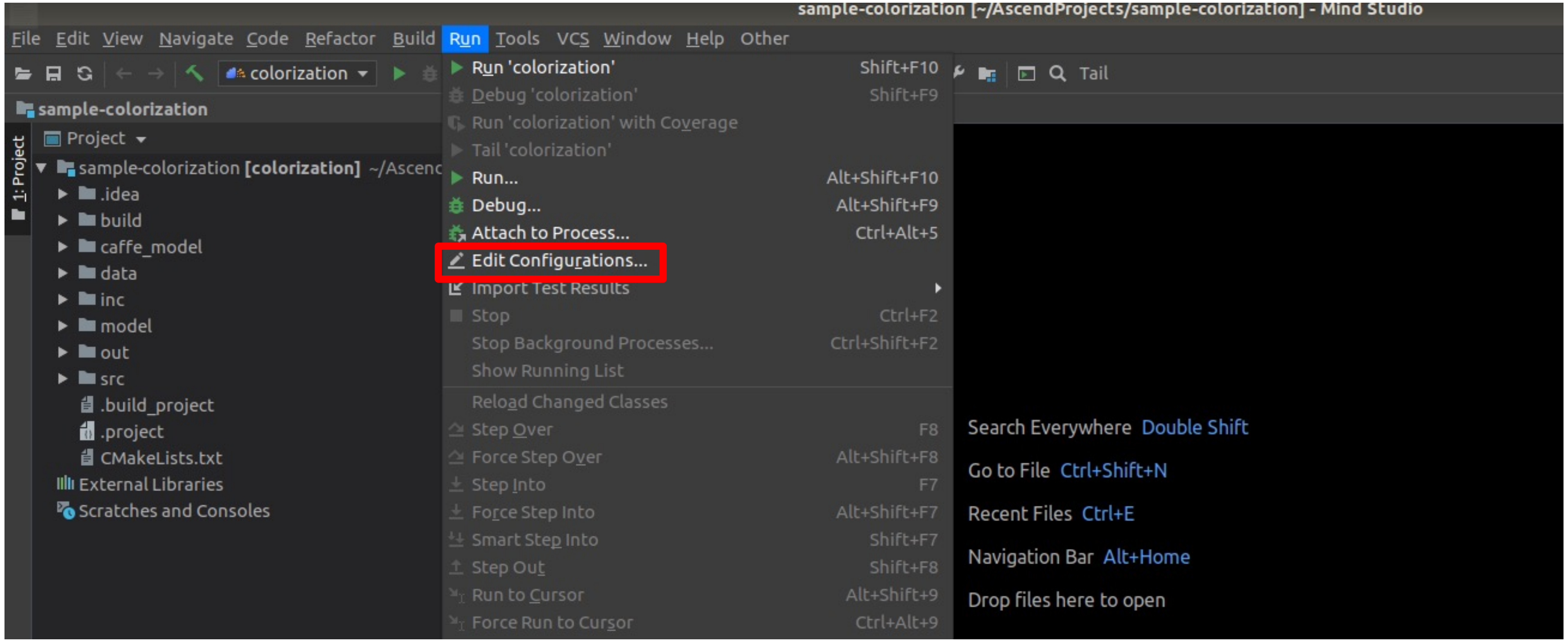
  Since the process ID of the ada tool is 1996, run the following command to kill the ada process:

```
kill -9 1996
```

  Run the following commands as a common user to restart ada:

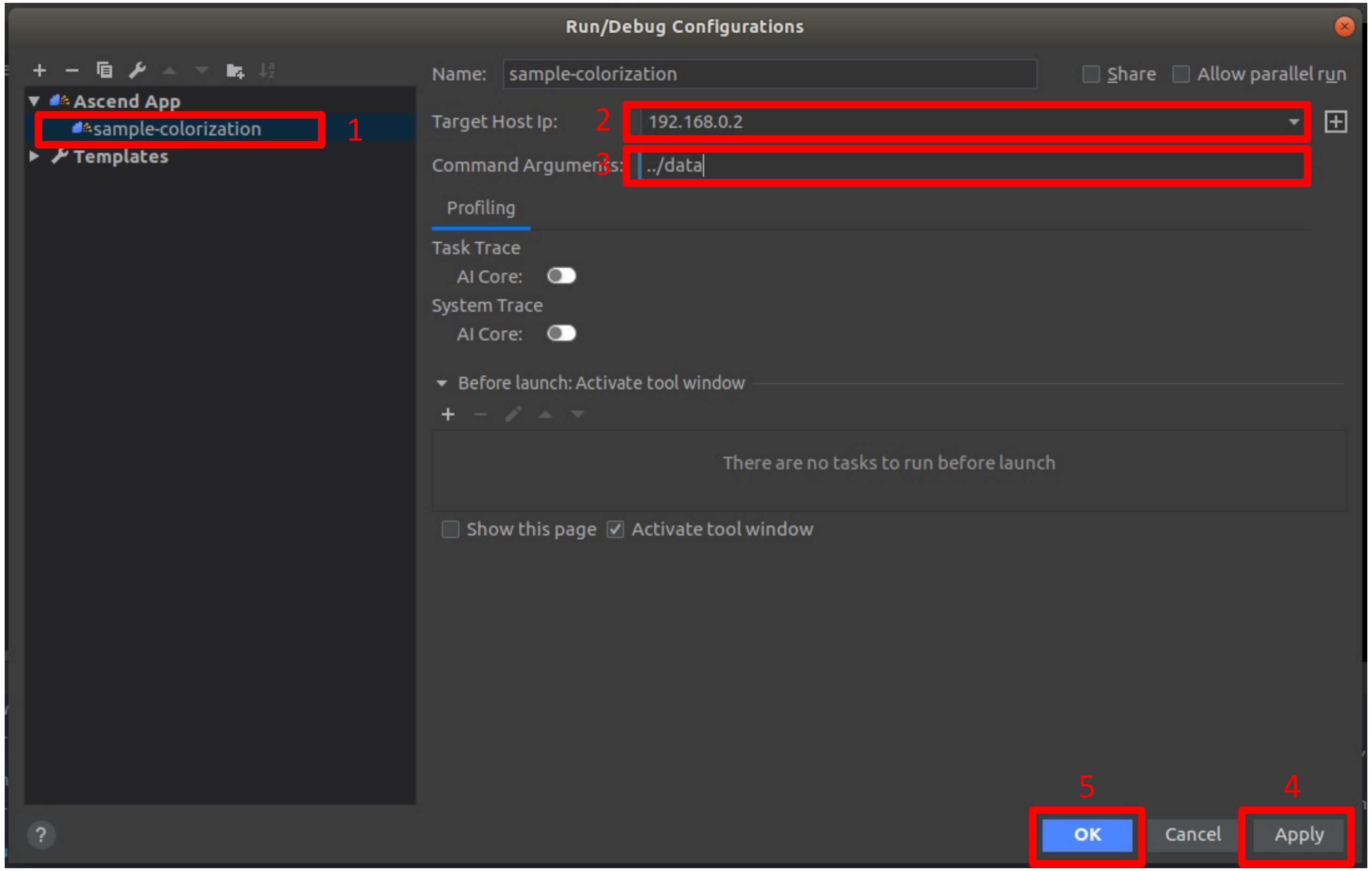
```
HwHiAiUser@davinci-mini:/var$ cd /var/
HwHiAiUser@davinci-mini:/var$ ./ada &
```

