



лекция

Edge inference NPU?

спикер

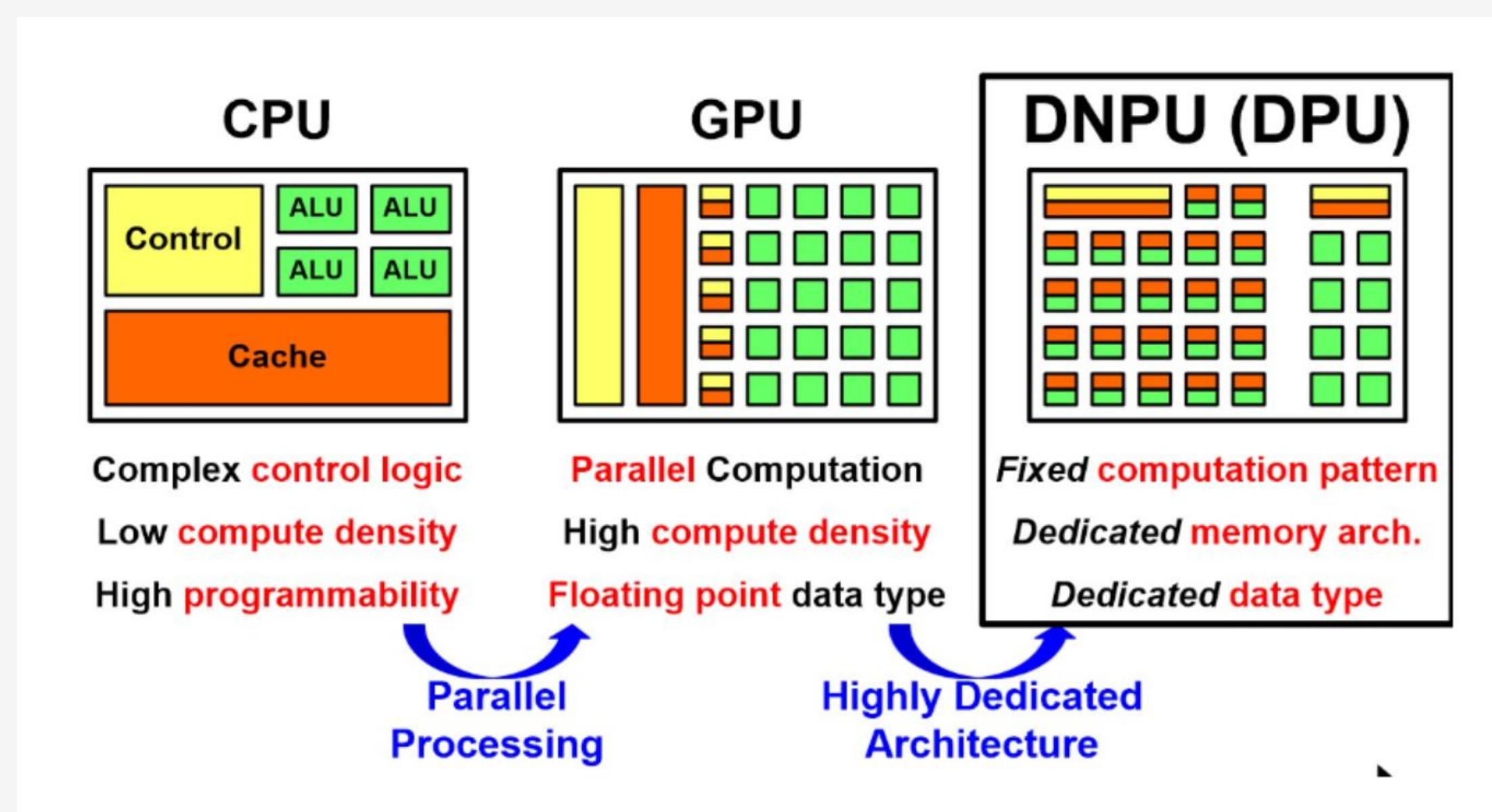


Антон
Мальцев



rembrain.ai

В чем разница между NPU CPU GPU



Зачем NPU?

Цель NPU не в том чтобы ускорить вычисления

Цель NPU

- Разгрузить вычислитель
 - Оптимизировать энергопотребление
 - Снизить стоимость устройства
-

NPU усложняет работу

- Меньше сетей доступно
 - Сложнее экспортировать сеть
 - Более сложная инфраструктура
-

Результат

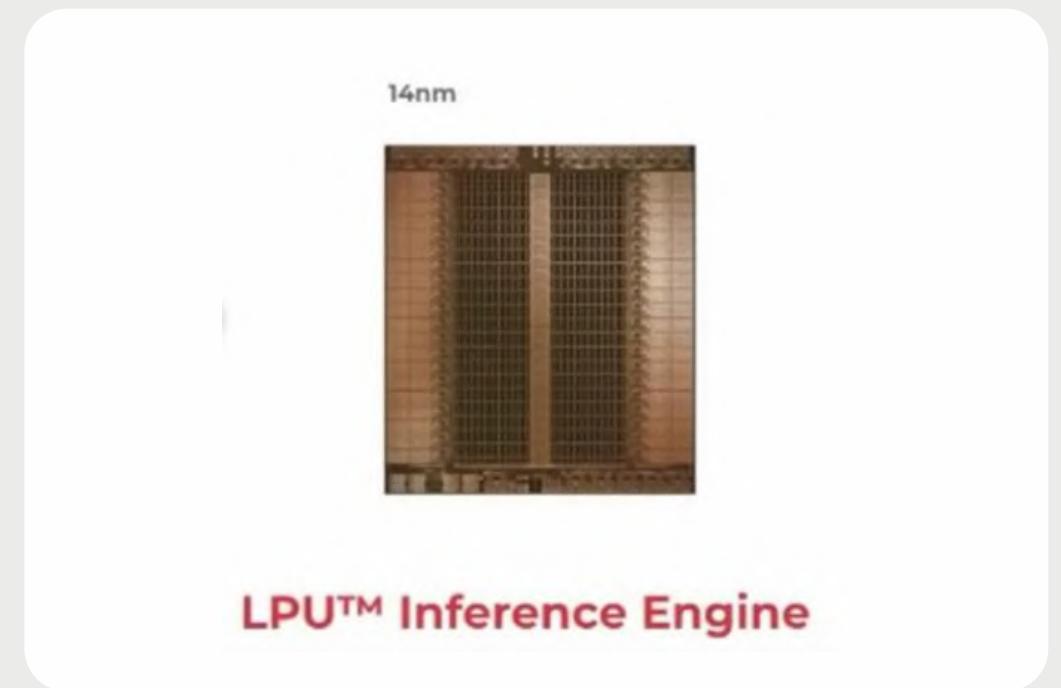
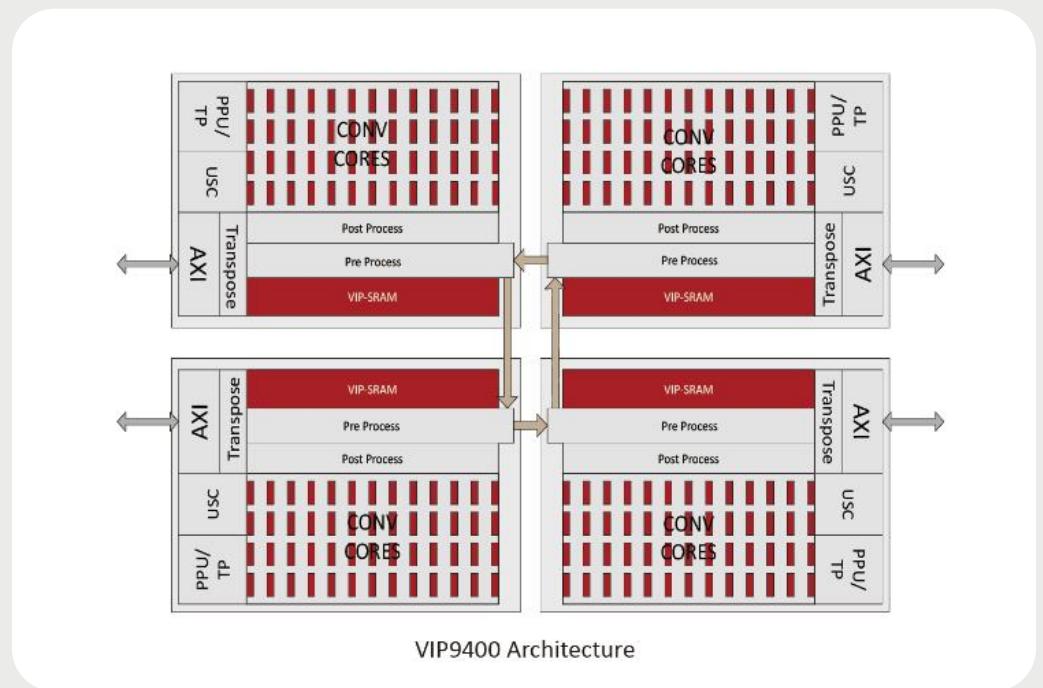
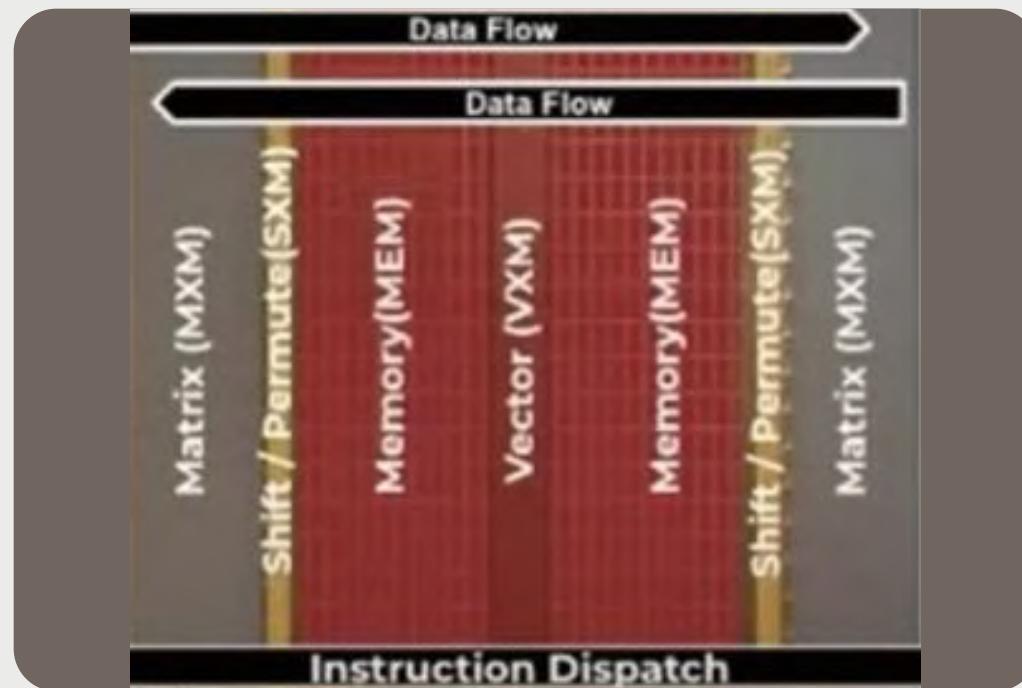
- Разработка дороже
 - Устройство дешевле
 - Потребление меньше
-

Название?

