Perceptual-Centric Image Super-Resolution using Heterogeneous Processors on Mobile Devices

MobiCom 2024

Kai Huang
University of Pittsburgh
USA

Xiangyu Yin
University of Pittsburgh
USA

Tao Gu
Macquarie University
Australia

Wei Gao
University of Pittsburgh
USA

汇报人：刘天恩

2024年12月27日

Intelligent Systems Lab @ PITT

Leader： Wei Gao

University of Pittsburgh, USA

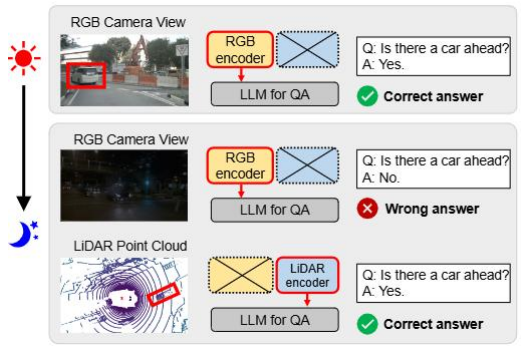
https://pittisl.github.io/

Projects:

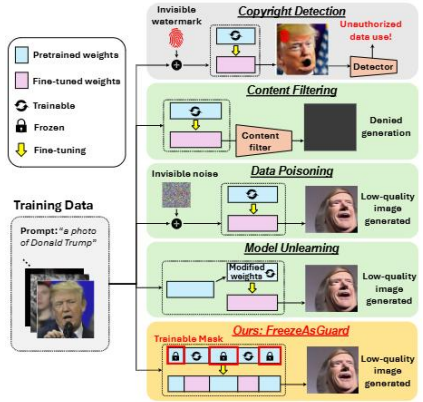
- Inference-time Computation in Generative AI
- On-device AI
- Mobile and Edge Computing Systems
- Intelligent Wireless Systems

Papers:

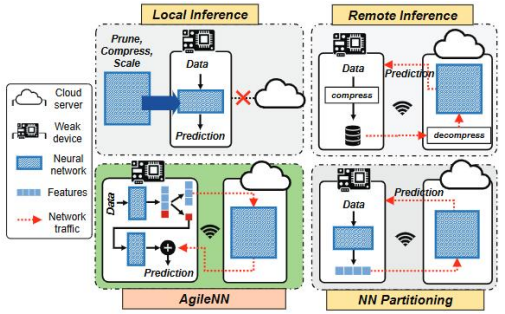
- [MobiCom'25] Modality Plug-and-Play: Runtime Modality Adaptation in LLM-Driven Autonomous Mobile Systems
- [MobiCom'25] FreezeAsGuard: Mitigating Illegal Adaptation of Diffusion Models via Selective Tensor Freezing
- [MobiCom'22] Real-time Neural Network Inference on Extremely Weak Devices: Agile Offloading with Explainable AI



MobiCom'25
mPnP-LLM



MobiCom'25
FreezeAsGuard



MobiCom'22
AgileNN

Outline

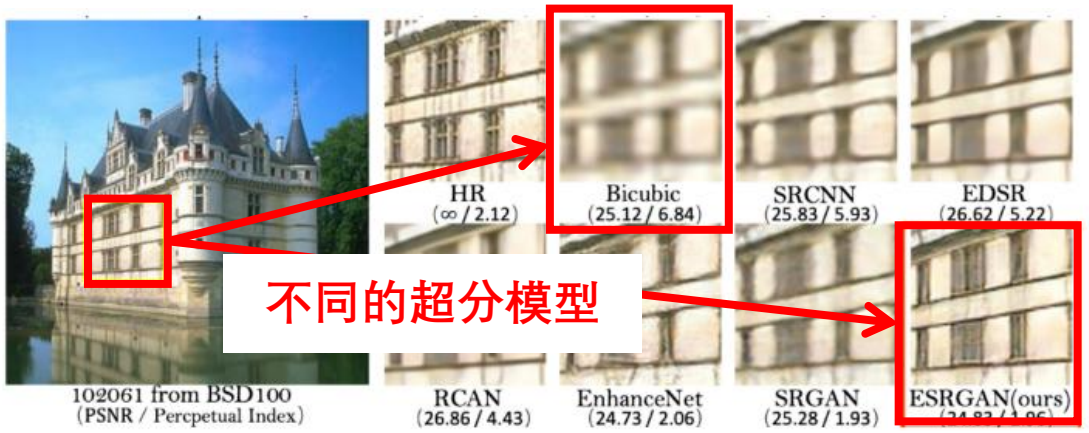
- Background
- Motivation
- Design
- Evaluation
- Conclusion