Running inference /2

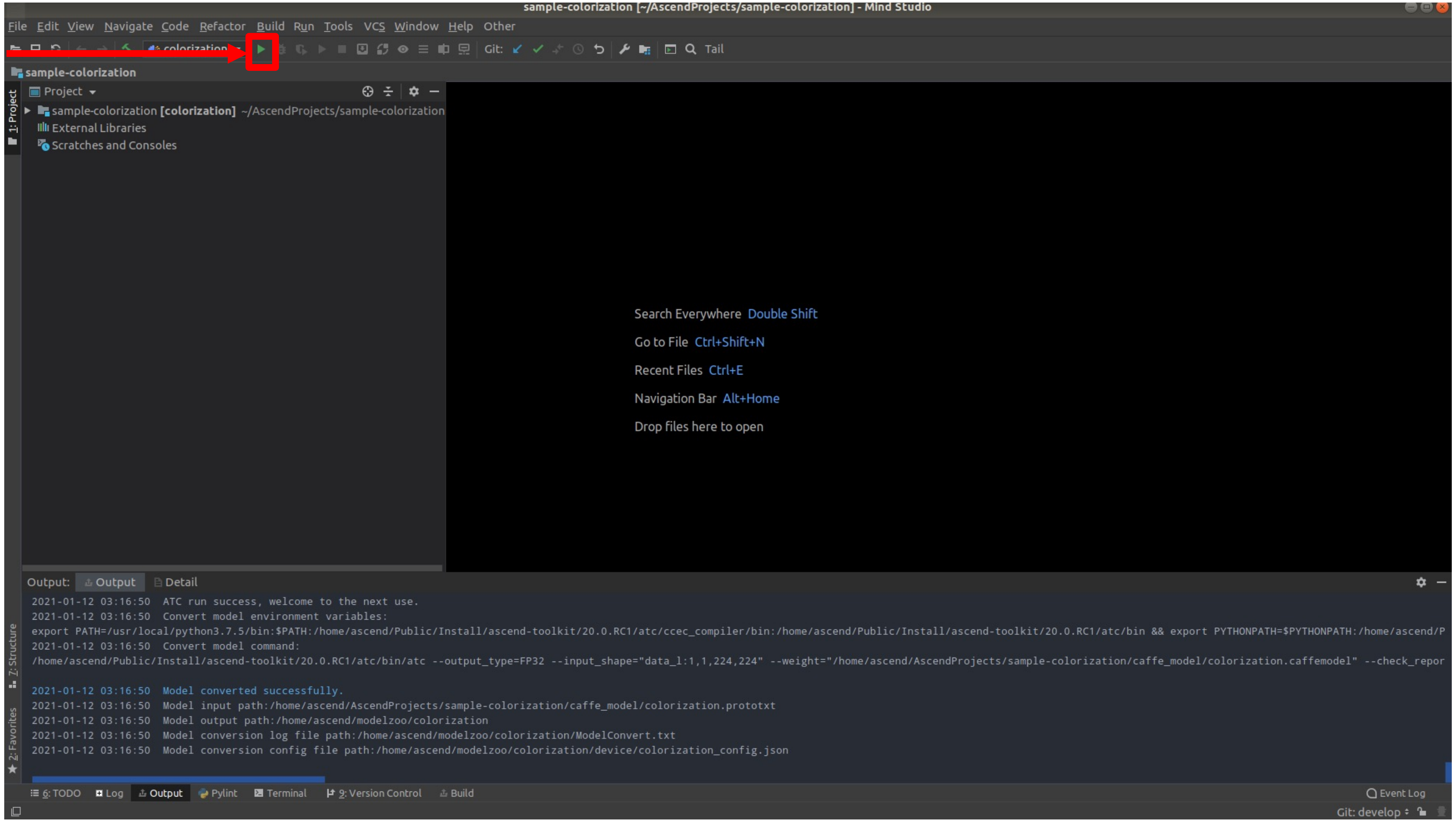


Running inference /3

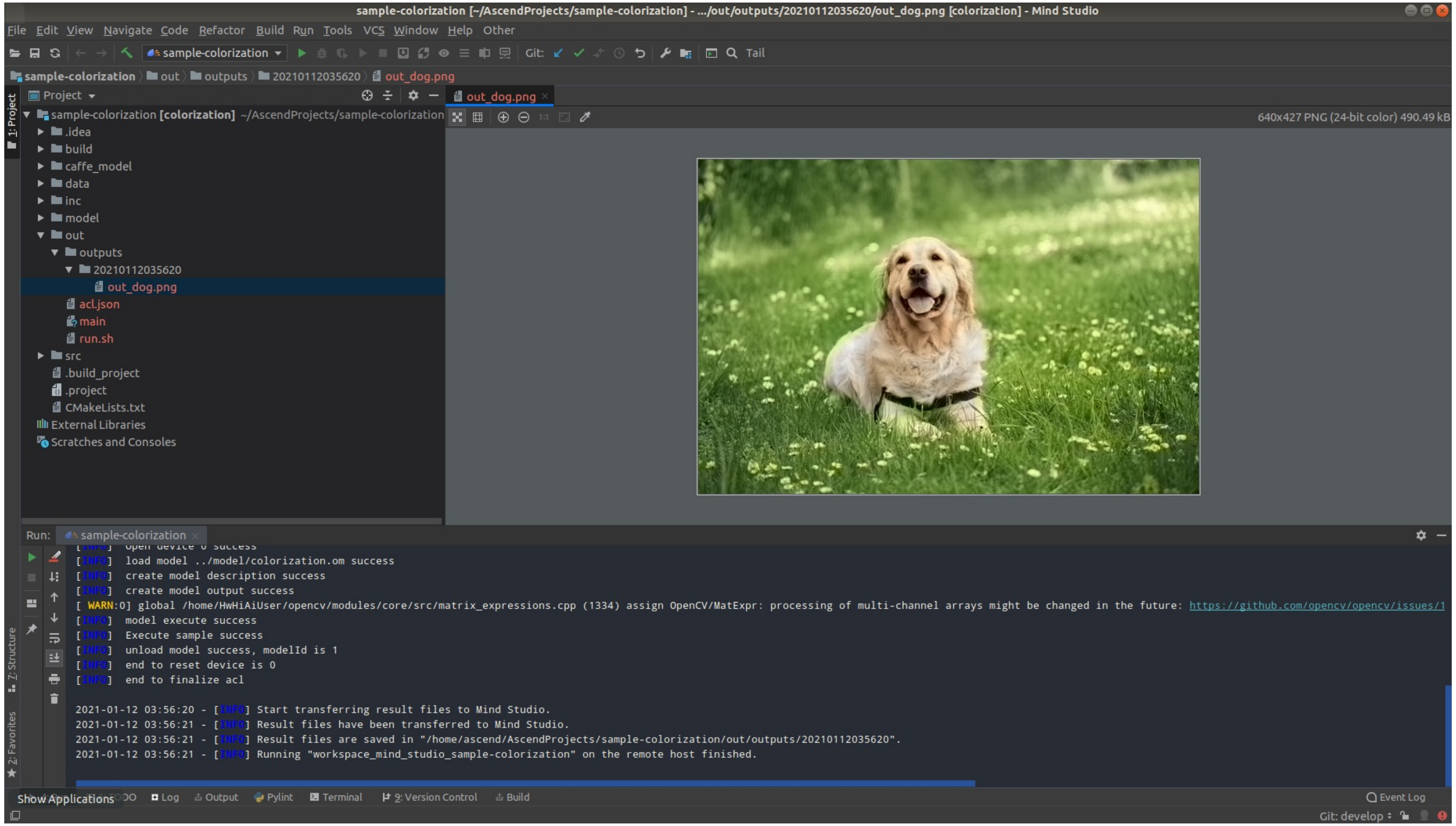
Select target host IP and set input data path