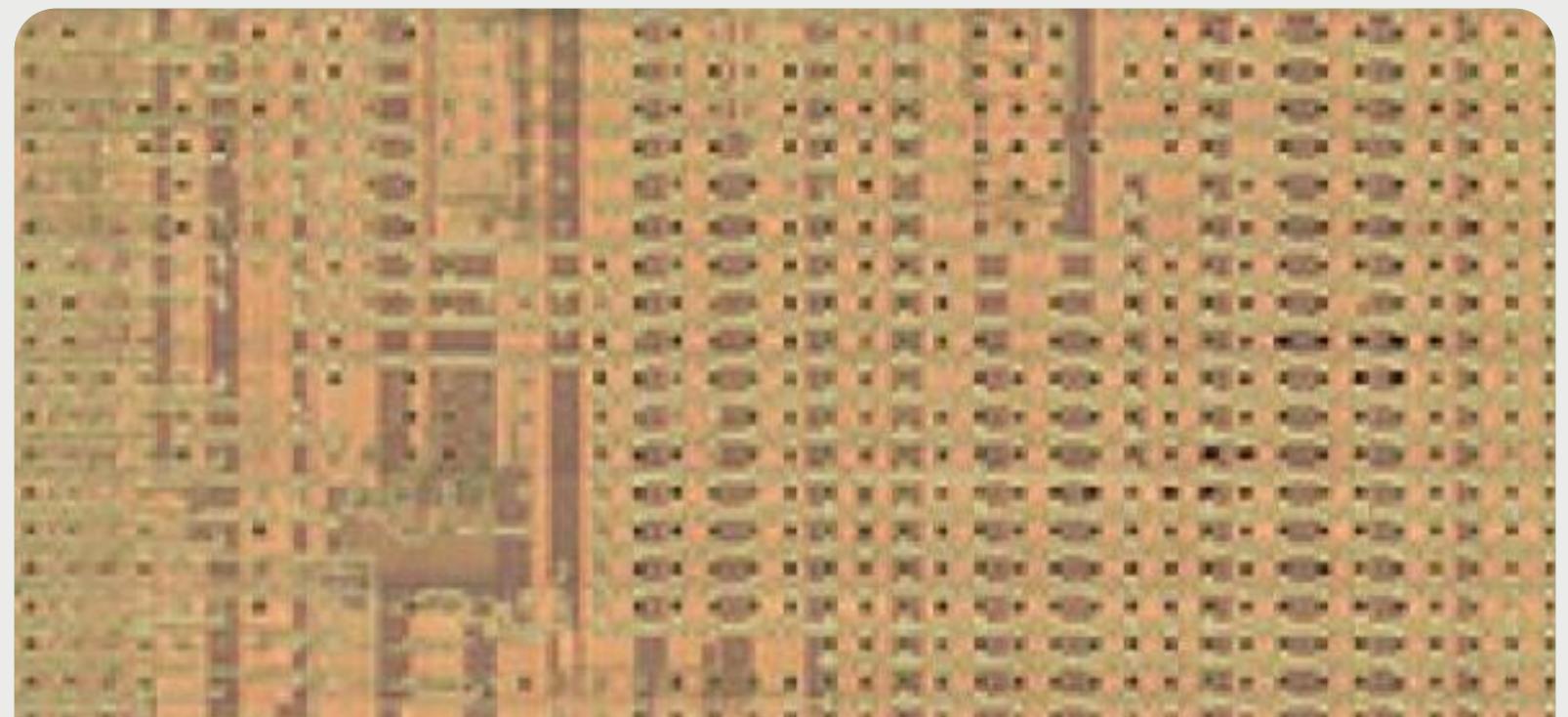
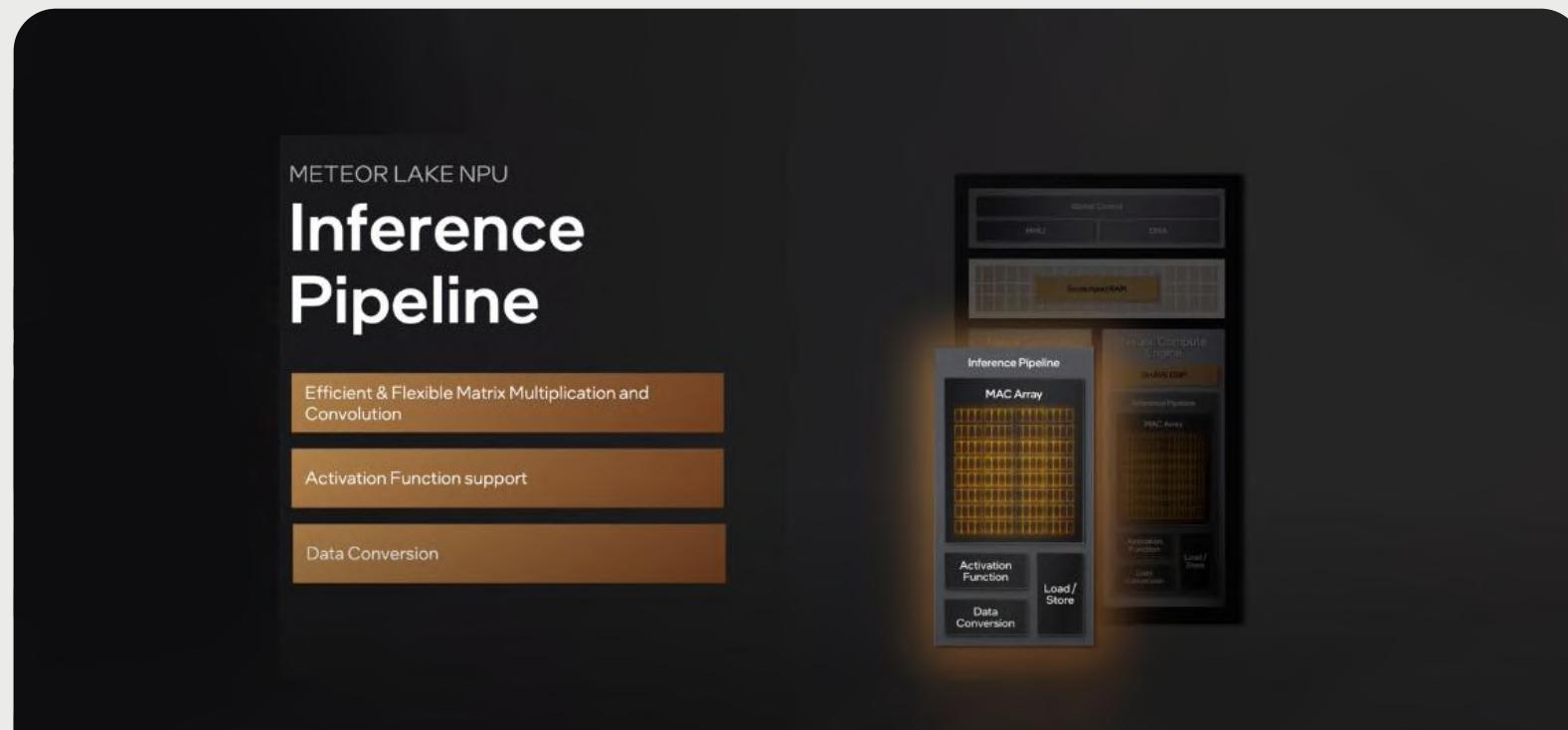
NPU	→	Neural Processing Unit
LPU	→	Language processing unit
TPU	→	Tensor Processing Unit
VPU	→	Versatile Processing Unit
DLA	→	Deep Learning Accelerator

BPU	→	Brain Processing Unit
DPU	→	Deep Learning Processing Unit
IPU	→	Intelligence Processing Unit
VPU	→	Vision Processing Unit
...		

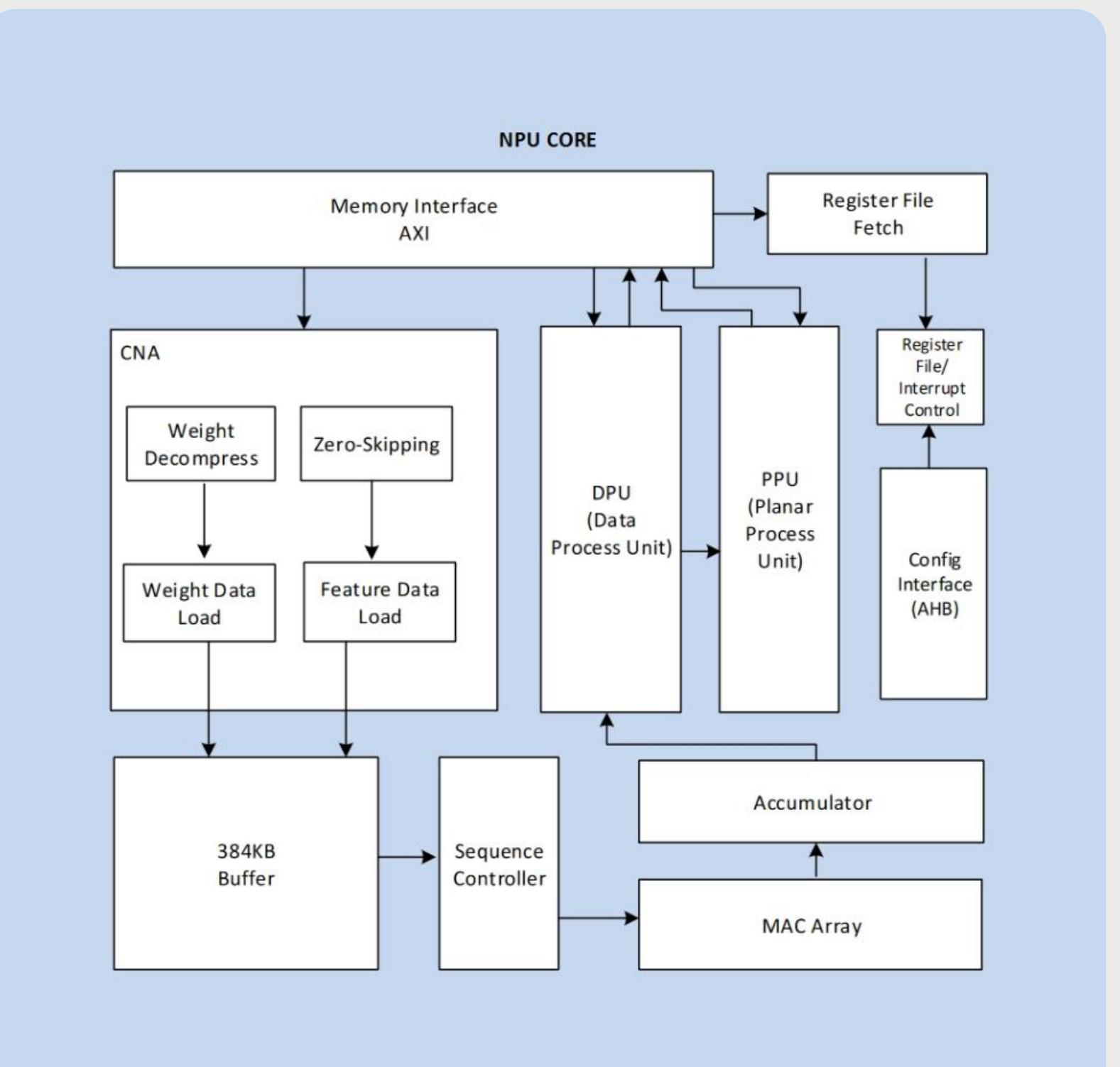
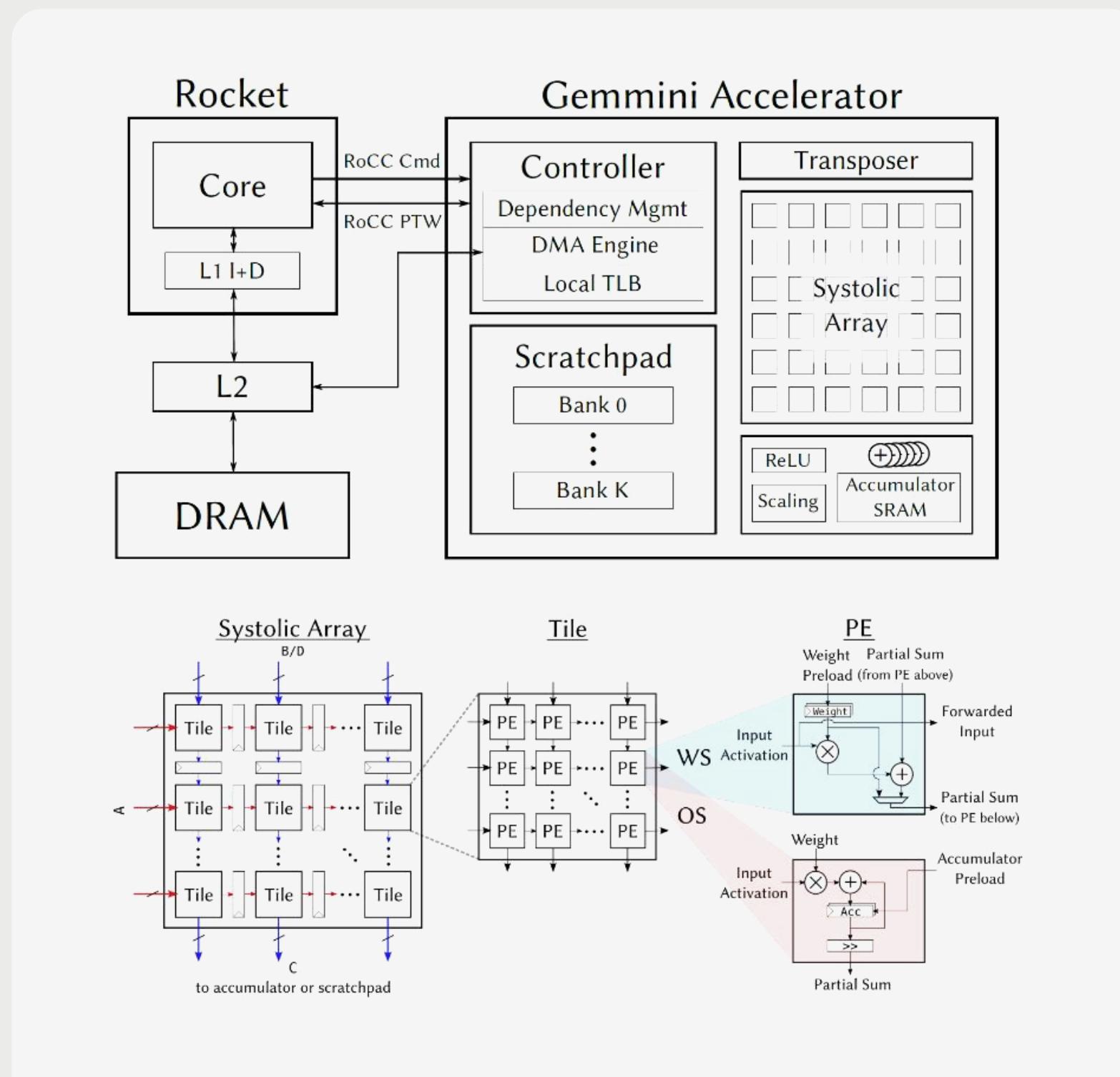
А есть ли разница?