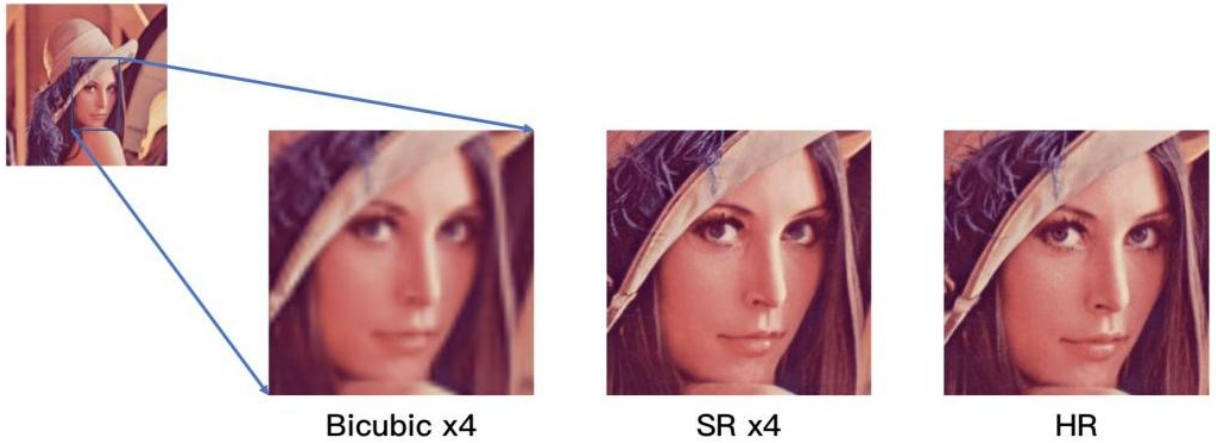
Outline

- **Background**
- Motivation
- Design
- Evaluation
- Conclusion

Background

- 图像超分辨率（Image super-resolution, SR）通过提高图像分辨率来提高图像质量





- 超分技术已被广泛应用于移动设备，以增强用户体验

手机游戏渲染

摄像头监控视频

观看高清视频

增强现实场景渲染

Background

Image super-resolution

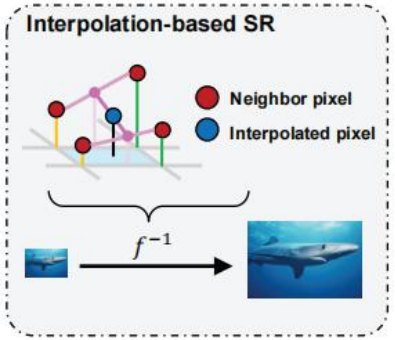
$$g : LR \rightarrow HR$$

低分辨率(LR)

基于插值的SR、基于过滤器的SR、基于NN的SR

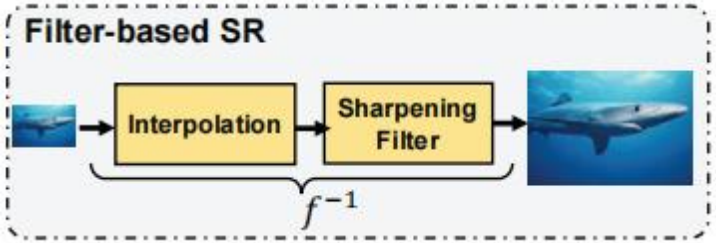
高分辨率(HR)

基于插值的SR



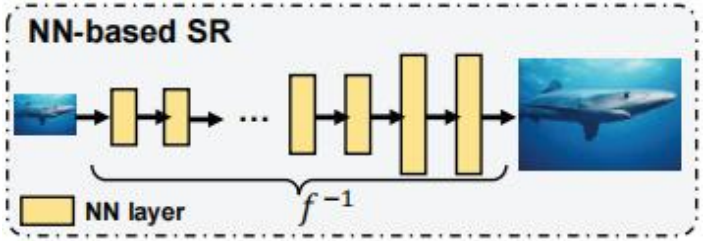
- 假设 g **完全线性**，通过简单的算术（例如，加权平均）在现有附近像素上创建新像素
- 缺乏高频图像信息（例如，物

基于过滤器的SR



- 基于滤波器的SR在 g **中加入了适度的非线性**，如锐化滤波器
- HR图像质量仍然有限。

基于NN的SR



- 通过回归任务学习 g **的通用非线性近似**，如生成式对抗网络
- 虽然提高了HR质量，但是计算成本高昂

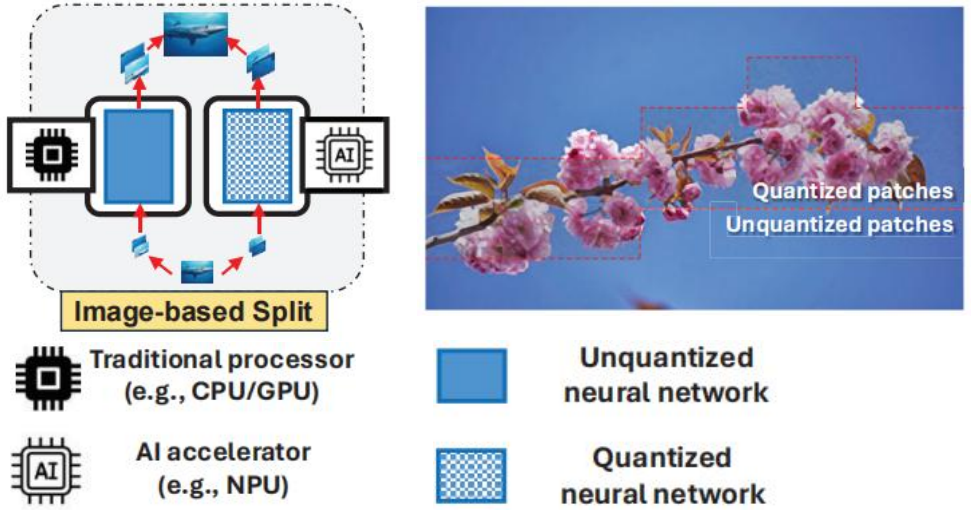
Problem: 如何实现超分模型在移动端的高效部署?