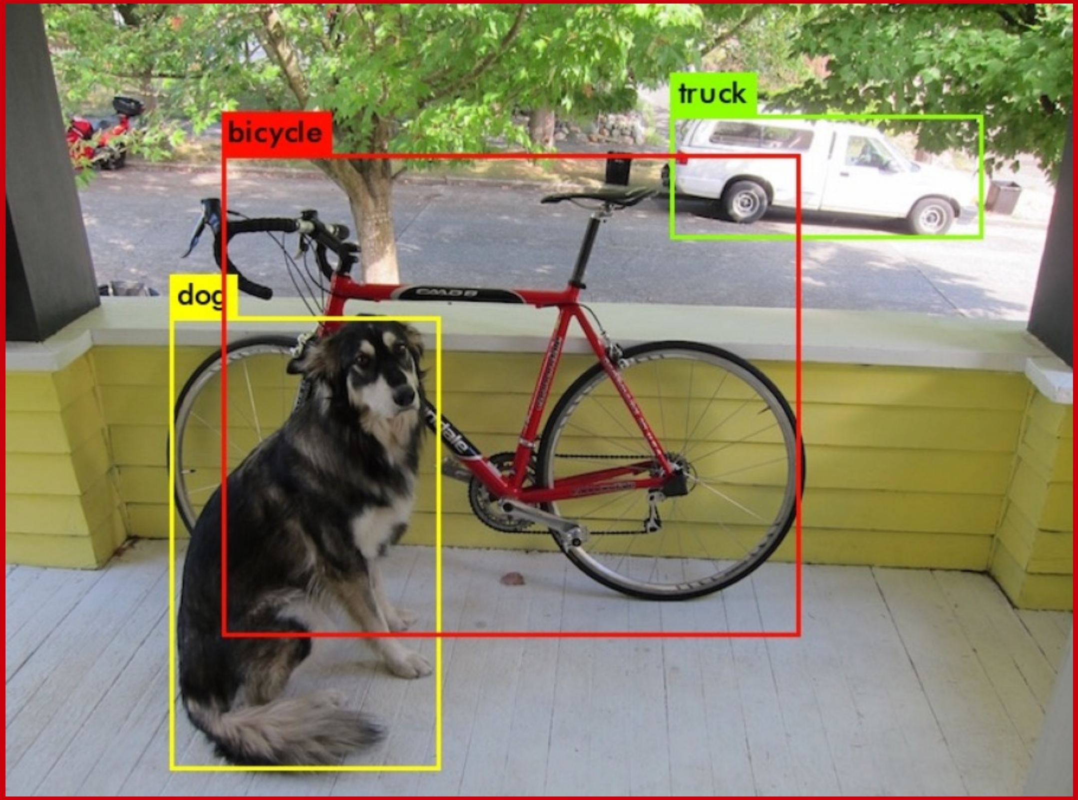
Running inference /4

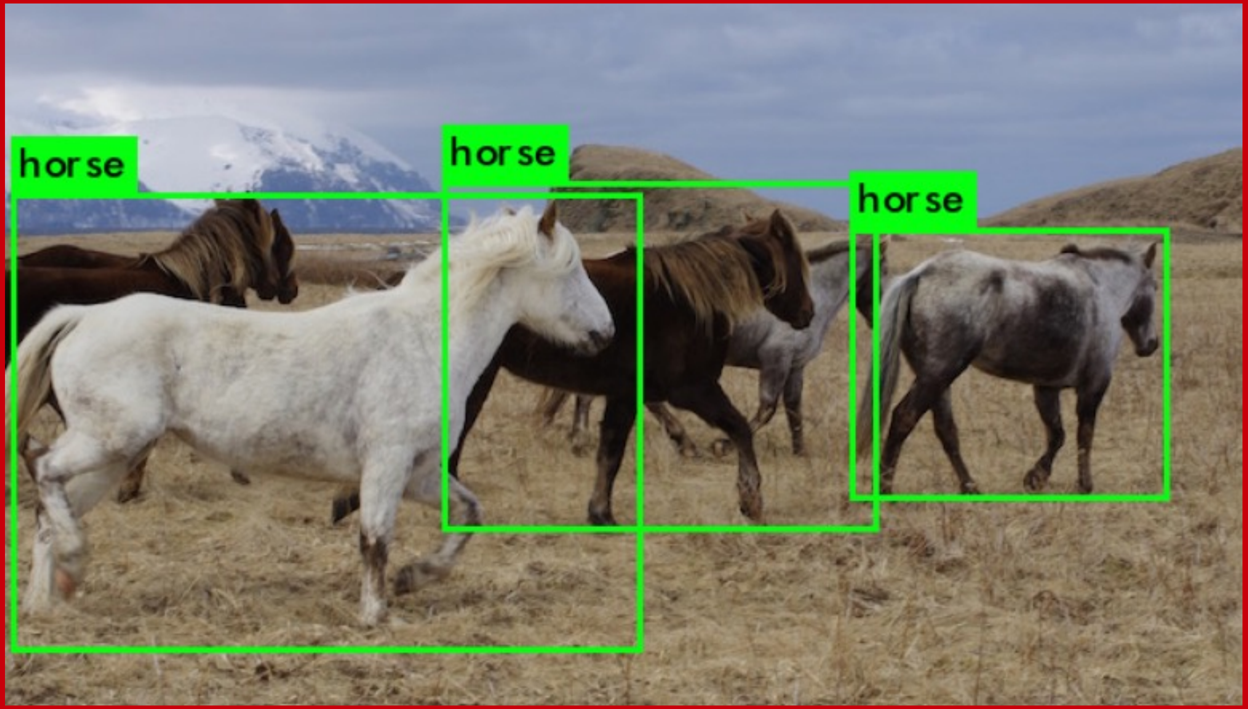


Running inference /5



Object Detection using YOLOv3 [Redmon et al., 2016]





Download project source code

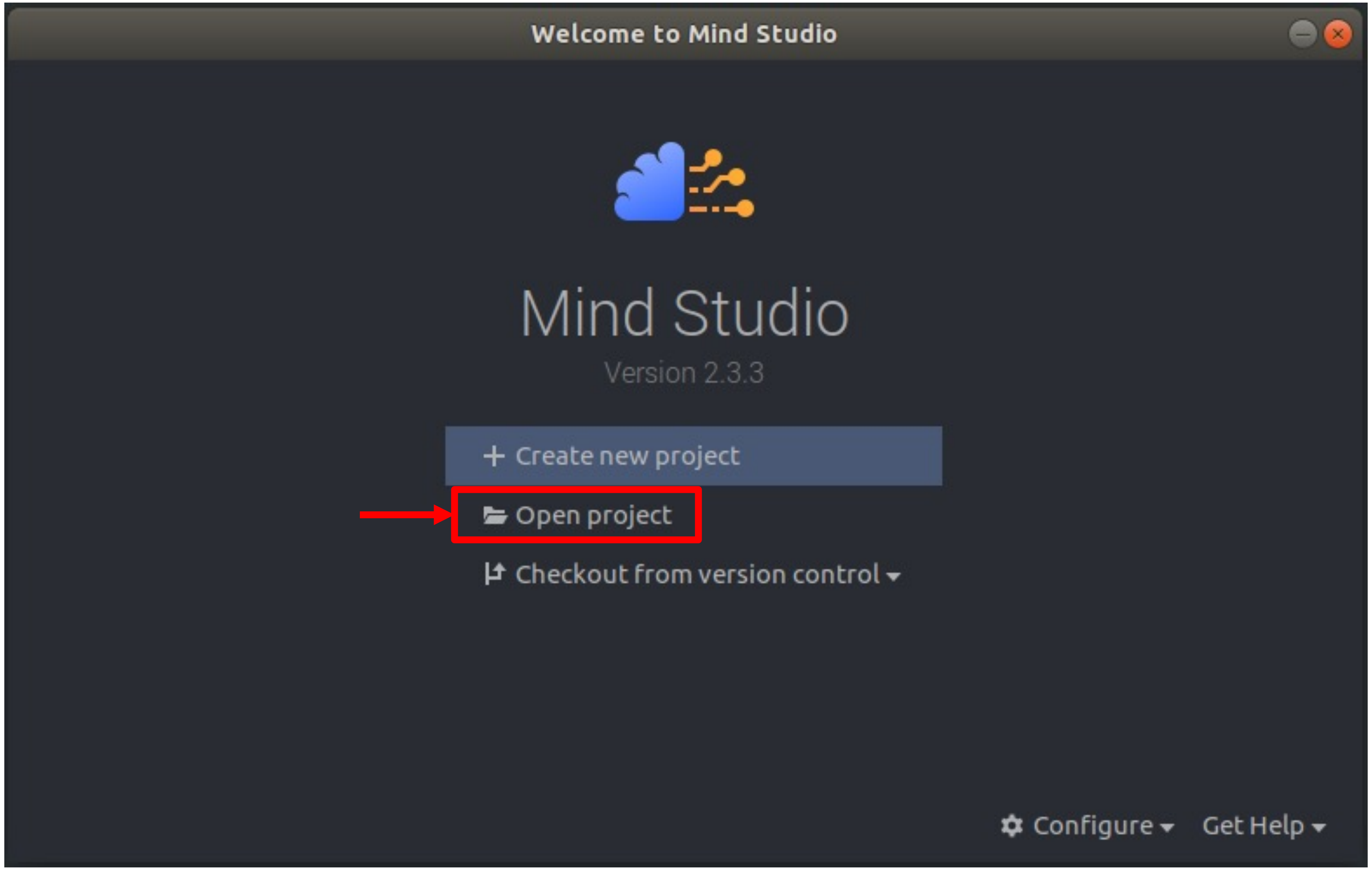
```
mkdir -p $HOME/AscendProjects
cd $HOME/AscendProjects
```

```
# Obtain the object detection project package.
git clone https://gitlab.schihei.de/schihei/sample-objectdetection.git
```