LPU™ Inference Engine



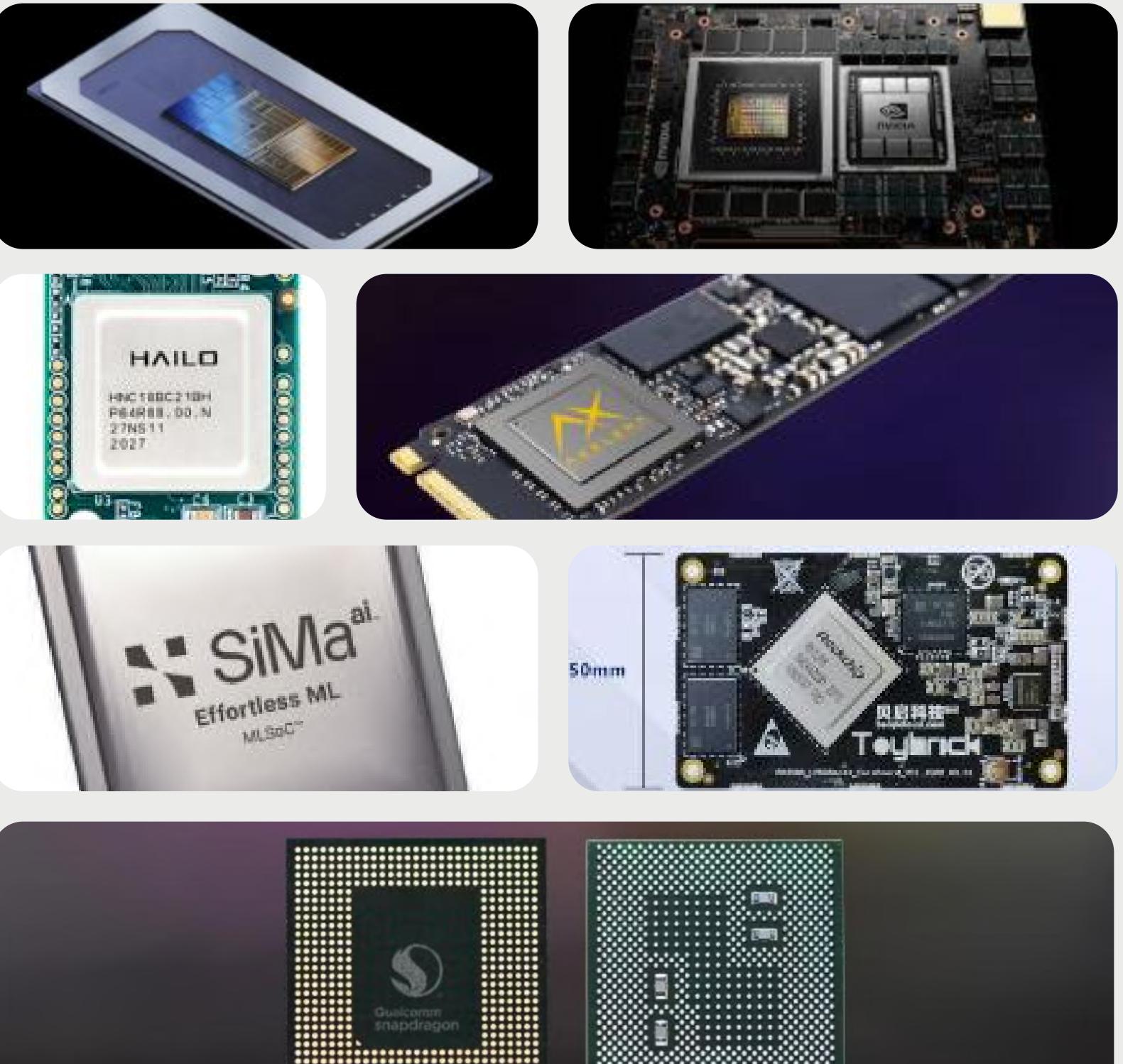
Схемы работы



А кто делает?

- Intel
- Nvidia
- Hailo
- Axelera
- SiMa
- RockChip
- Qualcomm
- VeriSilicon
- Amlogic / STM /
NXP / BroadCom /
Synaptics
- ...

- Texas Instruments
- MediaTek
- Ambarella
- SiFive
- Sophone
- BrainChip
- Analog Devices
- Nordic
- Semiconductor
- Axera

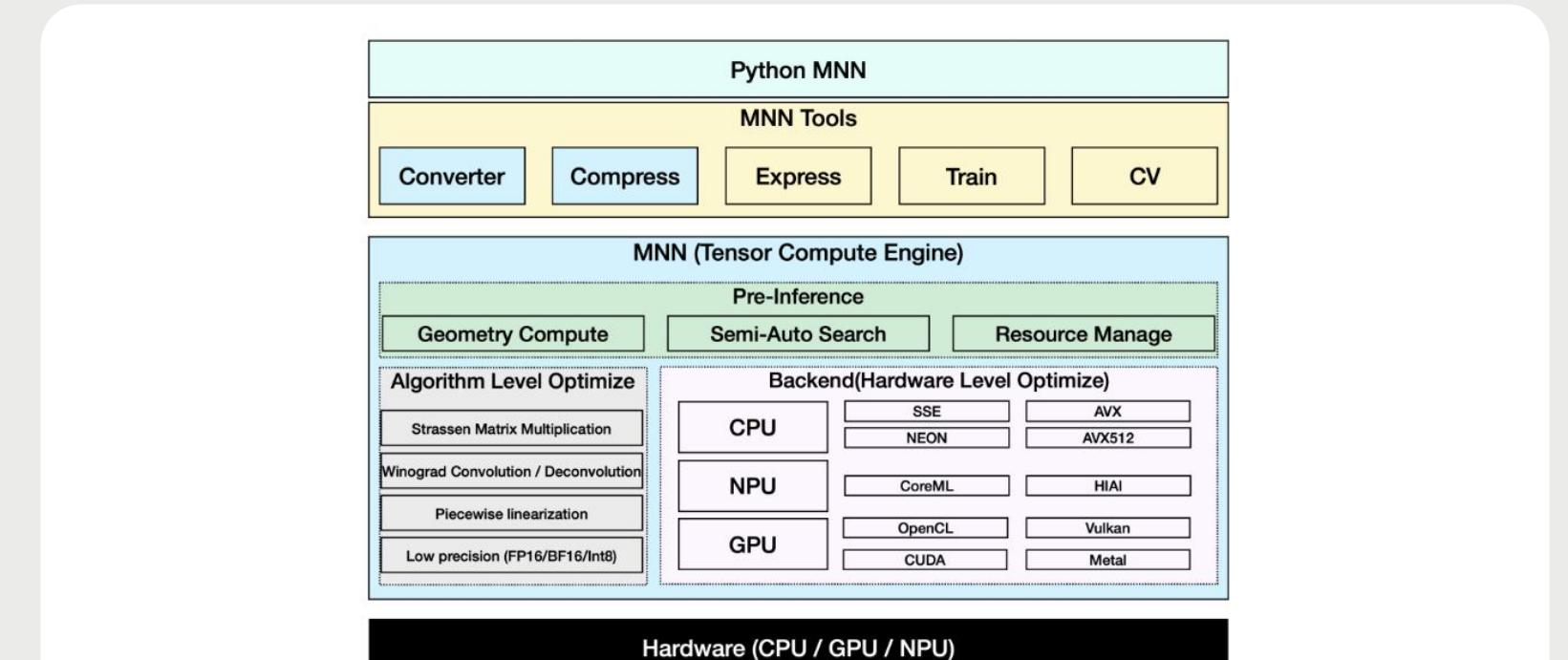
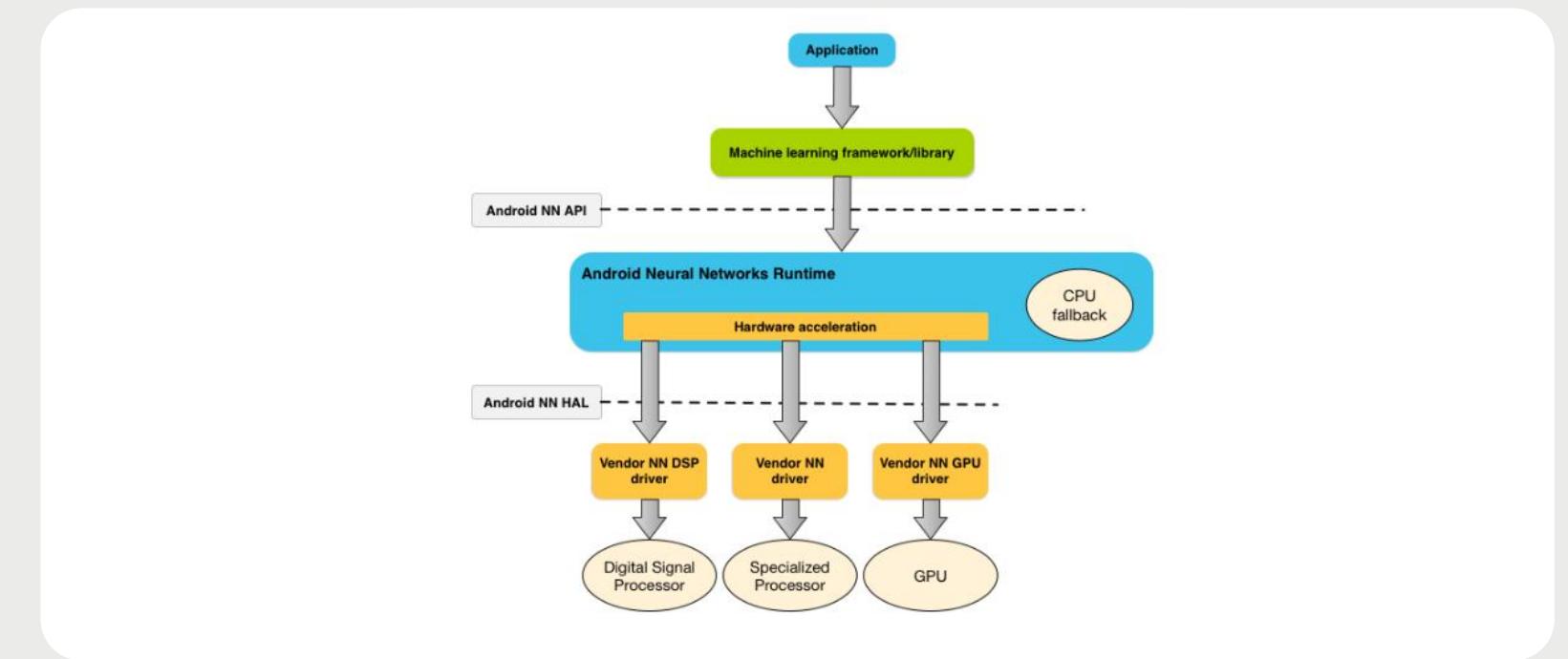


Что у кого есть

	CPU	GPU	NPU
Intel	✓	?	✓
Jetson	✓	✓	✓
Hailo / Axelera / Sima	✗	✗	✓
RockChip	✓	✗	✓
Qualcomm	✓	✓	✓
VeriSilicon	?	?	✓
Ti	✓	✗	✓
...

Фреймворки?