Related Work

- **设计轻量化超分辨率模型：**
  - 模型压缩
    - 卷积核压缩
  - 模型量化
    - 权重量化、激活量化

Limitation1: 压缩SR模型可以提高速度，但会严重损害图像质量。

- **基于异构处理器的高效神经网络计算：**
  - 神经网络切分
  - 混合精度计算
  - 神经网络搜索



Limitation2: 模型切分时忽略了AI加速器运算时可能造成的精度损失。

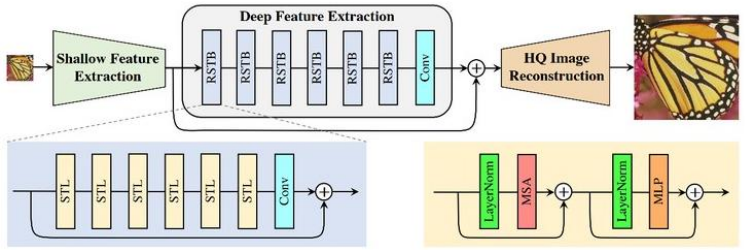
Outline

- Background
- **Motivation**
- Design
- Evaluation
- Conclusion

Motivation

- 超分辨率模型在相同参数大小的情况下需要更多的计算量

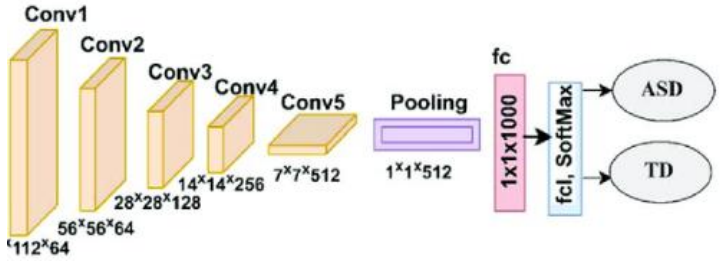
超分模型：SwinIR，1100万个参数



层深度

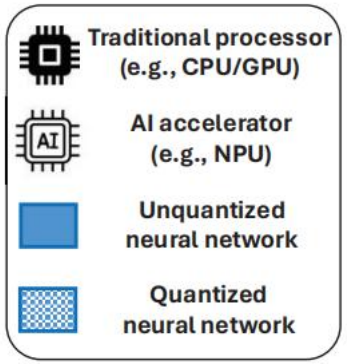
中间数据的计算量
（202 GFLOPs）

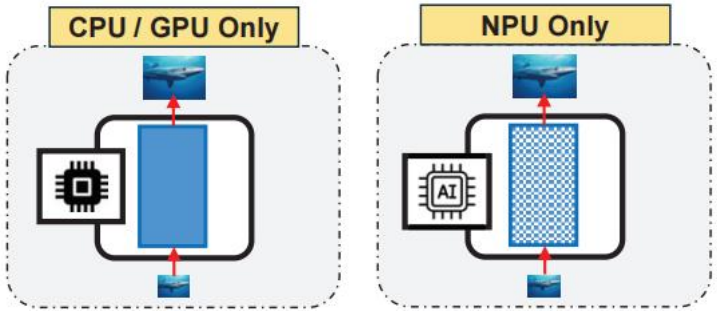
分类模型：ResNet18，1100万个参数



层深度

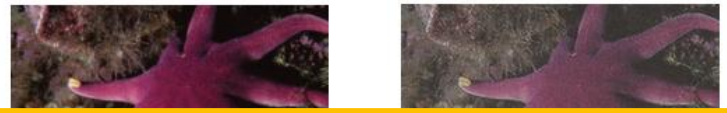
中间数据的计算量
（2 GFLOPs）





支持浮点精度　　支持整数精度

SR Model	CPU	GPU	Tensor NPU
SRCNN [11]	0.075s	0.022s	**0.006s**
EDSR [40]	1.91s	0.21s	**0.034s**
ESRGAN [73]	3.24s	0.46s	**0.074s**



NPU:

***v.s.* GPU = 6.3x**

***v.s.* CPU = 55.8x**

图像质量

Opportunity1: 相比于CPU/GPU，SR模型在NPU上更快，但质量更低

Motivation

- **超分辨率模型产生的超分图像质量可以通过不同的指标进行评估，不同的指标评估了不同方面的图像质量**

- **Structural metrics:**
  - PSNR(↑better): 比较失真图像与原始图像相似程度
  - SSIM(↑better): 考虑图像的亮度、对比度和结构信息
- **Perceptual metrics:**
  - LPIPS(↓better): 比较高级特征差异，如纹理、形状
  - PIQE(↓better): 比较图像的块状结构和噪声特征

基于图像的结构信息来衡量图像之间的相似度
(较多研究)

人眼对图像质量的感知
(较少研究)
更能反应图片的质量

(a) Original Image (b) Partially Cropped (c) Random Noise

相同的图像失真可能导致PIQE和PSNR质量评分之间的巨大差异

Opportunity2：需要一个Perceptual-Centric 的综合指标来评估质量

Motivation

- SR模型中不同层对量化精度的敏感性不同

(a) **Original Image** PIQE: 18.03 (excellent)

(b) **After SR** PIQE: 36.73 (fair)

(c) **Quantized SR** PIQE: 53.35 (poor)

(a) First 3 blocks quantized PIQE: 40.02

(b) Middle 3 blocks quantized PIQE: 70.57

(c) Last 3 blocks quantized PIQE: 13.77

PIQE(↓better)

图像质量

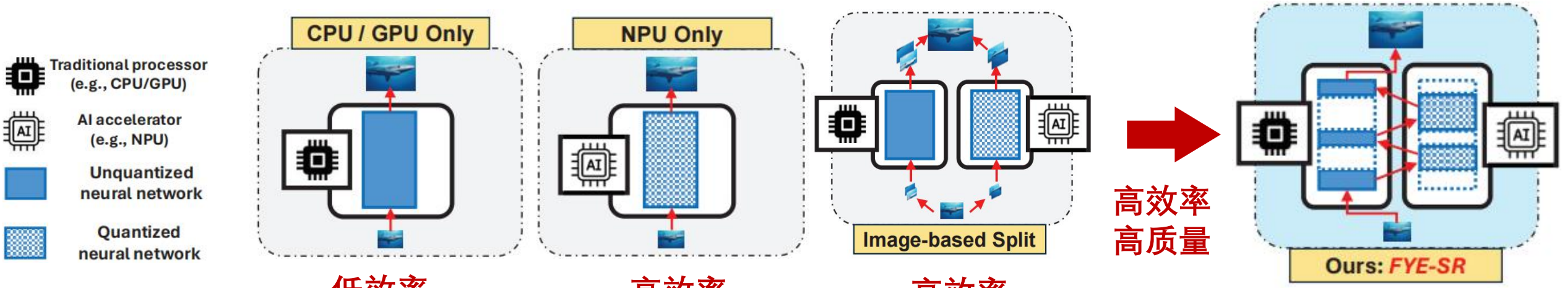
量化后三个blocks是图片质量损失最少的

Opportunity3：对不同的层采用不同的量化精度，tradeoff效率和质量

Outline

- Background
- Motivation
- **Design**
- Evaluation
- Conclusion

Design objectives



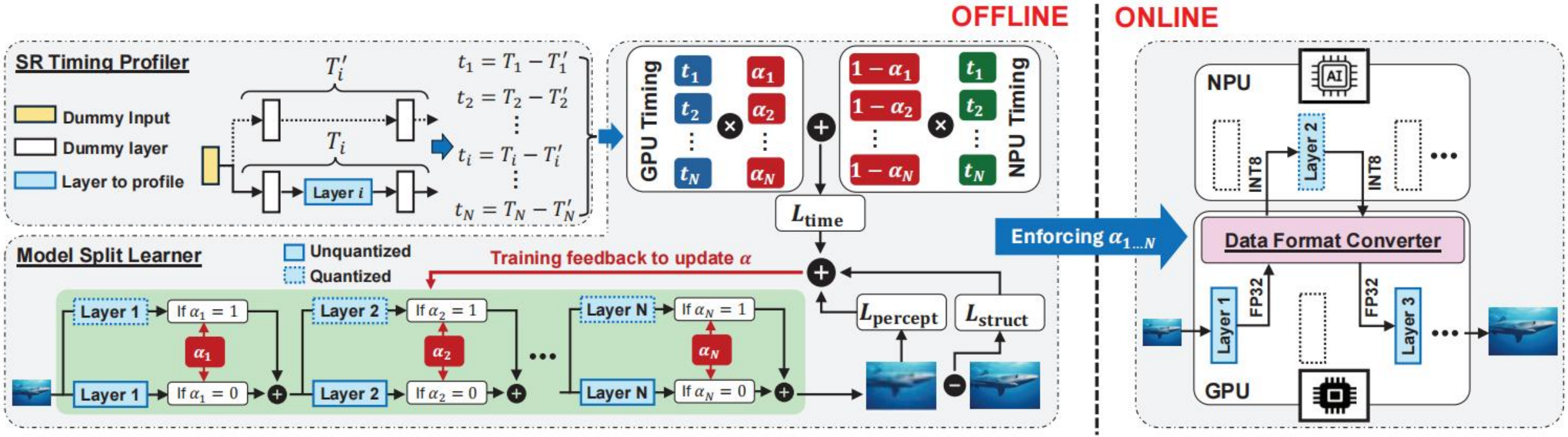
设计目标： 同时最小化超分模型的端到端延迟(计算+切换) $T(\mathcal{P})$，并最大化超分图像的感知质量 $Q(\mathcal{P})$