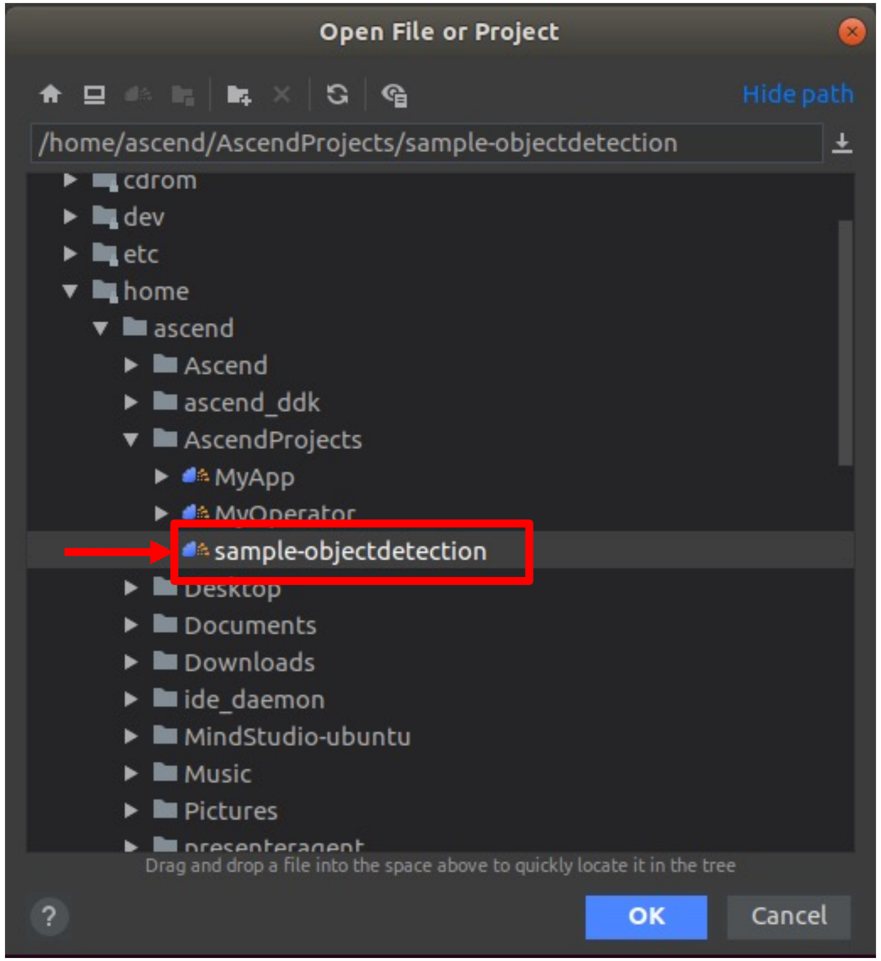
The repository includes the original Caffe model that will be converted to offline model adapted to Ascend platforms.

Loading the project /1

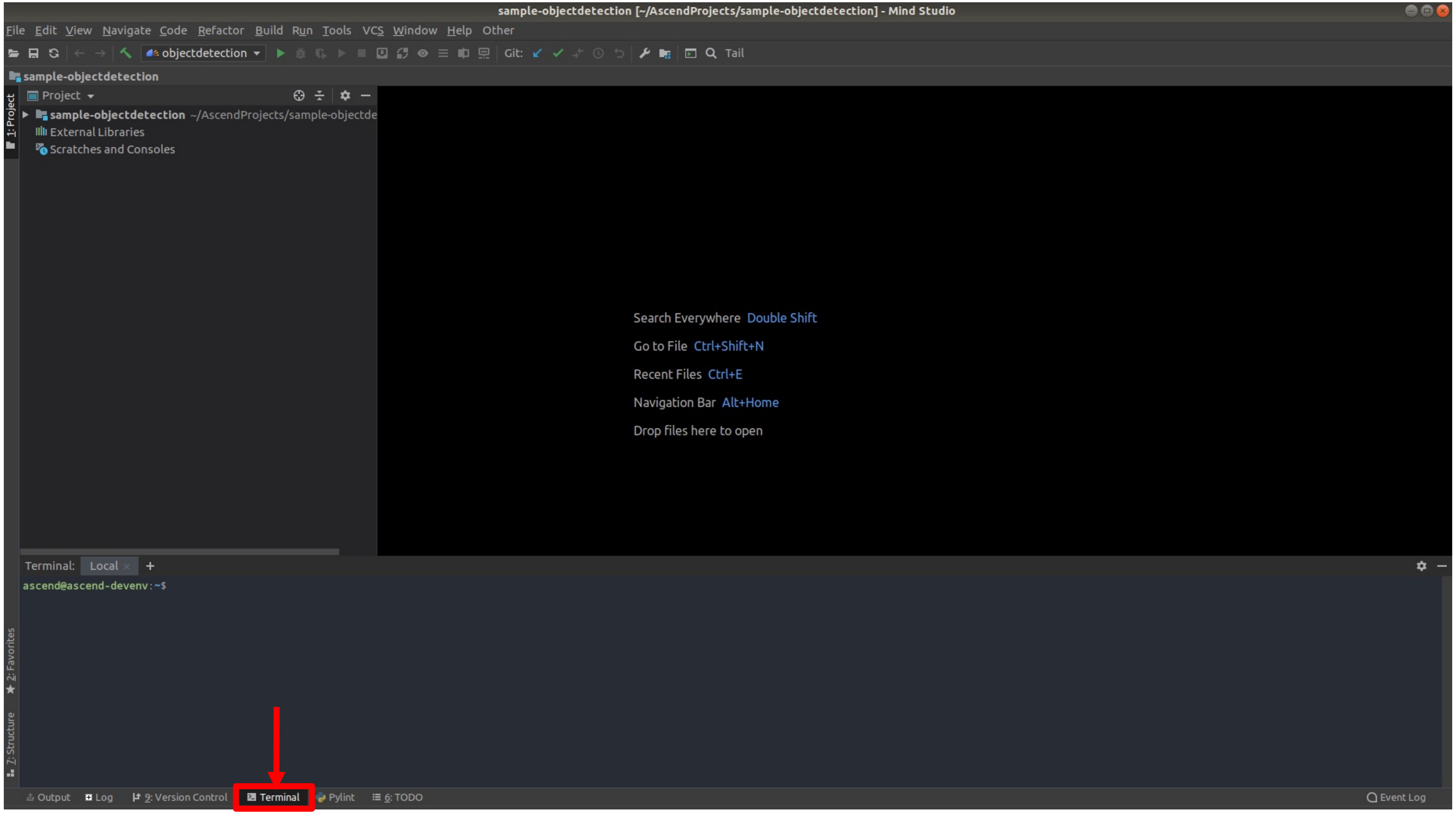
```
cd $HOME/MindStudio-ubuntu/bin && ./MindStudio.sh &
```