- TensorFlow Lite
- ONNX Runtime
- MNN
- Tengine
- Кастомные...
Nvidia / OpenVino /
SnapDragon / RKNN
Toolkit / HailoRT / etc...
- Уровень поддержки
- Скорость до и после
- Число поддерживаемых слоев
- Языки



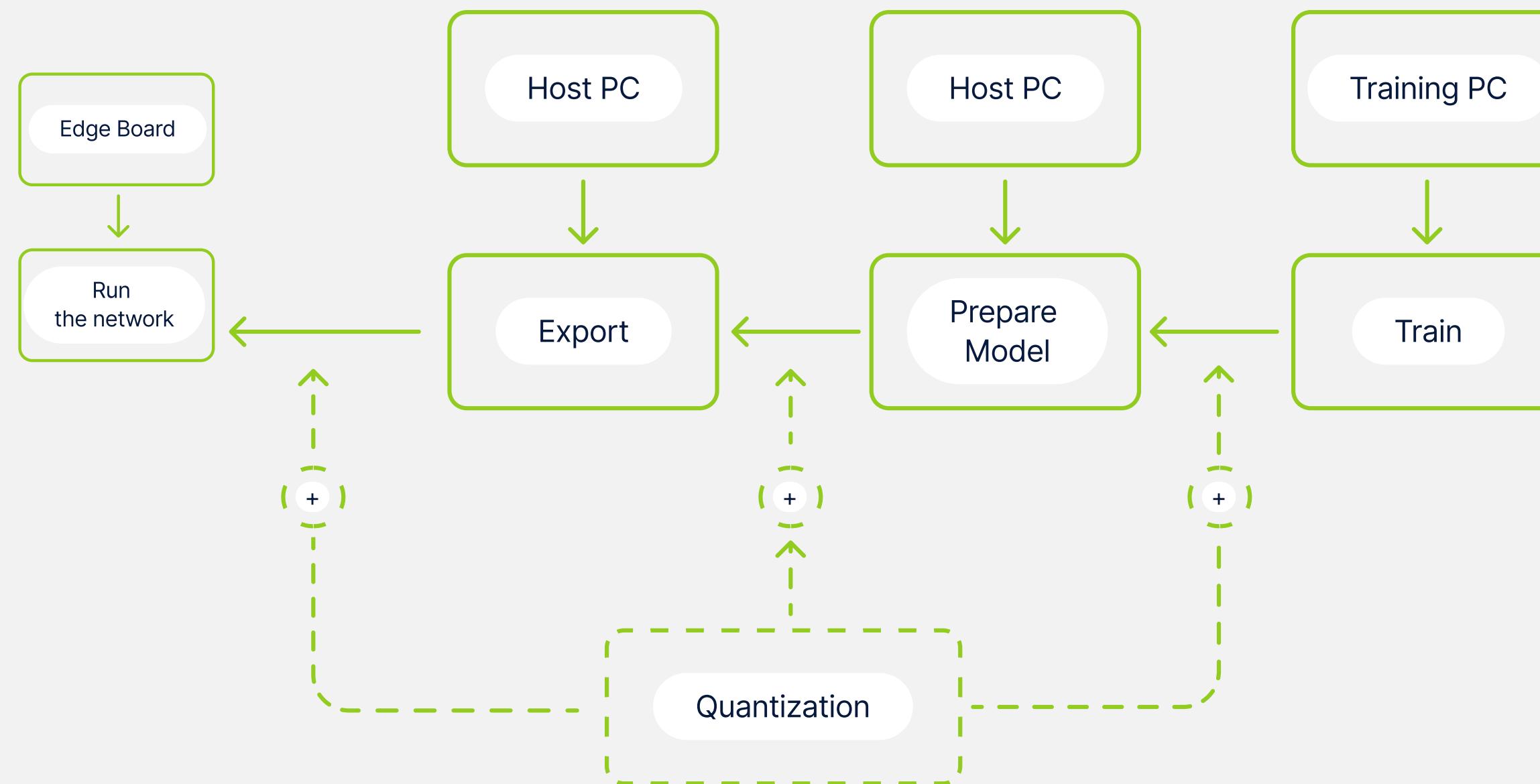
Экспорт как устроен

-
- Чаще всего классический пайплайн → Натренировать модель
→ Пропатчить слои (SiLU -> ReLu, etc.)
→ Выгрузить ONNX/Torch Script/TFLite
→ Экспорт на хост машине
→ Инференс на целевой плате
-

- Поддержка классического фреймворка → ONNXRuntime
→ TFlite
→ PyTorch
-

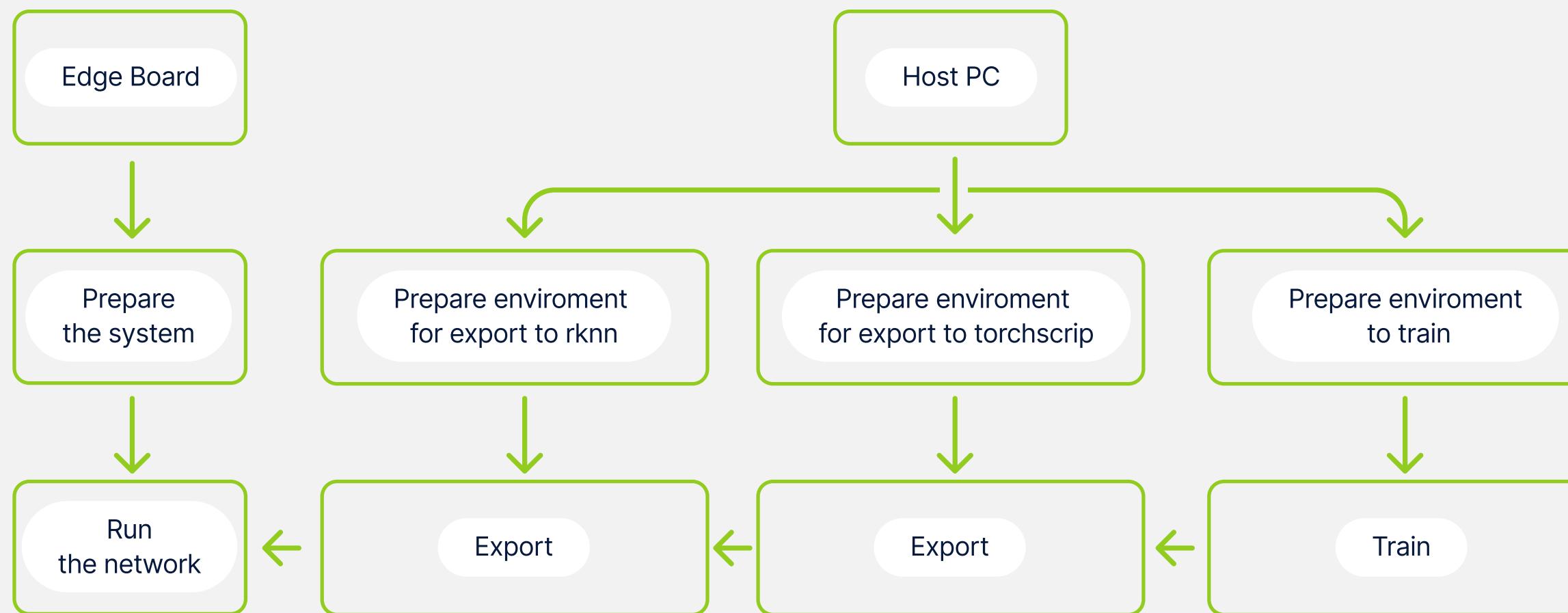
Прочее TF Micro, custom, FPGA, etc

Классика

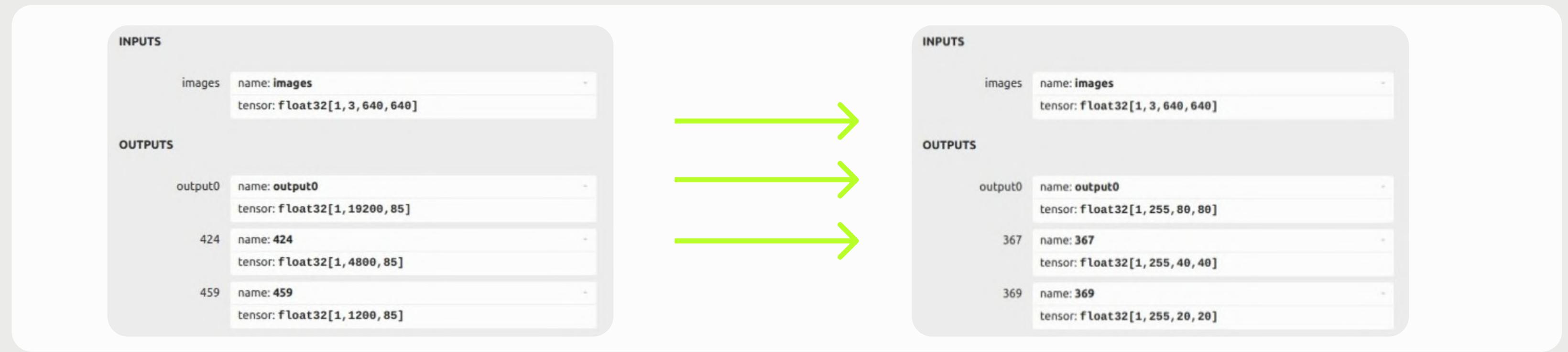


Пример RockChip

- Нет готовых контейнеров
- Минимум три энвайронаента



Пример RockChip



- ① Optimize focus/SPPF block, getting better performance with same result
- ② Using ReLU as activation layer instead of SiLU. Only when training new model

- ① Optimize focus/SPPF block, getting better performance with same result
- ② Change output node, remove post_process from the model. Post process block in model is unfriendly for quantization

Пример RockChip

```
# for detection model
python export.py --rknpu --weight yolov5s.pt

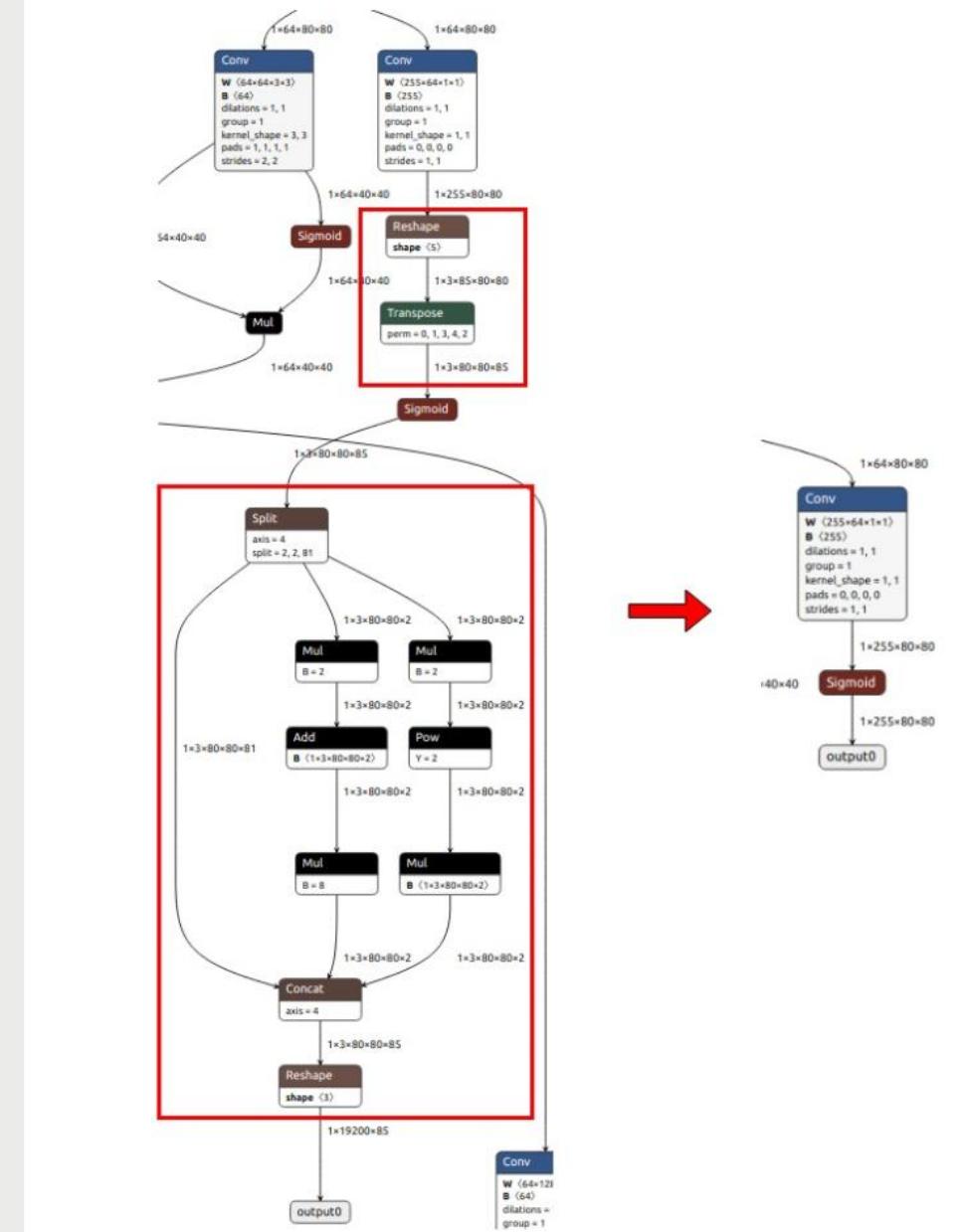
# for segmentation model
python export.py --rknpu --weight yolov5s-seg.pt
```

[github.com/airockchip/yolov5/ ↗](https://github.com/airockchip/yolov5/)

```
cd python
python convert.py <onnx_model> <TARGET_PLATFORM> <dtype(optional)> <output_rknn_path(optional)>

# such as:
python convert.py ./model/yolov5s_relu.onnx rk3566
# output model will be saved as ./model/yolov5.rknn
```