$\mathcal{P}$ 表示传统处理器和AI加速器之间的SR计算分割

$$\min_{\mathcal{P}} (T(\mathcal{P}), -Q(\mathcal{P})) \longrightarrow \min_{\mathcal{P}} (-Q(\mathcal{P})) \quad \text{s.t. } T(\mathcal{P}) \leq T_{obj}.$$

用 ϵ - 约束
方法简化

System overview



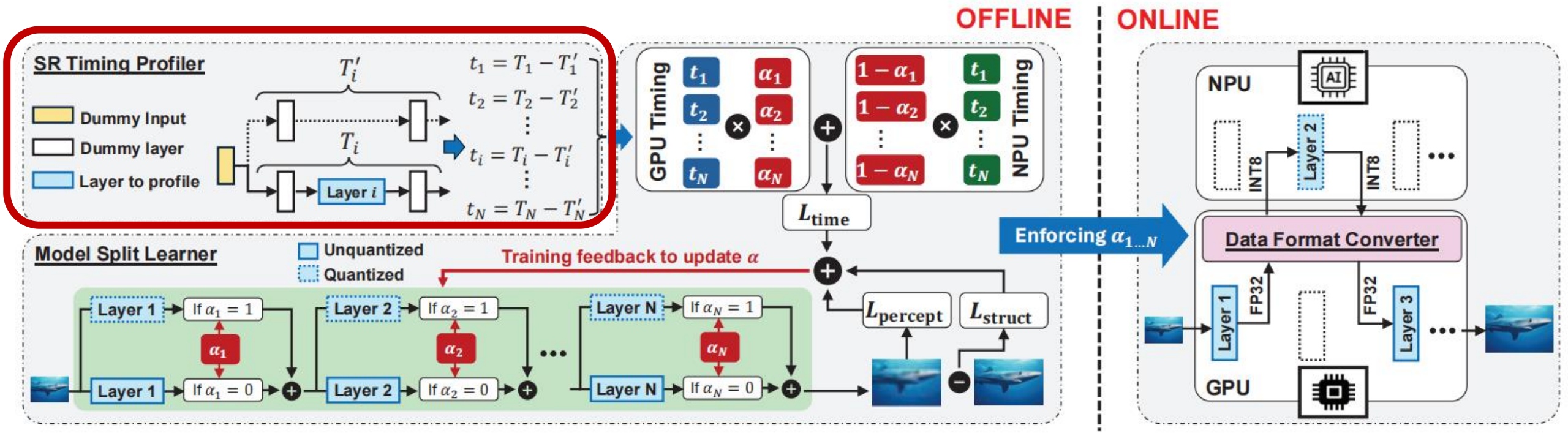
两个阶段：

- **Offline:** SR模型在异构处理器上的切分方案
- **Online:** 在异构处理器上执行SR模型的切分方案

三个模块：

- **SR Timing Profiler：** 测量SR模型的不同NN层在传统处理器(例如GPU)和AI加速器(例如NPU)上的端到端延迟；
- **Model Split Learner：** 解决最优SR模型分割优化问题；
- **Data Format Converter：** 将中间特征图转换为正确的数据格式(例如INT8和FP32)，以便在异构处理器之间正确切换SR模型的计算。

System overview



两个阶段：

- **Offline:** SR模型在异构处理器上的切分方案
- **Online:** 在异构处理器上执行SR模型的切分方案

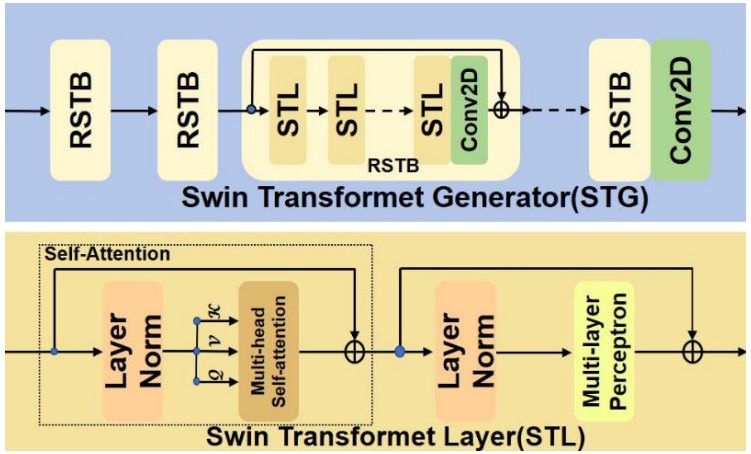
三个模块：

- **SR Timing Profiler：** 测量SR模型的不同NN层在传统处理器(例如GPU)和AI加速器(例如NPU)上的端到端延迟；
- **Model Split Learner：** 解决最优SR模型分割优化问题；
- **Data Format Converter：** 将中间特征图转换为正确的数据格式(例如INT8和FP32)，以便在异构处理器之间正确切换SR模型的计算。