Loading the project /2



Loading the project /3

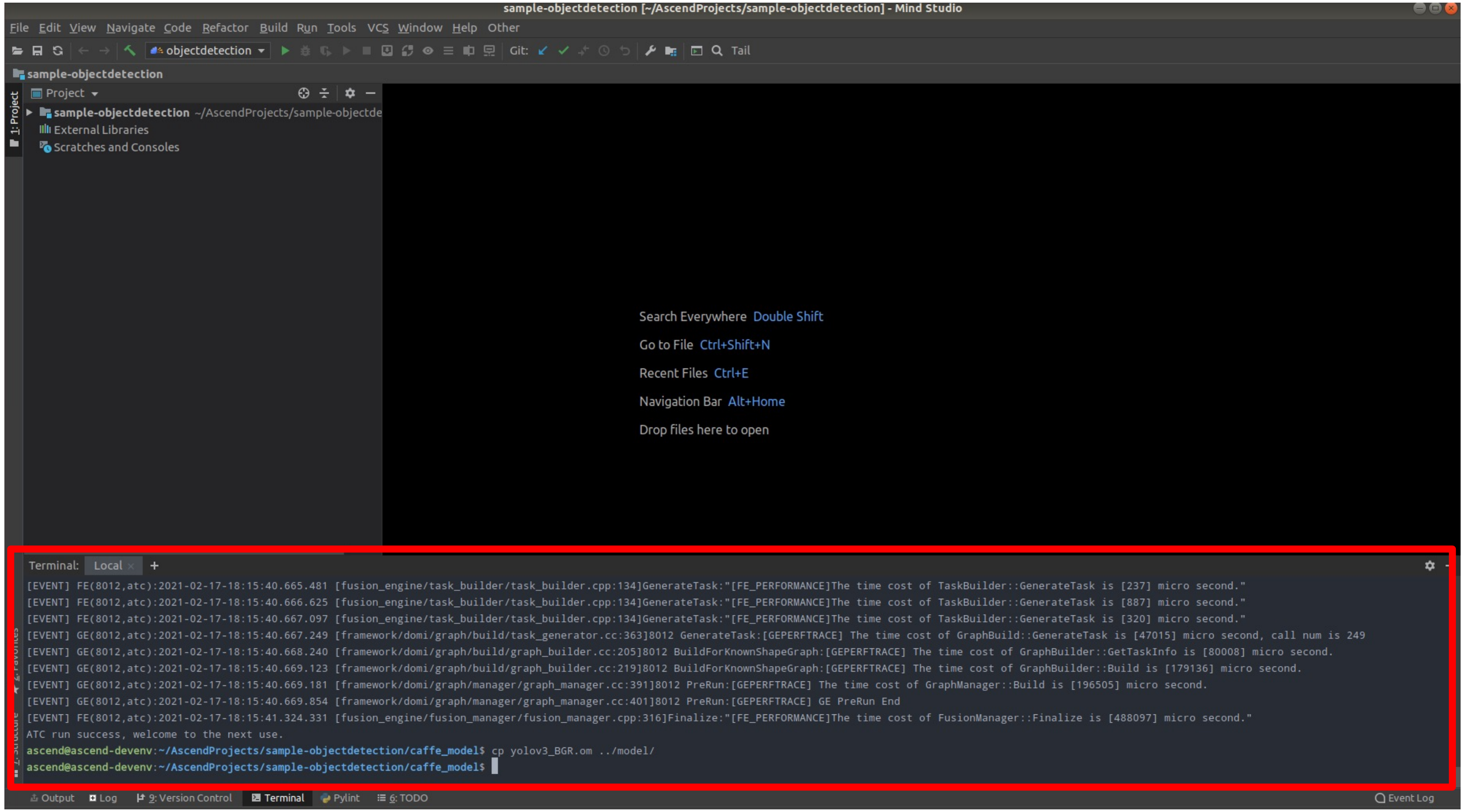


Model conversion /1

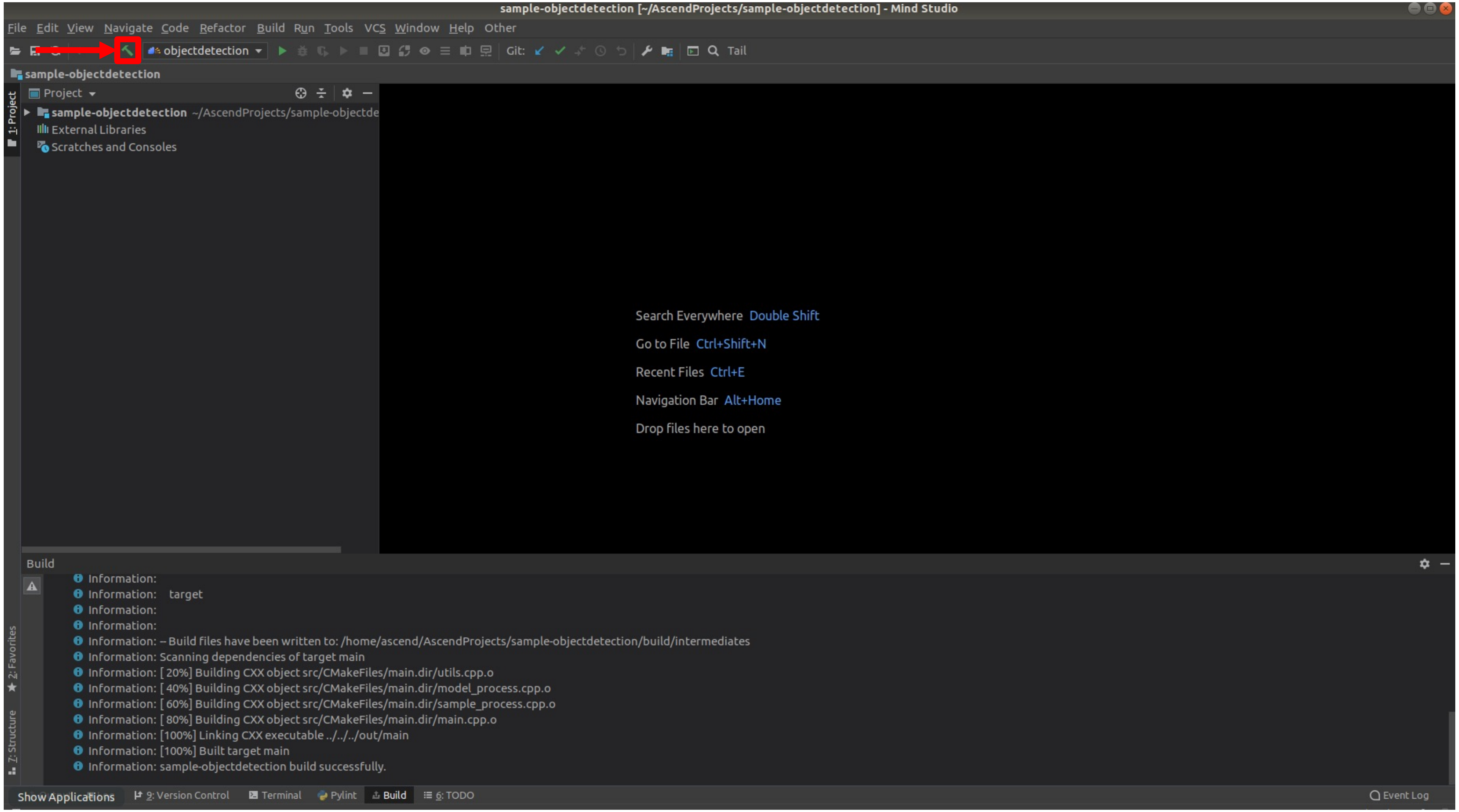
```
cd $HOME/AscendProjects/sample-objectdetection/caffe_model
```

```
atc --model=yolov3.prototxt --weight=yolov3.caffemodel --framework=0 --output=yolov3_BGR \
--soc_version=Ascend310 --insert_op_conf=aipp_bgr.cfg
```

```
cp yolov3_BGR.om ../model/
```