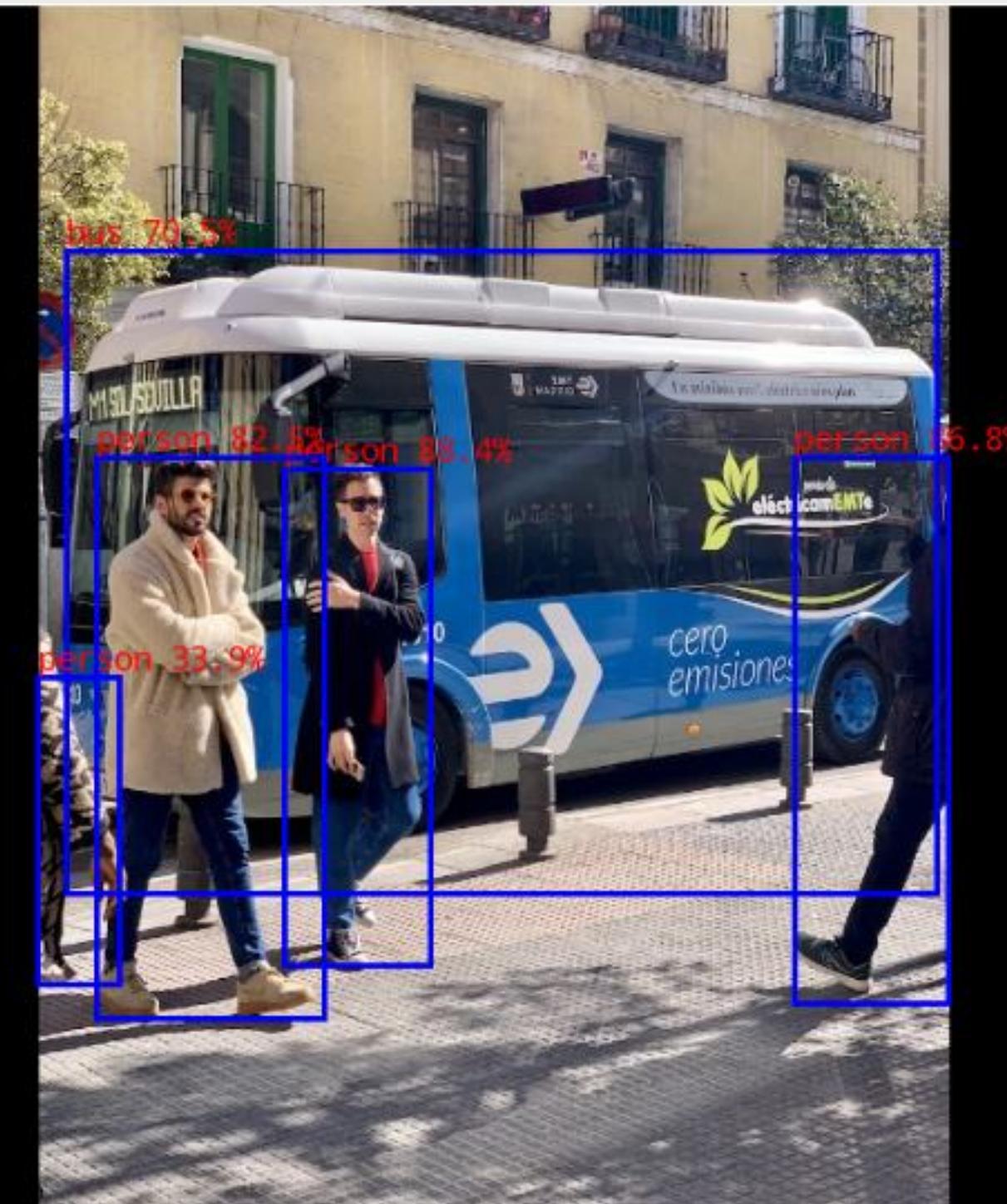
[github.com/airockchip/rknn_model_zoo/tree/main/examples/yolov5 ↗](https://github.com/airockchip/rknn_model_zoo/tree/main/examples/yolov5)



Пример RockChip

```
cd python
# Inference with PyTorch model or ONNX model
python yolov5.py --model_path <pt_model/onnx_model> --img_show

# Inference with RKNN model
python yolov5.py --model_path <rknn_model> --target <TARGET_PLATFORM> --img_show
```



Пример RockChip

```
def dfl(position):
    # Distribution Focal Loss (DFL)
    import torch
    x = torch.tensor(position)
    n,c,h,w = x.shape
    p_num = 4
    mc = c//p_num
    y = x.reshape(n,p_num,mc,h,w)
    y = y.softmax(2)
    acc_metrix = torch.tensor(range(mc)).float().reshape(1,1,mc,1,1)
    y = (y*acc_metrix).sum(2)
    return y.numpy()
```

```
def softmax(x, axis=None):
    x = x - x.max(axis=axis, keepdims=True)
    y = np.exp(x)
    return y / y.sum(axis=axis, keepdims=True)

def dfl(position):
    # Distribution Focal Loss (DFL)
    n,c,h,w = position.shape
    p_num = 4
    mc = c//p_num
    y = position.reshape(n,p_num,mc,h,w)
    y = softmax(y,2)
    acc_metrix = np.array(range(mc), dtype=float).reshape(1,1,mc,1,1)
    y = (y*acc_metrix).sum(2)
    return y
```

Пример Hailo

HailoRT

→ Подготовка модели в PyTorch

Обучение, замена слоев

→ Экспорт модели в ONNX

→ Экспорт модели в HAR

Под капотом ONNX → TF /

Под капотом экспорт TF → HAR

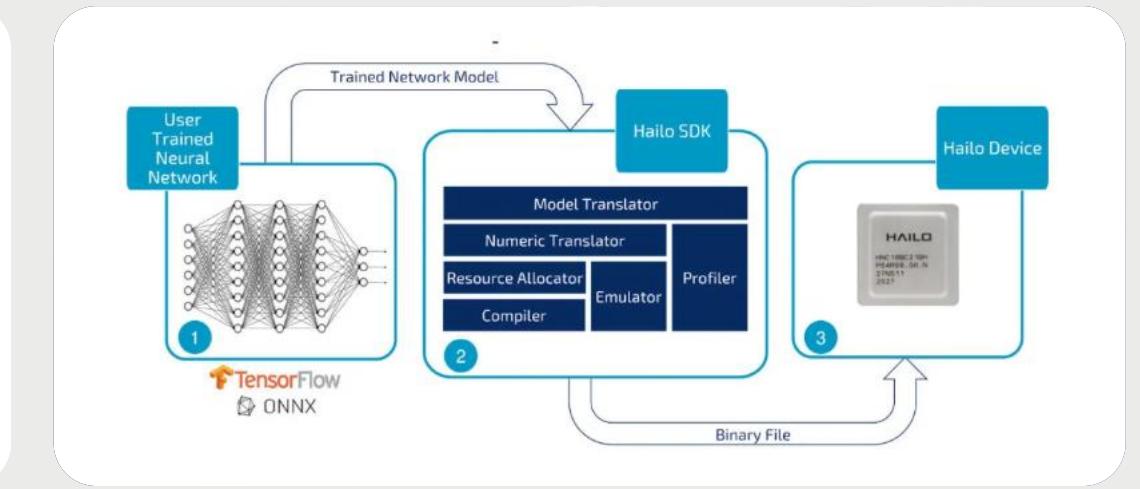
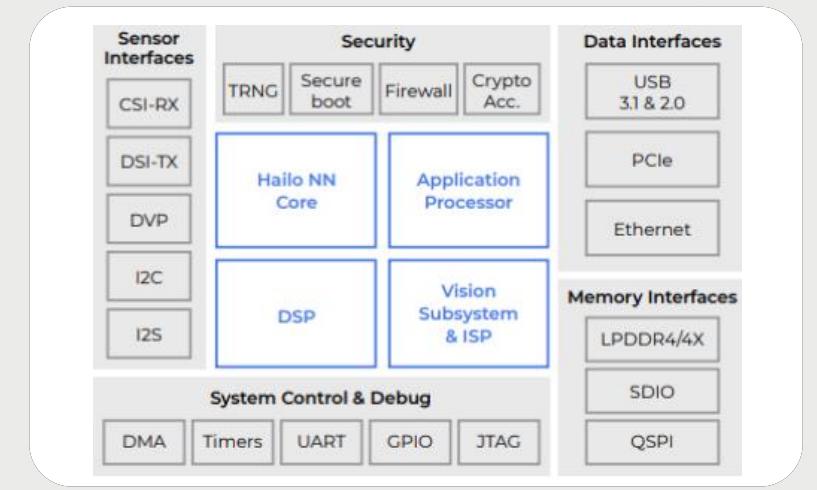
→ Модификация модели