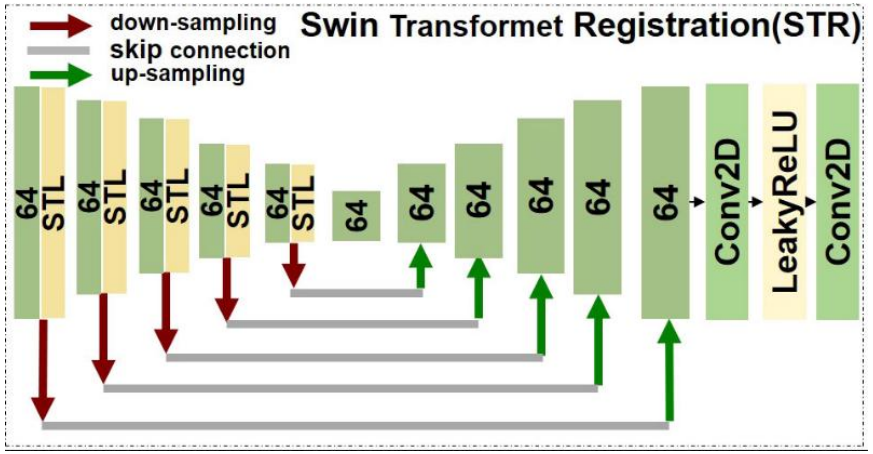
Challenge1

- 直接测量一个神经网络层的时间开销比较困难

难以区分SR模型每一层的计算时延

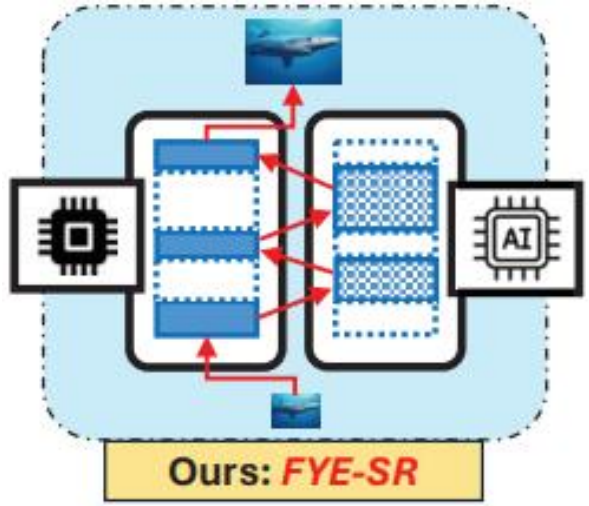


- 层之间连续耦合结构



- 输入和输出数据处理的开销(维度的转换)难以直接排除

切换时延难以计算



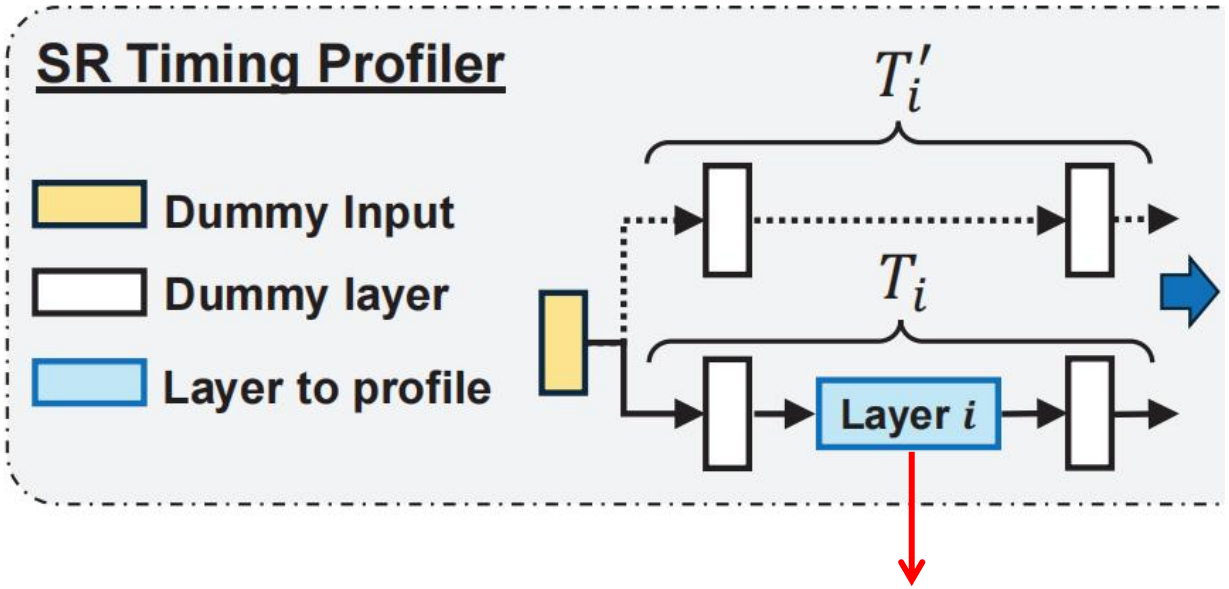
- 需要检索OS kernel的多个硬件访问事件的时间戳

Challenge1: SR模型每一层的端到端时延难以计算

Design1—SR Timing Profiler

- 计算目标层的计算延迟