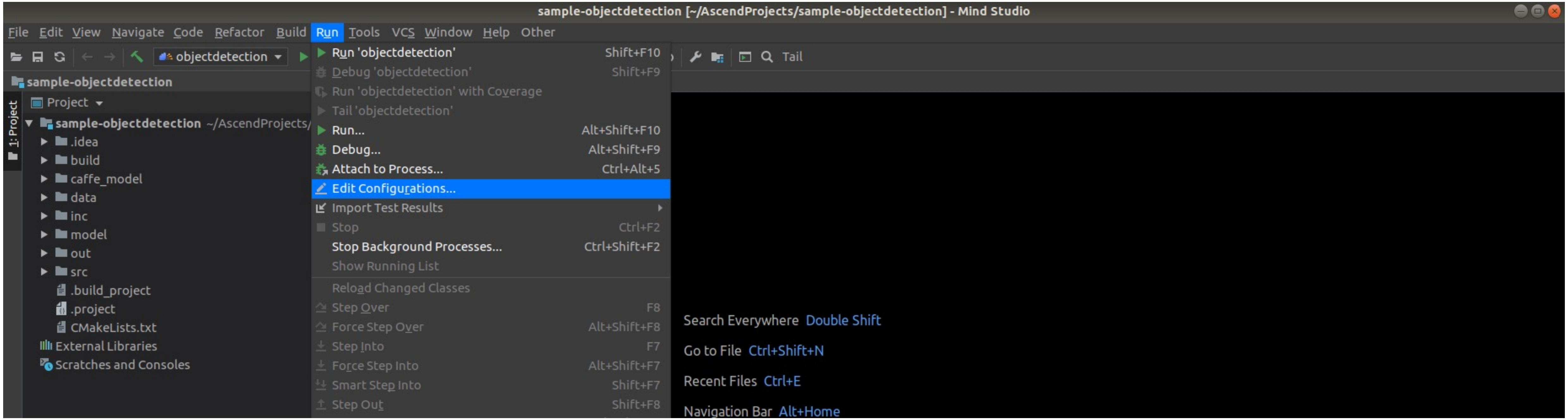


Building the project

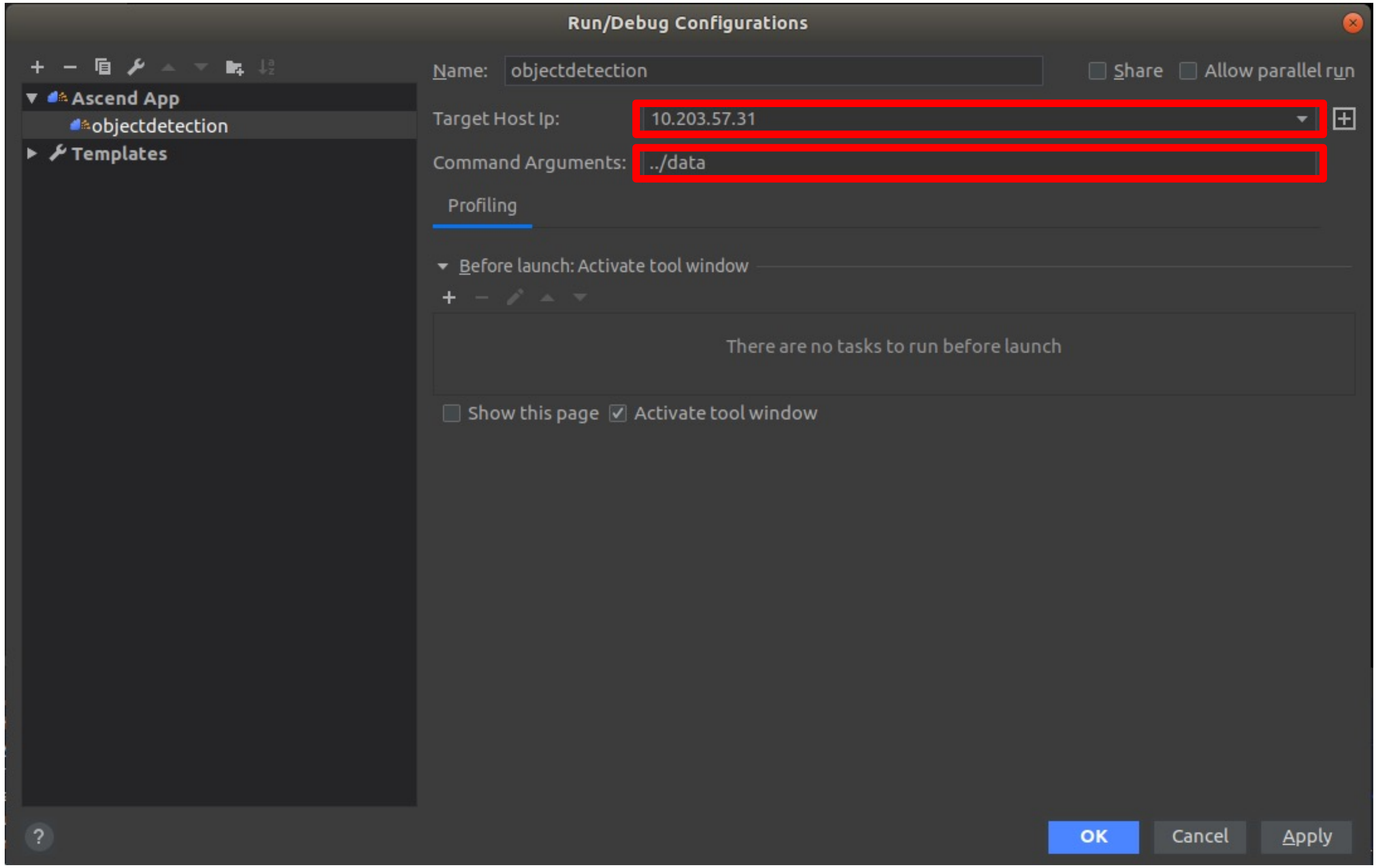


Running inference /1

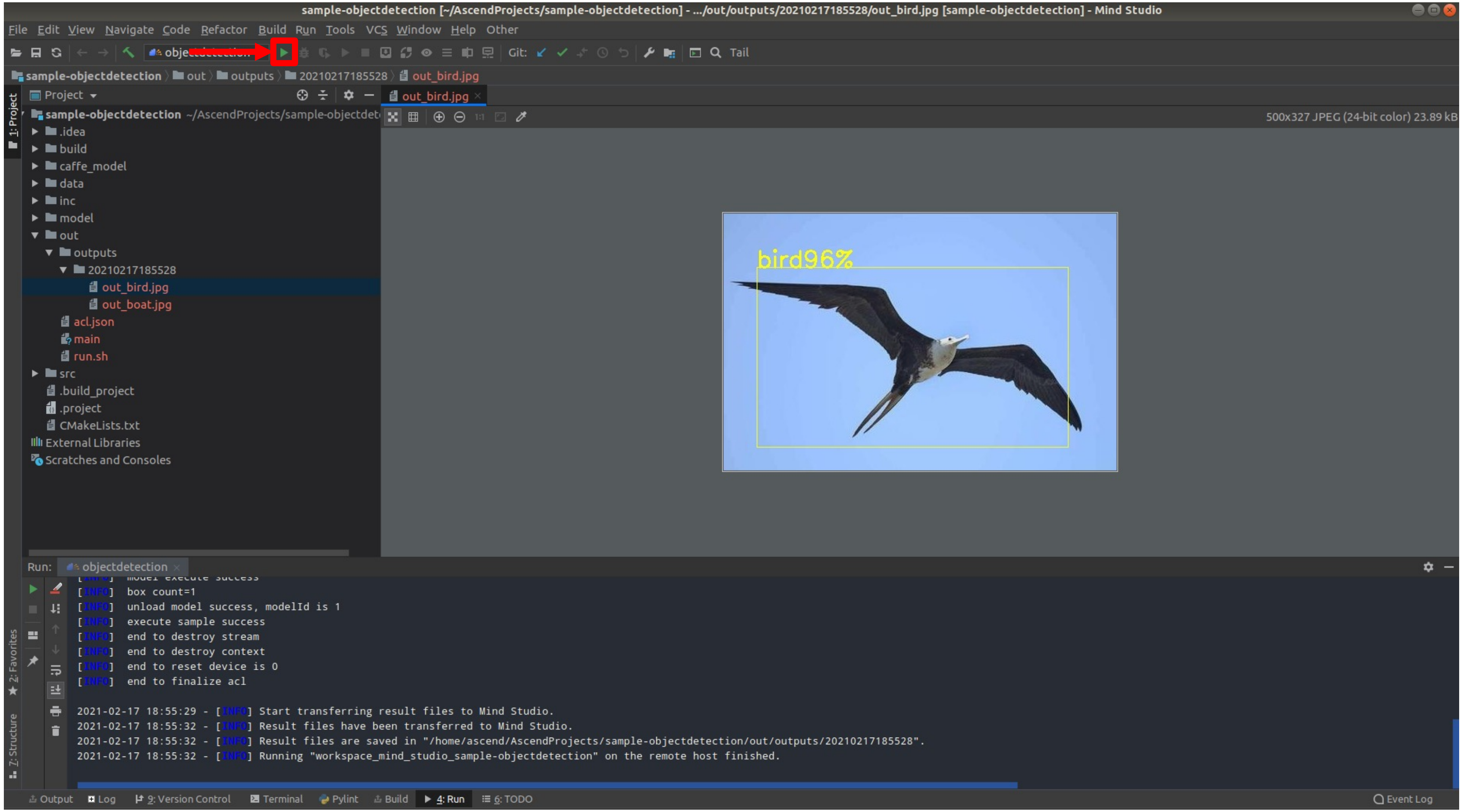
Edit run configuration for selecting the target host and specifying the application input argument