Вход-выход

→ Калибровка модели

→ Компиляция модели HEF

→ Запуск модели



Пример ESP32

1

Espressif C++ ToolKit

2

Random library from random dude

3

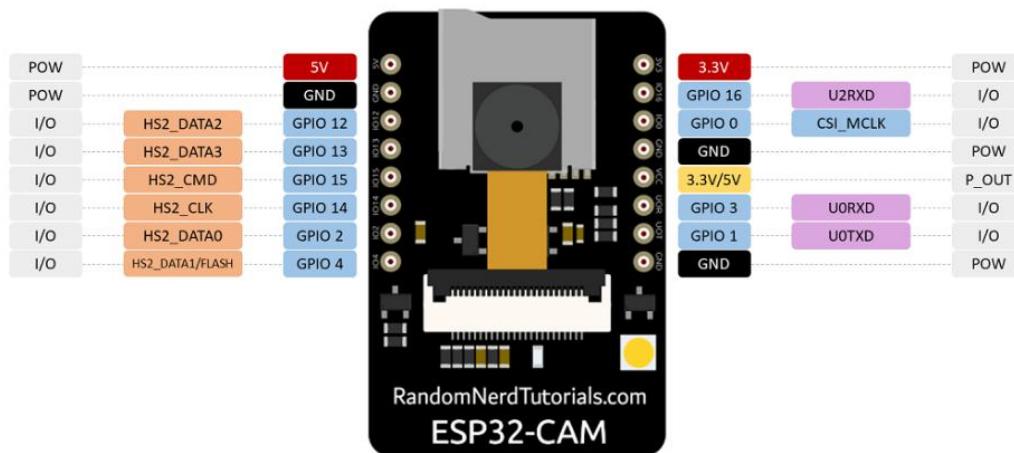
TFMicro official c++ example

4

MicroPython

5

Edge Impulse



Проблемы: НМС

Есть встроенный

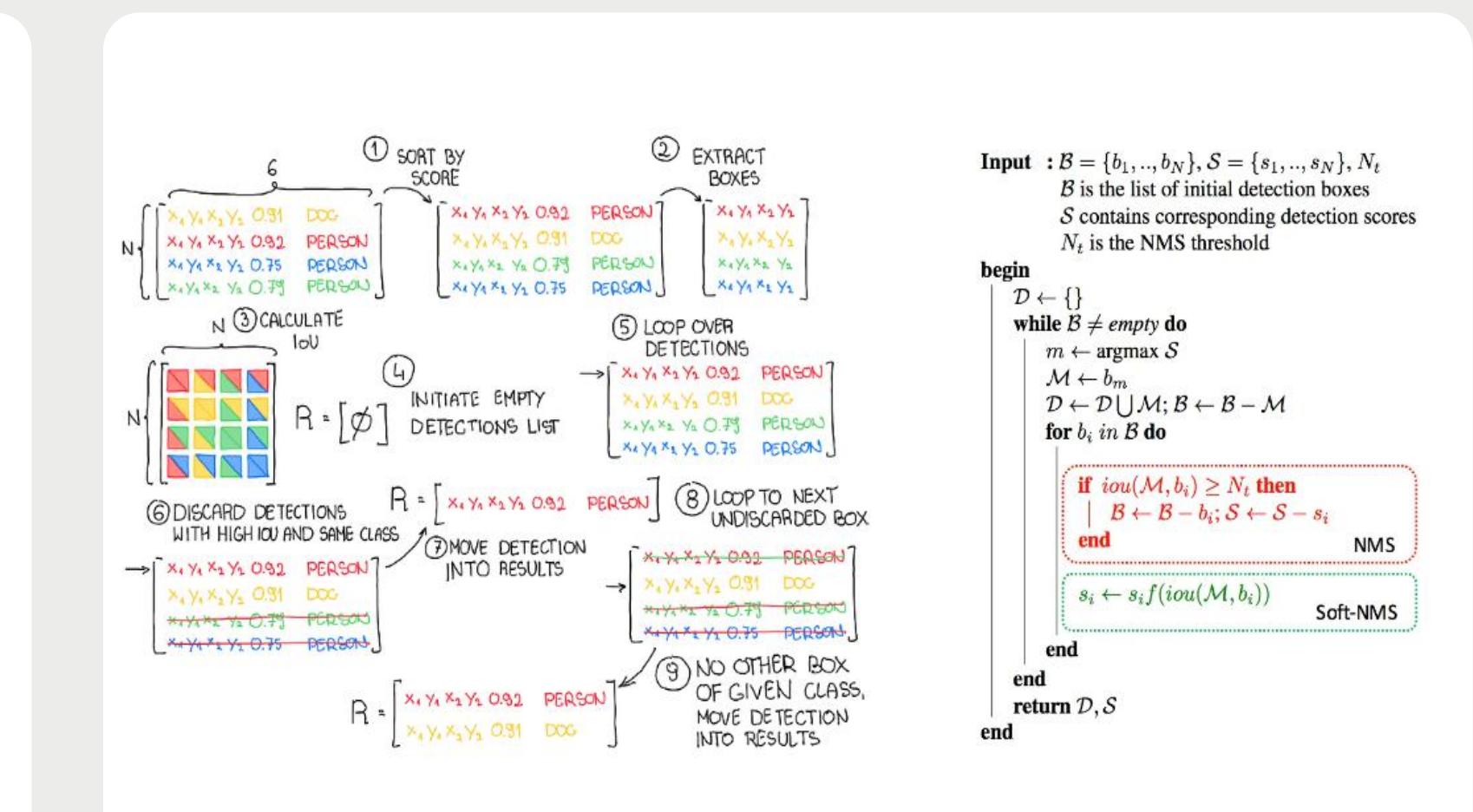
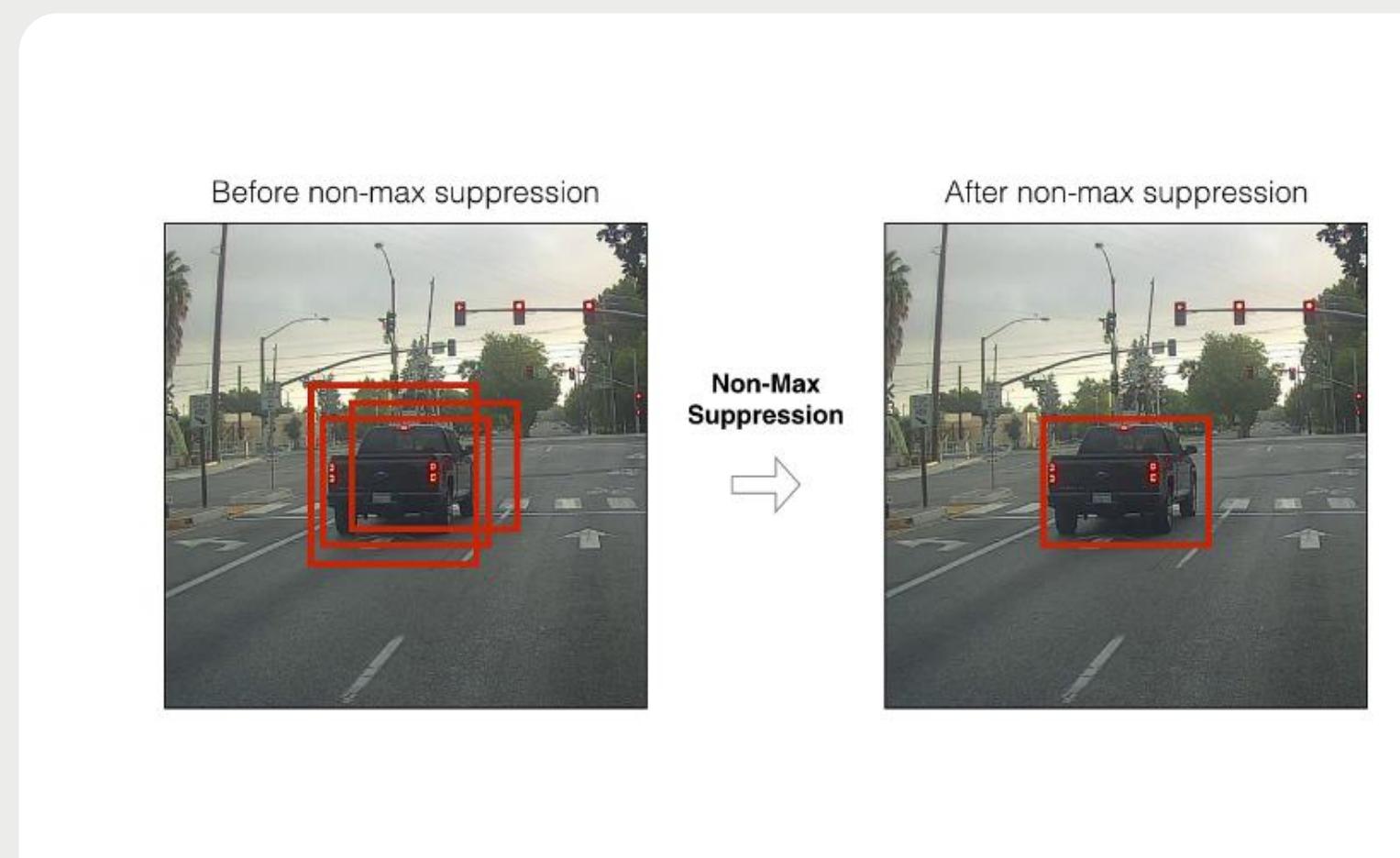
→ Jetson (TRT) / OpenVino NMS / Бывает ещё, но редко

Корректно парсится системой но на CPU

→ STM32 / Axelera / Etc... становится популярно

Некорректно парсится

→ Надо отрезать и сделать независимо:
Hailo / RockChip / etc...



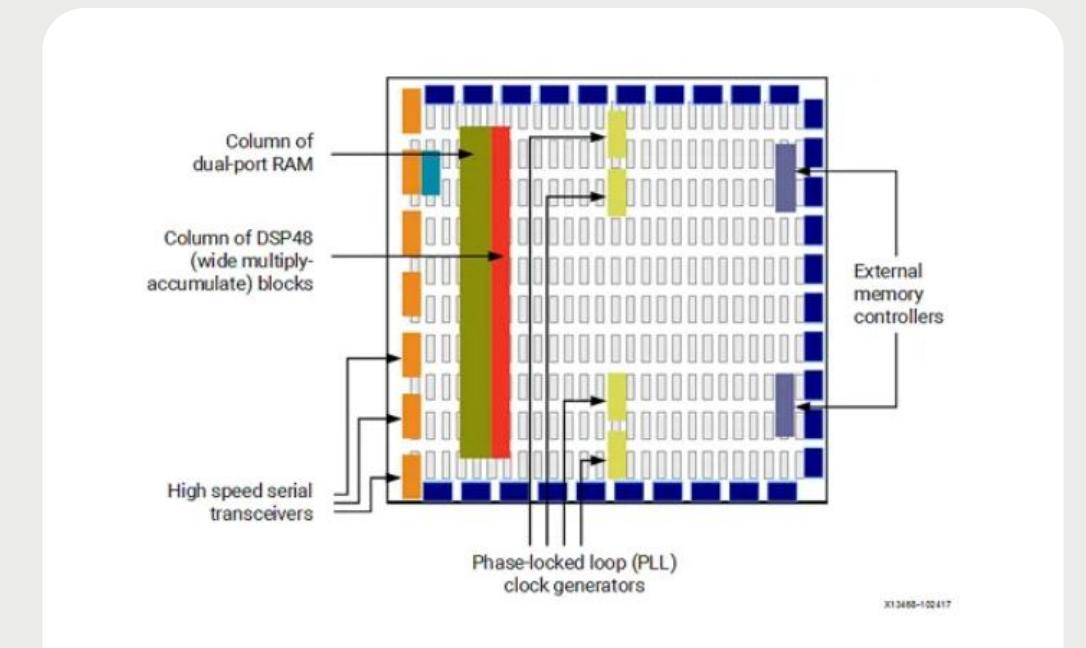
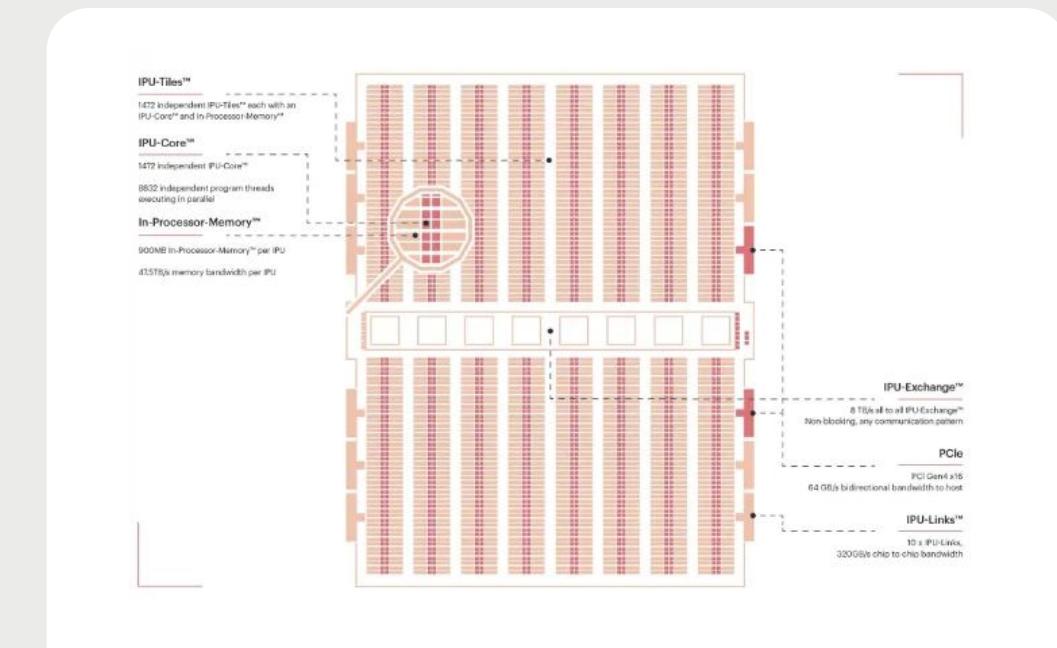
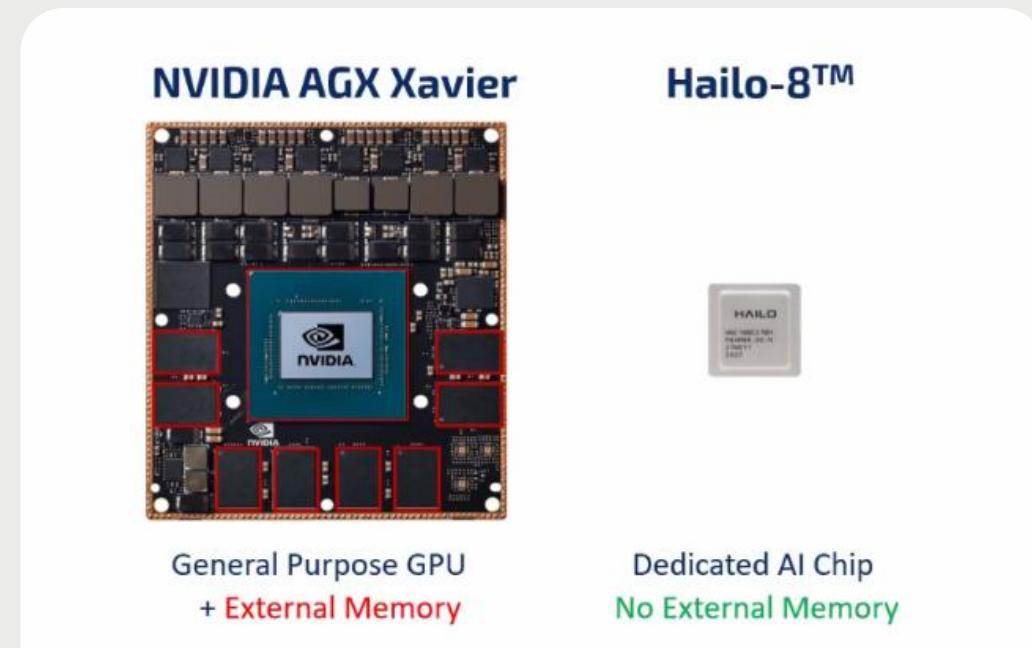
Проблемы: Размер входа

Ограничение размера со стороны фреймворка на размер изображения

- Память на NPU (Hailo/SiMa), ограничение размера, динамическая подгрузка:
Сеть не более нескольких сотен мегабайт /
Работает только с небольшими картинками
- Размер матрицы в операции (Hailo/RockChip):
Проблемы с некоторыми трансформерами