虚拟层：
人为添加的
多个卷积层



虚拟层的
计算延迟

目标层的
计算延迟

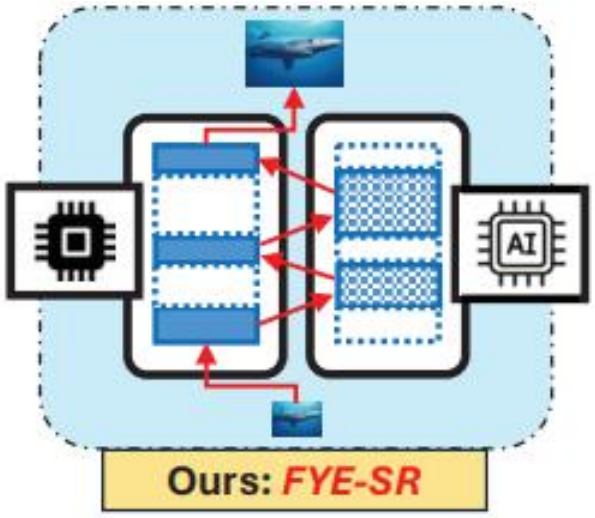
$$t_1 = T_1 - T_1'$$
$$t_2 = T_2 - T_2'$$
$$\vdots$$
$$t_i = T_i - T_i'$$
$$\vdots$$
$$t_N = T_N - T_N'$$

间接计算目标层Layer i的计算延迟：

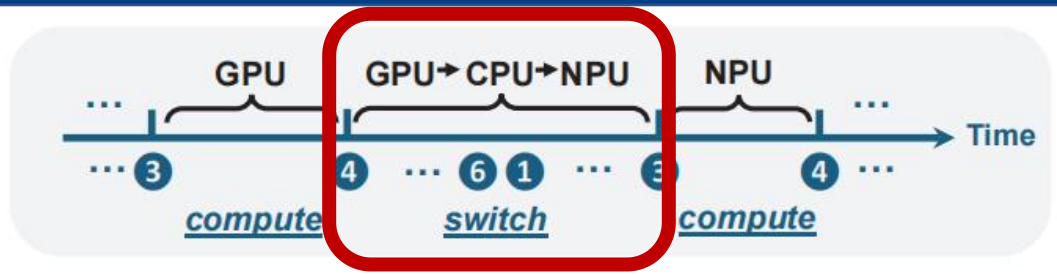
1. 在目标层前后添加虚拟层（多个卷积层）
2. 在虚拟层之间插入想要测量的目标层
3. 测量包含虚拟层的整个网络的运行时间
4. 减去虚拟层的运行时间