Running inference /2

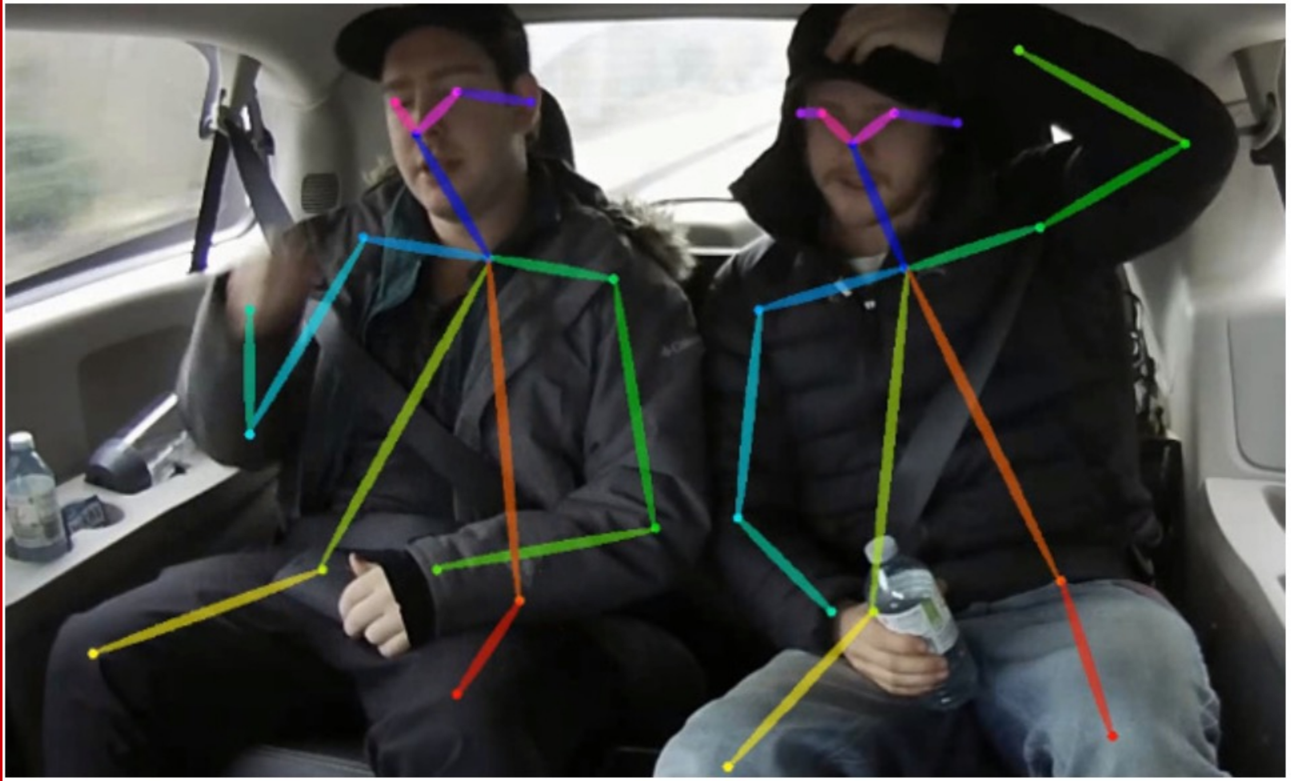


Running inference /2



Body Pose Estimation

[Osokin et al., 2016]



Download project source code

This Python project will be directly executed on the Atlas 200DK

Download the project locally in the development environment and copy it to the board

1) Obtain the object detection project package

```
cd $HOME/AscendProjects
```

```
git clone https://gitlab.schihei.de/schihei/sample-bodypose.git
```

2) Copy the project to the board

```
scp -r sample-bodypose/ HwHiAiUser@192.168.0.2:~/HIAI_PROJECTS/sample-bodypose
```

NOTES:
- The IP address of the board (here 192.168.0.2) needs to be changed in the `scp` command to match that of your board
- If the board is connected to the internet, one could download the project directly on the board by cloning the repository

The repository includes the offline model adapted to Ascend platforms that will be used for estimating the body pose. The original model is developed in PyTorch and available at https://github.com/Daniil-Osokin/lightweight-human-pose-estimation.pytorch

Running inference /1

Body pose estimation (image)

Connect to the board

```
ssh HwHiAiUser@192.168.0.2
```

Launch inference

```
cd ~/HIAI_PROJECTS/sample-bodypose

python3 code_image/main.py --model='model/body_pose.om' --frames_input_src='code_image/tennis_player.jpg' \
--output_dir='code_image/outputs'
```

After successful inference execution, the output image is stored in the directory `~/HIAI_PROJECTS/sample-bodypose/code_image/outputs`

Open a terminal on the development environment and copy back result from the board to the local folder

```
cd $HOME/AscendProjects/sample-bodypose/code_image

scp -r HwHiAiUser@192.168.0.2:~/Scratch/sample-bodypose/code_image/outputs .
```

Running inference /2

Body pose estimation (image)

Running inference /3

Body pose estimation (video)

Connect to the board
```
ssh HwHiAiUser@192.168.0.2
```

Launch inference
```
cd ~/HIAI_PROJECTS/sample-bodypose

python3 code_video/main.py --model='model/body_pose.om' --frames_input_src='code_video/yoga.mp4' \
--output_dir='code_video/outputs'
```

After successful inference execution, the output video is stored in the directory `~/HIAI_PROJECTS/sample-bodypose/code_video/outputs`

Open a terminal on the development environment and copy back result from the board to the local folder
```
cd $HOME/AscendProjects/sample-bodypose/code_video

scp -r HwHiAiUser@192.168.0.2:~/Scratch/sample-bodypose/code_video/outputs .
```

HUAWEI

Stay safe — stay healthy

References /1

Alake, R. (2020). 10 Stages Of A Machine Learning Project In 2020 (And Where You Fit). Retrieved February 20, 2020, from https://towardsdatascience.com/10-stages-of-a-machine-learning-project-in-2020-and-where-you-fit-cb73ad4726cb

Amodei, D., Hernandez, D., Sastry, G., Clark, J., Brockman, G., & Sutskever, I. (2019). AI and Compute. Retrieved February 16, 2020, from https://openai.com/blog/ai-and-compute/

Badgeley, M. A., Zech, J. R., Oakden-Rayner, L., Glicksberg, B. S., Liu, M., Gale, W., … Dudley, J. T. (2018). Deep Learning Predicts Hip Fracture using Confounding Patient and Healthcare Variables. *Npj Digital Medicine*, *2*(1). Retrieved from http://arxiv.org/abs/1811.03695

Batra, G., Jacobson, Z., Madhav, S., Queirolo, A., & Santhanam, N. (2018). *Artificial-intelligence hardware: New opportunities for semiconductor companies*. https://www.mckinsey.com/~/media/McKinsey/Industries/Semiconductors/Our Insights/Artificial intelligence hardware New opportunities for semiconductor companies/Artificial-intelligence-hardware.pdf

Carneiro, G., Oakden-Rayner, L., Bradley, A. P., Nascimento, J., & Palmer, L. (2016). Automated 5-year Mortality Prediction using Deep Learning and Radiomics Features from Chest Computed Tomography. *Proceedings - International Symposium on Biomedical Imaging*, 130–134. Retrieved from http://arxiv.org/abs/1607.00267

Chuang, K. V., & Keiser, M. J. (2018). Adversarial Controls for Scientific Machine Learning. *ACS Chemical Biology*, *13*(10), 2819–2821. https://doi.org/10.1021/acschembio.8b00881

Coudray, N., Ocampo, P. S., Sakellaropoulos, T., Narula, N., Snuderl, M., Fenyö, D., … Tsirigos, A. (2018). Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature Medicine*, *24*(10), 1559–1567. https://doi.org/10.1038/s41591-018-0177-5

de Groof, A. J., Struyvenberg, M. R., van der Putten, J., van der Sommen, F., Fockens, K. N., Curvers, W. L., … Bergman, J. J. (2019). Deep-Learning System Detects Neoplasia in Patients With Barrett's Esophagus With Higher Accuracy Than Endoscopists in a Multi-Step Training and Validation Study with Benchmarking. *Gastroenterology*. https://doi.org/10.1053/j.gastro.2019.11.030

de Groof, A. J., Struyvenberg, M. R., Fockens, K. N., van der Putten, J., van der Sommen, F., Boers, T. G., … Bergman, J. J. G. H. M. (2020). Deep learning algorithm detection of Barrett's neoplasia with high accuracy during live endoscopic procedures: a pilot study (with video). *Gastrointestinal Endoscopy*. https://doi.org/10.1016/j.gie.2019.12.048

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118. https://doi.org/10.1038/nature21056

Finlayson, S. G., Chung, H. W., Kohane, I. S., & Beam, A. L. (2018). *Adversarial Attacks Against Medical Deep Learning Systems*. Retrieved from http://arxiv.org/abs/1804.05296

Gerges, S. (2017). Software Complementing Hardware: Artificial intelligence in the clinic. Retrieved February 17, 2020, from http://sitn.hms.harvard.edu/flash/2017/software-complementing-hardware-artificial-intelligence-clinic/

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., … Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA - Journal of the American Medical Association*, *316*(22), 2402–2410. https://doi.org/10.1001/jama.2016.17216

Hopkins, B. (2017). Why You Are Getting Disrupted. Retrieved February 17, 2020, from https://go.forrester.com/blogs/17-05-09-why_you_are_getting_disrupted/?_lrsc=94d1ae97-b805-4567-b1b6-f15ff44dee02&utm_source=linkedin&utm_medium=ppc&utm_campaign=2017_tmbg

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., … Yoon, D. H. (2017). *In-Datacenter Performance Analysis of a Tensor Processing Unit TM*.

Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K., & Zhavoronkov, A. (2017). The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, *8*(7), 10883–10890. https://doi.org/10.18632/oncotarget.14073

Liu, Y., Chen, P. H. C., Krause, J., & Peng, L. (2019, November 12). How to Read Articles That Use Machine Learning: Users' Guides to the Medical Literature. *JAMA - Journal of the American Medical Association*, Vol. 322, pp. 1806–1816. https://doi.org/10.1001/jama.2019.16489

References /2

Lowe, D. (2018). Machine Learning: Be Careful What You Ask For. Retrieved November 21, 2019, from https://blogs.sciencemag.org/pipeline/archives/2018/11/20/machine-learning-be-careful-what-you-ask-for

McCandlish, S., Kaplan, J., Amodei OpenAI, D., Brockman, G., Chan, B., Debiak, P., … Zhang, S. (2018). *An Empirical Model of Large-Batch Training*.

McCandlish, S., Kaplan, J., & Amodei, D. (2018). How AI Training Scales. Retrieved February 16, 2020, from https://openai.com/blog/science-of-ai/

Medical Artificial Intelligence Index. (2020). Retrieved February 17, 2020, from http://medicalindex.ai/

MJH Life Sciences. (2000). *Computer Technology Helps Radiologists Spot Overlooked Small Breast Cancers*. Retrieved from https://www.cancernetwork.com/breast-cancer/computer-technology-helps-radiologists-spot-overlooked-small-breast-cancers

Murnane, K. (2017). The Great Strengths and Important Limitations Of Google's Machine Learning Chip. Retrieved February 17, 2020, from https://www-forbes-com.cdn.ampproject.org/c/s/www.forbes.com/sites/kevinmurnane/2017/04/10/the-great-strengths-and-important-limitations-of-googles-machine-learning-chip/amp/?_lrsc=4e4c351d-8b27-4f48-8870-c8ae76bb1081

Nathan, A. (2018). Computational Biomedicine: How data can revolutionize the patient experience. Retrieved February 17, 2020, from http://sitn.hms.harvard.edu/flash/2018/computational-biomedicine-data-can-revolutionize-patient-experience/

Oakden-Rayner, L. (2020). The FDA has approved AI-based PET/MRI "denoising". How safe is this technology? Retrieved February 18, 2020, from https://lukeoakdenrayner.wordpress.com/2019/10/23/the-fda-has-approved-ai-based-pet-mri-denoising-how-safe-is-this-technology/

Oakden-Rayner, L. (2018). CheXNet: an in-depth review. Retrieved February 17, 2020, from https://lukeoakdenrayner.wordpress.com/2018/01/24/chexnet-an-in-depth-review/

Oakden-Rayner, L. (2018). The unreasonable usefulness of deep learning in medical image datasets. Retrieved February 17, 2020, from https://lukeoakdenrayner.wordpress.com/2018/04/30/the-unreasonable-usefulness-of-deep-learning-in-medical-image-datasets/

Oakden-Rayner, L. (2019). Exploring large scale public medical image datasets. *Academic Radiology*, *27*(1), 106–112. Retrieved from http://arxiv.org/abs/1907.12720

Oakden-Rayner, L. (2019). Improving Medical AI Safety by Addressing Hidden Stratification. Retrieved February 17, 2020, from https://lukeoakdenrayner.wordpress.com/2019/10/14/improving-medical-ai-safety-by-addressing-hidden-stratification/

Oakden-Rayner, L. (2019). Half a million x-rays! First impressions of the Stanford and MIT chest x-ray datasets. Retrieved February 17, 2020, from https://lukeoakdenrayner.wordpress.com/2019/02/25/half-a-million-x-rays-first-impressions-of-the-stanford-and-mit-chest-x-ray-datasets/

Oakden-Rayner, L. (2017). The End of Human Doctors – The Bleeding Edge of Medical AI Research (Part 2). Retrieved February 17, 2020, from https://lukeoakdenrayner.wordpress.com/2017/06/05/the-end-of-human-doctors-the-bleeding-edge-of-medical-ai-research-part-2/

Oakden-Rayner, L. (2017). The End of Human Doctors – The Bleeding Edge of Medical AI Research (Part 1). Retrieved February 17, 2020, from https://lukeoakdenrayner-wordpress-com.cdn.ampproject.org/c/s/lukeoakdenrayner.wordpress.com/2017/05/24/the-end-of-human-doctors-the-bleeding-edge-of-medical-ai-research-part-1/amp/?utm_content=bufferf08a6&utm_medium=social&utm_source=linkedin.com&utm_cam

Oakden-Rayner, L., Dunnmon, J., Carneiro, G., & Ré, C. (2019). *Hidden Stratification Causes Clinically Meaningful Failures in Machine Learning for Medical Imaging*. Retrieved from http://arxiv.org/abs/1909.12475

Qiu, S., Liu, Q., Zhou, S., & Wu, C. (2019). Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences (Switzerland)*, *9*(5). https://doi.org/10.3390/app9050909

Rajpurkar, P., Irvin, J., Ball, R. L., Zhu, K., Yang, B., Mehta, H., … Lungren, M. P. (2018). Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. *PLoS Medicine*, *15*(11). https://doi.org/10.1371/journal.pmed.1002686

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., … Ng, A. Y. (2017). *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning*. Retrieved from http://arxiv.org/abs/1711.05225

References /3

Roy, A. G., Conjeti, S., Navab, N., & Wachinger, C. (2018). QuickNAT: A Fully Convolutional Network for Quick and Accurate Segmentation of Neuroanatomy. *NeuroImage*, *186*, 713–727. https://doi.org/10.1016/j.neuroimage.2018.11.042

Rupp, K. (2018). Microprocessor Trend Data. Retrieved February 17, 2020, from https://github.com/karlrupp/microprocessor-trend-data

Rupp, K. (2018). 42 Years of Microprocessor Trend Data. Retrieved February 17, 2020, from https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Rupp, K. (2015). 40 Years of Microprocessor Trend Data. Retrieved February 17, 2020, from https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/

Salian, I. (2020). DeepTek Detects Tuberculosis from X-Rays with AI. Retrieved February 17, 2020, from https://blogs.nvidia.com/blog/2020/02/11/deeptek-ai-tuberculosis-detection-xrays/?ncid=so-elev-52985#cid=ix11_so-elev_en-us

Sato, D., Wider, A., & Windheuser, C. (2019). Continuous Delivery for Machine Learning. Retrieved February 20, 2020, from https://martinfowler.com/articles/cd4ml.html

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., … Collins, J. J. (2020). A Deep Learning Approach to Antibiotic Discovery. *Cell*, *180*(4), 688-702.e13. https://doi.org/10.1016/j.cell.2020.01.021

Subtle Medical Inc. (2020). Subtle Medical Receives FDA 510(k) Clearance for AI-Powered SubtleMR. Retrieved February 18, 2020, from https://www.prnewswire.com/news-releases/subtle-medical-receives-fda-510k-clearance-for-ai-powered-subtlemr-300938602.html

Villain, N. (2019). The role of Artificial Intelligence in medical imaging: from research to clinical routine. https://www.aphc.info/wp-content/uploads/2019/01/pre232-villain-nicolas.pdf

Walters, P. (n.d.). Dissecting the Hype With Cheminformatics. Retrieved November 21, 2019, from https://practicalcheminformatics.blogspot.com/2019/09/dissecting-hype-with-cheminformatics.html

Wang, P., Berzin, T. M., Glissen Brown, J. R., Bharadwaj, S., Becq, A., Xiao, X., … Liu, X. (2019). Real-time automatic detection system increases colonoscopic polyp and adenoma detection rates: A prospective randomised controlled study. *Gut*, *68*(10), 1813–1819. https://doi.org/10.1136/gutjnl-2018-317500

Wikipedia. (2020). List of animals by number of neurons. https://en.wikipedia.org/wiki/List_of_animals_by_number_of_neurons

Wolff, M. (2019). I applied AI to my arthritis assessment. Here's what happened. Retrieved February 17, 2020, from https://blogs.sas.com/content/sascom/2019/10/29/i-applied-ai-to-my-arthritis-assessment-heres-what-happened/

Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful Image Colorization. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9907 LNCS, 649–666. http://arxiv.org/abs/1603.08511

Zhavoronkov, A. (2019). The Rise of the Deep: Eric Topol's Deep Medicine To Stand The Test Of Time. Retrieved February 17, 2020, from https://www.forbes.com/sites/cognitiveworld/2019/10/03/the-rise-of-the-deep-eric-topols-deep-medicine-to-stand-the-test-of-time/?fbclid=IwAR0WV7Tx_f5iAR3OgadReUpNRNrz-mMpDVuF0-rrE0HVgqqVIsRjpMWg-qI#6228d99f4e4c