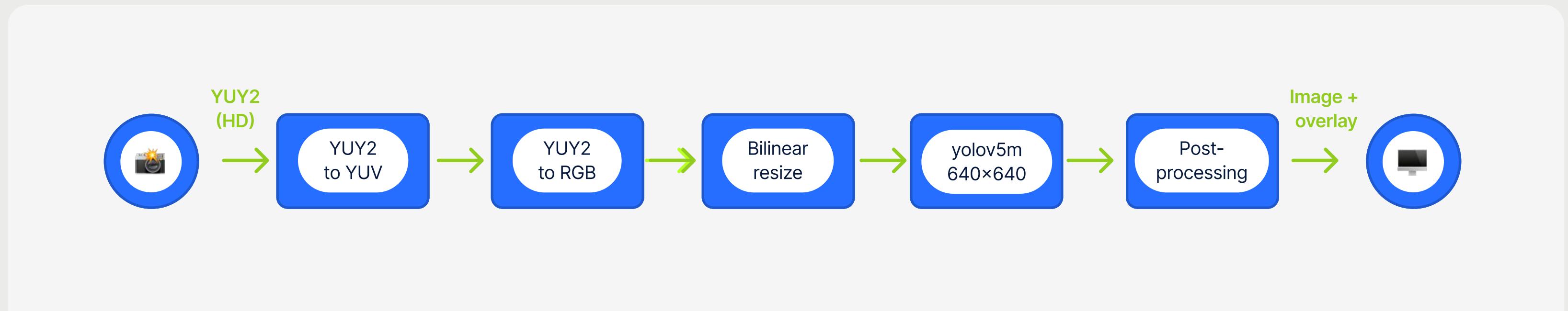
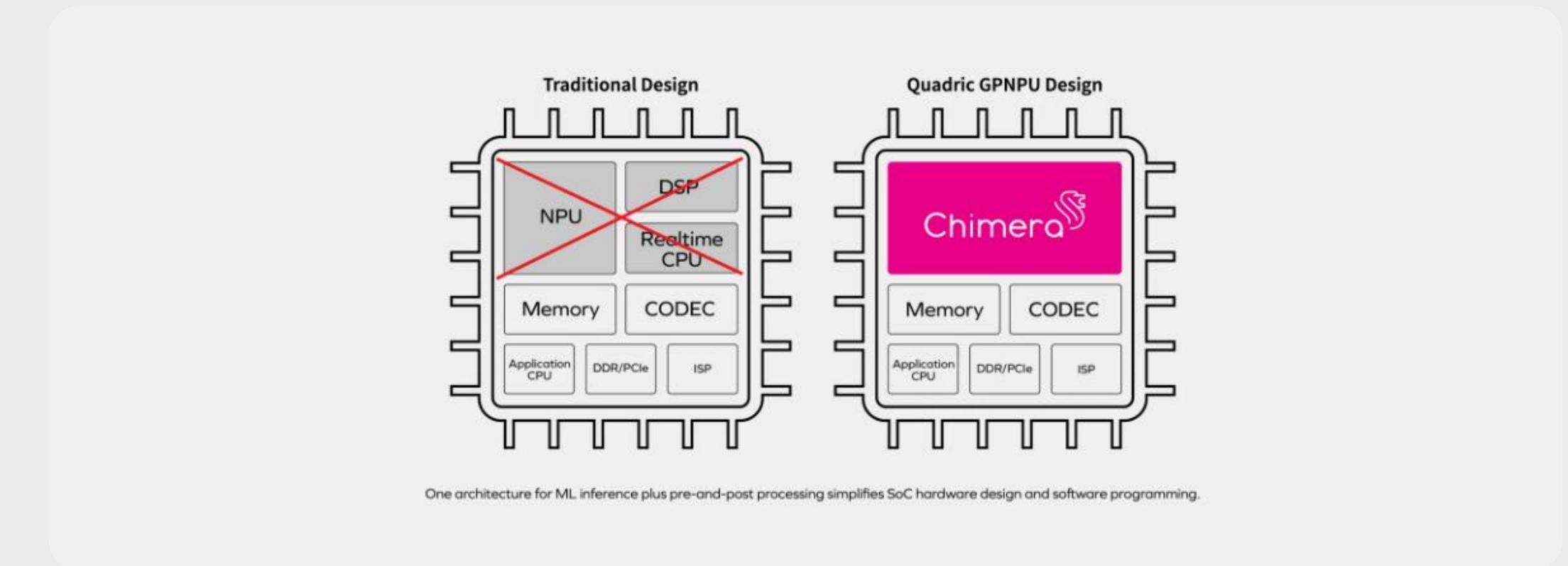
Число каналов и слоев

- Для некоторых вычислителей ограничение числа доступных слоев, сильное замедление работы



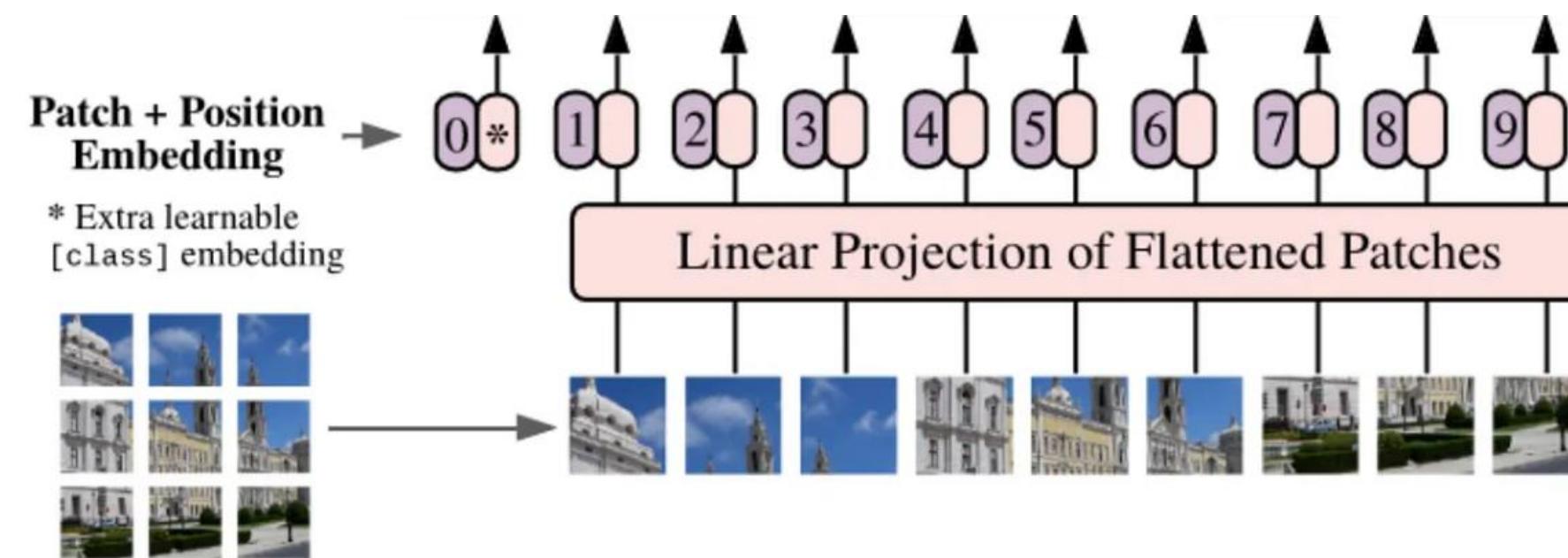
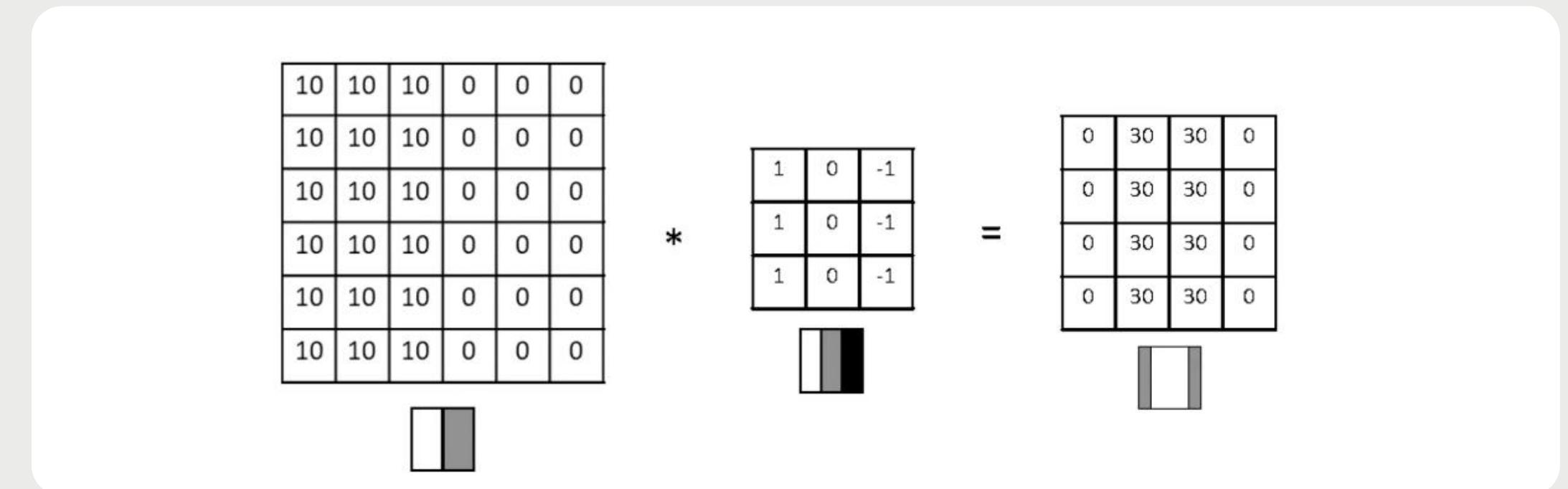
Проблемы: Препроцессинг

- ? На CPU?
- ? На сопроцессоре?
- ? На NPU?
- ? Что можно применить, особенности?



Проблемы: Трансформеры

- 1 Последовательность и тип операторов
- 2 Архитектурные предположения при разработке NPU
- 3 Нормализация, Softmax



Проблемы: Слои

Число слоёв поддерживаемых у всех разное

→ ONNX 193 / TensorRT ~37 / RKNN Toolkit 2 ~46 / TensorFlow Edge TPU ~40 / Hailo ~34

Ограничение функционала слоёв

→ Размерность слоя / Максимальный размер / Числовой тип слоя / Разные железки могут поддерживать по-разному

Conv3D is supported with the following parameters:

3D Convolution supported parameters

Kernel (HxWxD)	Stride (HxWxD)	Dilation (HxWxD)	Padding	Notes
3x3x3	1x1x1	1x1x1	SAME SAME_TENSORFLOW VALID	PREVIEW
3x3xAny	1x1xAny	1x1x1	SAME SAME_TENSORFLOW VALID	PREVIEW

Convolution kernel optimized parameters %

Kernel (HxW)	Stride (HxW)	Dilation (HxW)	Padding
1x1	1x1, 2x1, 2x2	1x1	SAME SAME_TENSORFLOW VALID
3x3	1x1, 1x2, 2x1, 2x2	1x1 2x2 (stride=1x1 only) 3x3 (stride=1x1 only) 4x4 (stride=1x1 only) 6x6 (stride=1x1 only) 8x8 (stride=1x1 only) 16x16 (stride=1x1 only)	SAME SAME_TENSORFLOW VALID
2x2	2x1	1x1	SAME SAME_TENSORFLOW VALID
2x2, 2x3, 2x5, 2x7, 3x2, 5x2, 7x2	2x2	1x1	SAME SAME_TENSORFLOW
5x5, 7x7	1x1, 1x2, 2x1, 2x2	1x1	SAME SAME_TENSORFLOW VALID (stride=1x1 only)
6x6	2x2	1x1	SAME SAME_TENSORFLOW VALID
1x3, 1x5, 1x7	1x1, 1x2	1x1	SAME SAME_TENSORFLOW VALID (stride=1x1 only)
3x5, 3x7, 5x3, 5x7, 7x3, 7x5	1x1, 1x2	1x1	SAME SAME_TENSORFLOW VALID (stride=1x1 only)

3x1	1x1, 2x1	1x1	SAME SAME_TENSORFLOW VALID
5x1, 7x1	1x1	1x1	SAME SAME_TENSORFLOW VALID
1x9, 3x9, 5x9, 7x9	1x1	1x1	SAME SAME_TENSORFLOW VALID
9x1, 9x3, 9x5, 9x7, 9x9	1x1	1x1	SAME SAME_TENSORFLOW VALID
3x4, 5x4, 7x4, 9x4	1x1	1x1	SAME SAME_TENSORFLOW VALID
3x6, 5x6, 7x6, 9x6	1x1	1x1	SAME SAME_TENSORFLOW VALID
3x8, 5x8, 7x8, 9x8	1x1	1x1	SAME SAME_TENSORFLOW VALID
1xW	1x1	1x1	SAME SAME_TENSORFLOW VALID
MxN	MxN	1x1	SAME SAME_TENSORFLOW VALID
MxN, where M,N in	AxB, where A,B in	CxD, where C,D in	SAME SAME_TENSORFLOW VALID
Any other	Any other	Any other	SAME SAME_TENSORFLOW VALID

Проблемы, квантизация

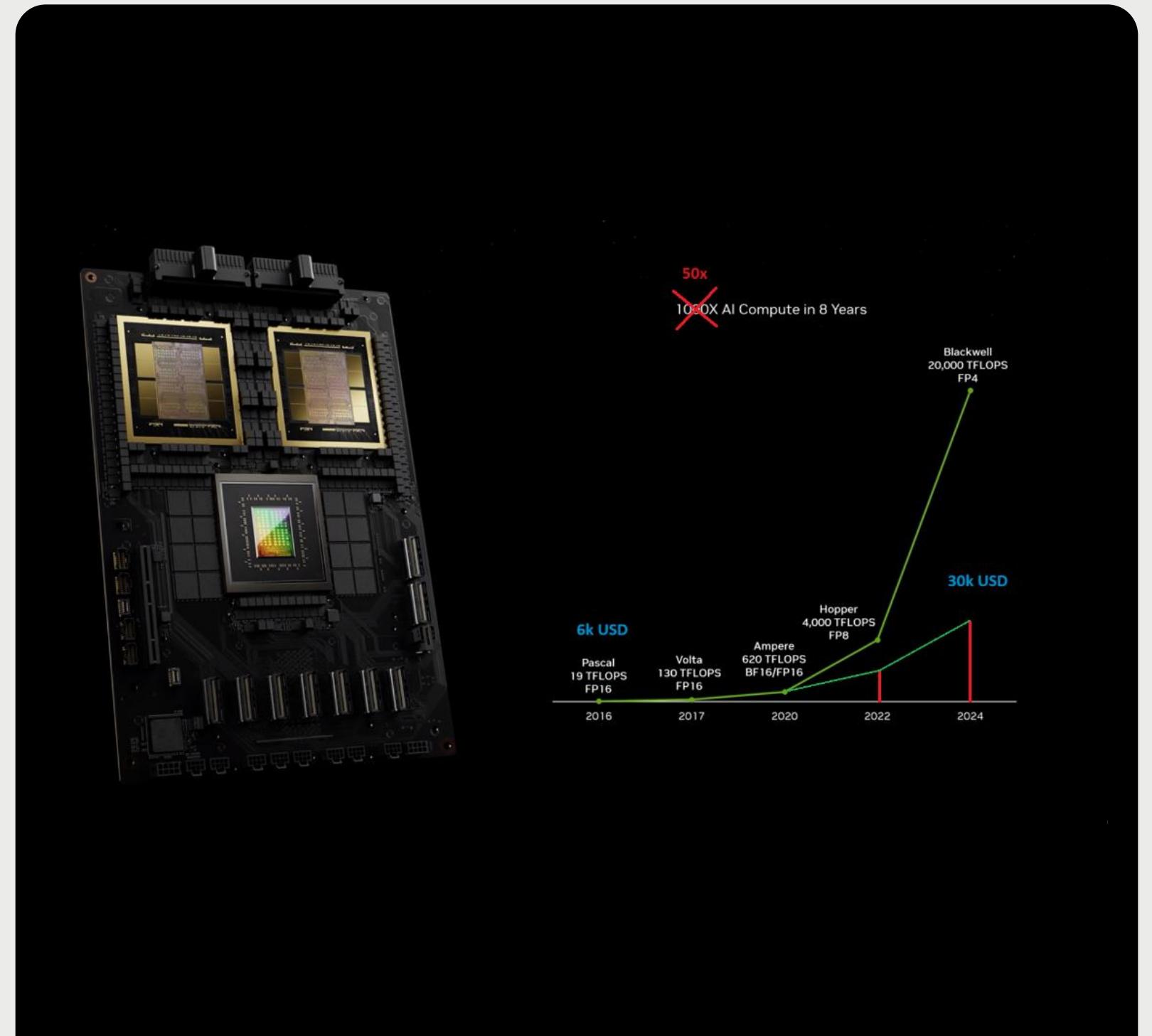
Ограниченный объём инструментов квантизации

	Hailo	Intel	TensorRT
Калибрация базовая	✓	✓	✓
Ассиметричная	✓	✓	✗
Дистилляция оригинальной модели	✓	✓	✗
Bias correction	✓	✓	✗
AdaRound	✓	✗	✗
Клиппинг весов и активаций	✓	✗	✗
Квантизация части слоев	✗	✓	✗
INT4	✓	✓	✓

Часть сетей может не квантоваться

Quantization aware training — не всегда возможен

Бенчмарки



И не только бенчмарки

- 1 Считать скорость
FP32 → FP16 → INT8 → INT4 → Binary
- 2 Выбирать в качестве референсной
модели лучшую
- 3 Не считать NMS и прочие части
модели
- 4 Считать с большим Batch → большой
Latency
- 5 Считать на разогнанном чипе
- 6 Просто врать:)



EVERYBODY
LIES

ThatGuyAaron

Какие есть оценки?

	NCNN - RPi5	CORAL	NCNN - RPi3	KHADAS vim3	Rock Pi 3A 3568	Jetson Nano	Hailo8	RK3588 - RKNN	RK3588 - NCNN	MAIX-III AX620A
MobileNet v2 224*224	14ms	3ms(int8)		5ms(int8)	10(int8)	20ms	3.5			
Yolov5S										
640*640/NoNMS	165-210ms	200ms(int8)	3000-3300ms'	100ms(int8)	170(int8)	120(slow nms)	20	12ms (fp16)	85ms	22ms
224*224/NoNMS	21ms	47ms(int8)	490-550ms'	30ms(int8)	18(int8) 50(fp16)	30	5			
ResNet50	67-89ms	51.5ms(int8)		27ms(int8)	44(int8)	29	28			

Model	Jetson Orin Nano 4GB	Jetson Orin Nano 8GB	Jetson Orin NX 8GB	Jetson Orin NX 16GB	Jetson AGX Orin 32GB	Jetson AGX Orin 64GB
Resnet50	1.6 ms	1 ms	0.57 ms	0.45 ms	0.269 ms	0.20 ms
SSD-Mobil net	1 ms	0.5ms	0.34 ms	0.28 ms	0.155 ms	0.13 ms
Yolov5s	6.32 ms	3.32 ms	2.63 ms	2.22 ms	1.27 ms	1 ms

Почему медленно: Как устроена память?

→ Общая память

Rockchip, intel

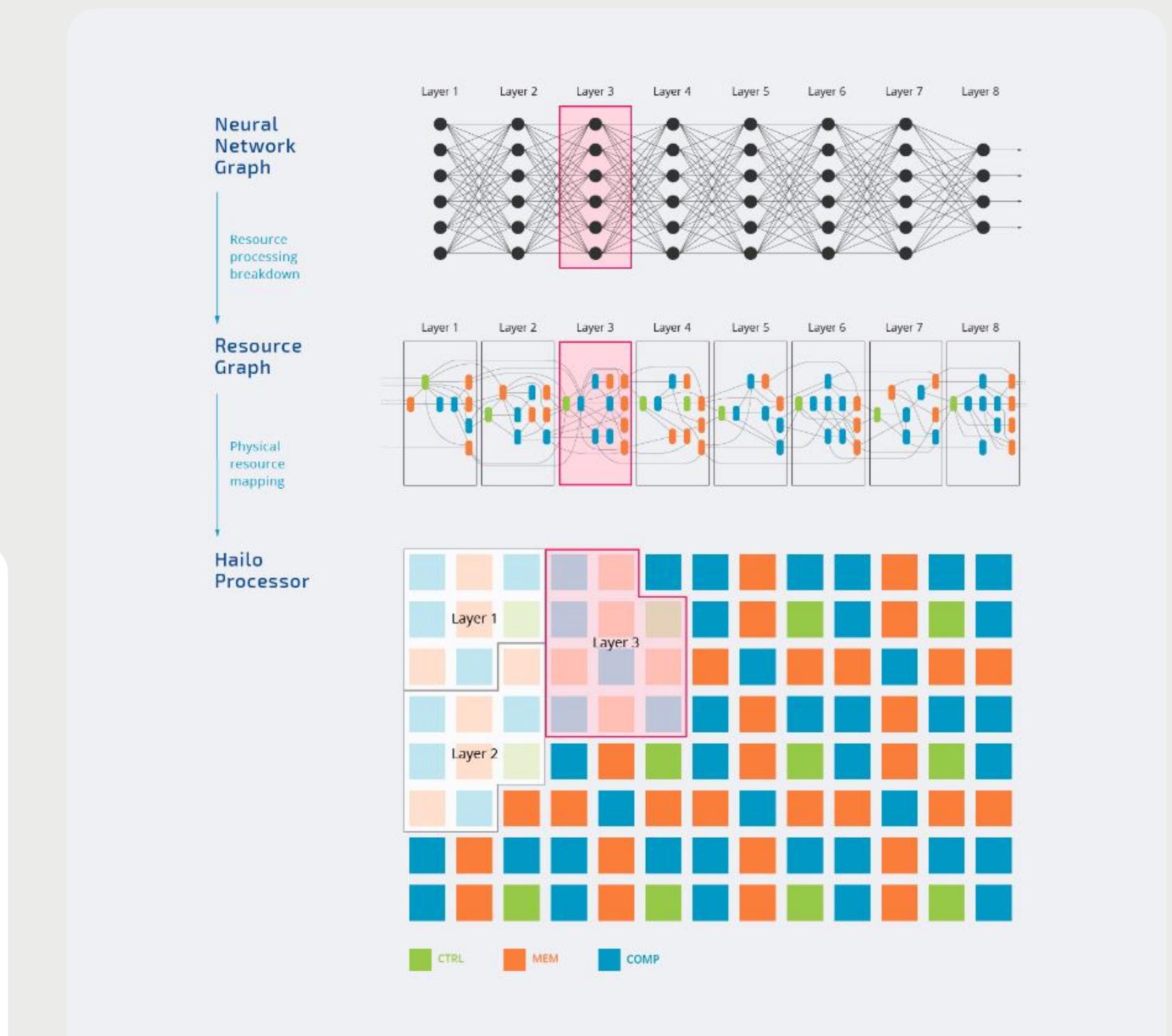
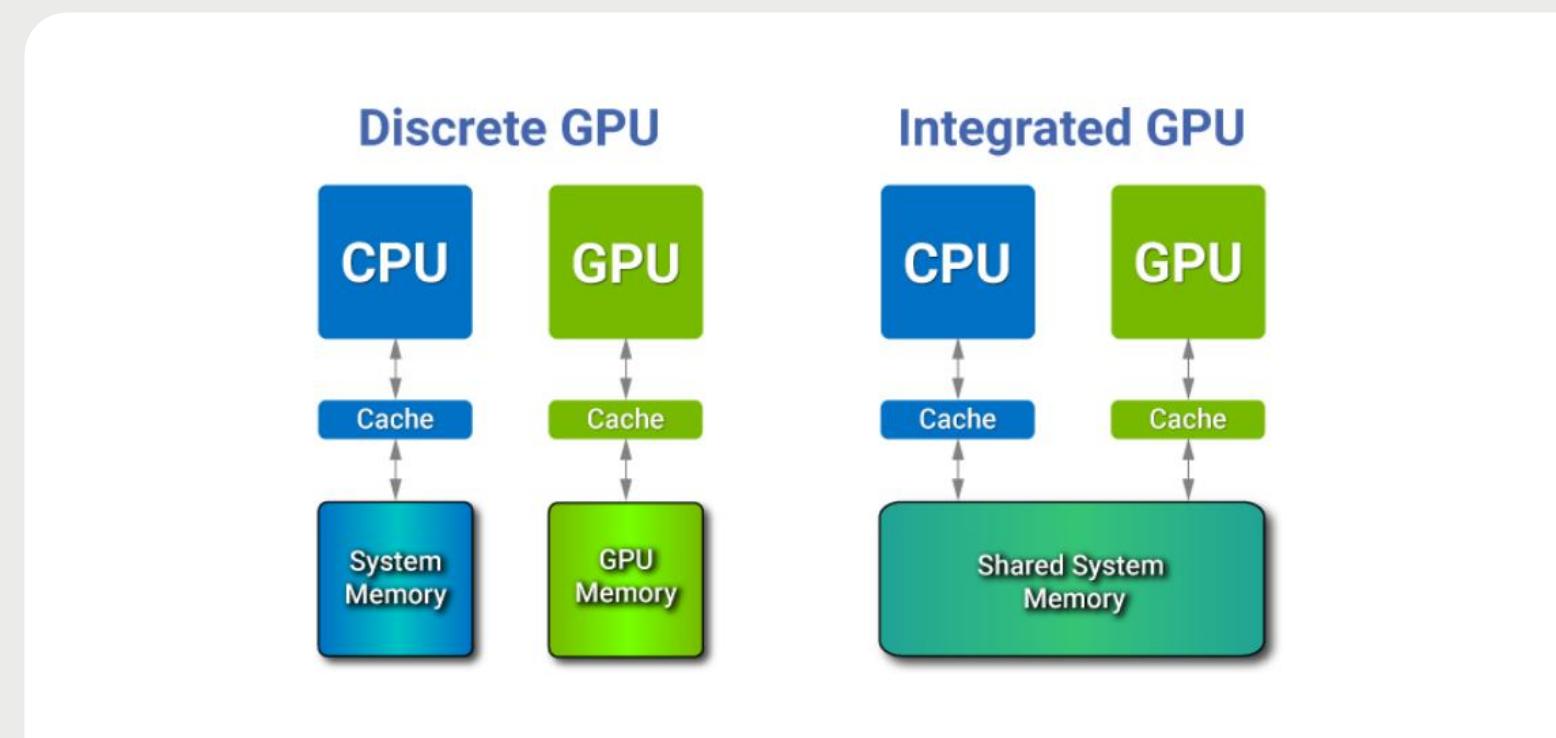
→ Общая память но заранее разделенная

→ M2 шина

Hailo, axelera

→ USB шина

Stick



Почему медленно: CPU

1 Получение картинок

USB шина / TCP / MIPI / etc

3 Препроцессинг картинок

5 Прочие процессы, мешают утилизации NPU

RockChip пример ниже

2 Декодинг картинок

Gstreamer?

4 Менеджмент шины

PCIe

模型\线程数	1	2	3	4	5	6
yolov5s	27.4491	49.0747	65.3673	63.3204	71.8407	72.0590

Почему медленно: прочее

Fallback невозможных слоёв
на GPU/CPU



C++ / Python

Обучение

→ Python

Экспорт

→ Python

Инференс?

→ Python — 70% платформ
C++ — 99% платформ



Обучение на NPU

-
- 1 Google TPU
 - 2 META MTIA v2
 - 3 Tesla D1 (Dojo)
 - 4 Amazon's Trainium2
 - 5 Microsoft Athena AI
 - 6 Apple M1, etc.
 - 7 etc..
-

Спасибо за внимание



Телеграмм канал где я рассказываю
про Computer Vision и его работу
на Edge платформах

[t.me/CVML_team ↗](https://t.me/CVML_team)