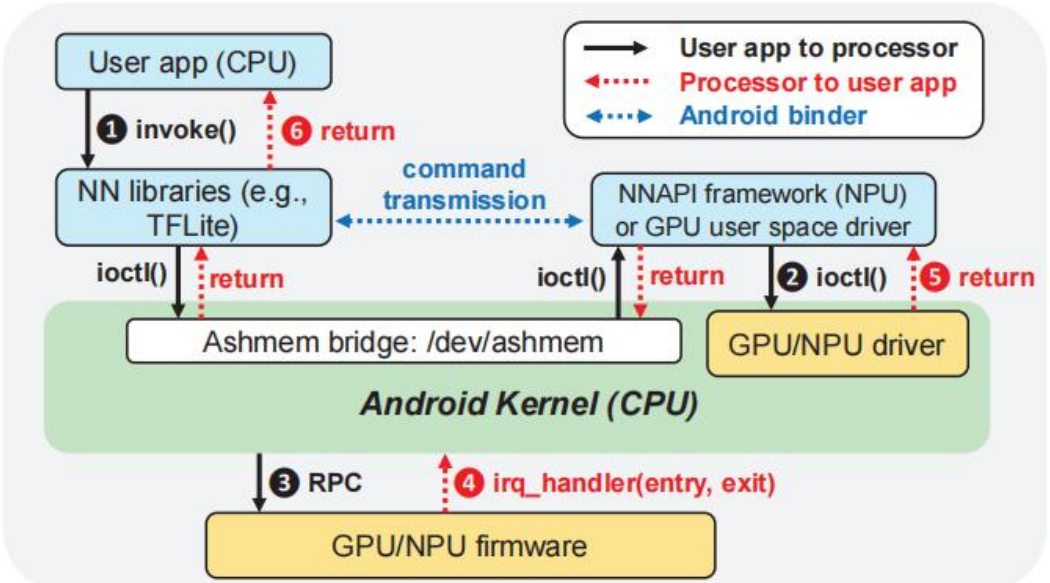
Design1—SR Timing Profiler

- 计算目标层的切换延迟



SR模型分割时考虑不同层在异构处理器上切换时延

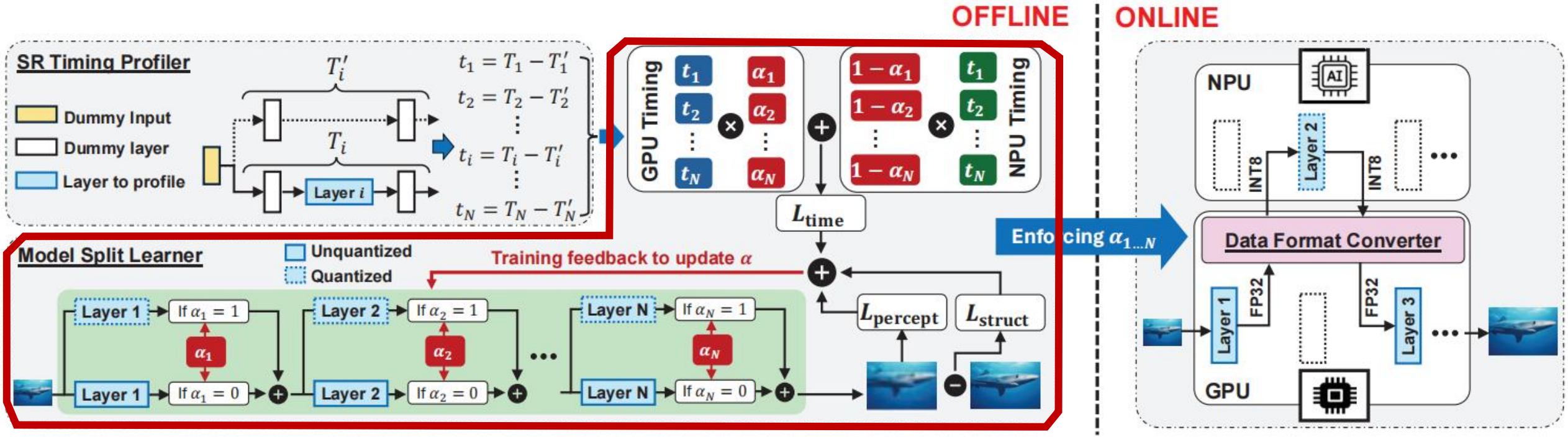




我们通过**检索Android OS kernel中的硬件访问事件的时间戳**来测量在不同处理器之间切换SR计算所需的时间

- Android内核的远程过程调用(RPCs)
- 处理器(例如GPU或NPU)发送的硬件中断(irq_handler)

System overview



两个阶段：

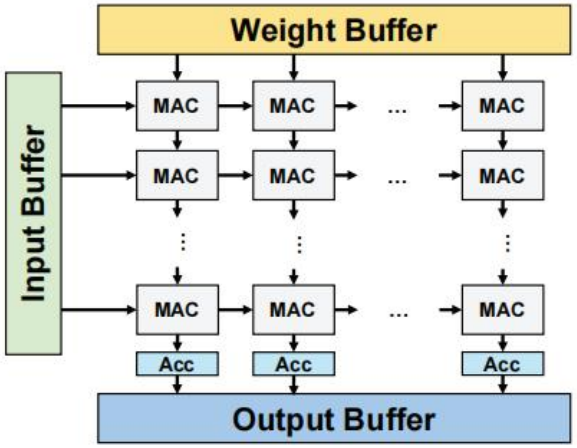
- **Offline:** SR模型在异构处理器上的切分方案
- **Online:** 在异构处理器上执行SR模型的切分方案

三个模块：

- **SR Timing Profiler：** 测量SR模型的不同NN层在传统处理器(例如GPU)和AI加速器(例如NPU)上的端到端延迟；
- **Model Split Learner：** 解决最优SR模型分割优化问题；
- **Data Format Converter：** 将中间特征图转换为正确的数据格式(例如INT8和FP32)，以便在异构处理器之间正确切换SR模型的计算。

Challenge2

- 大多数AI加速器只支持整数计算（例如，INT8）以节省芯片空间



AI加速器(NPU)的微架构
(MAC：乘法&加法，Acc：累加器)

支持INT8 —> 支持FP32:
25x 空间 & 3x 晶体管数量

高通骁龙8 Gen 3提供了对FP16的支持，但性能很低

Device Model	GPU	NPU (FP16)	NPU (INT8)
Google Pixel 6	0.022s	-	0.006s
Galaxy S24 Ultra	0.019s	0.043s	0.005s

时延

高于　高于

(a) **Original Image** PIQE: 18.03 (excellent)
(b) **After SR** PIQE: 36.73 (fair)
(c) **Quantized SR** PIQE: 53.35 (poor)

图像质量

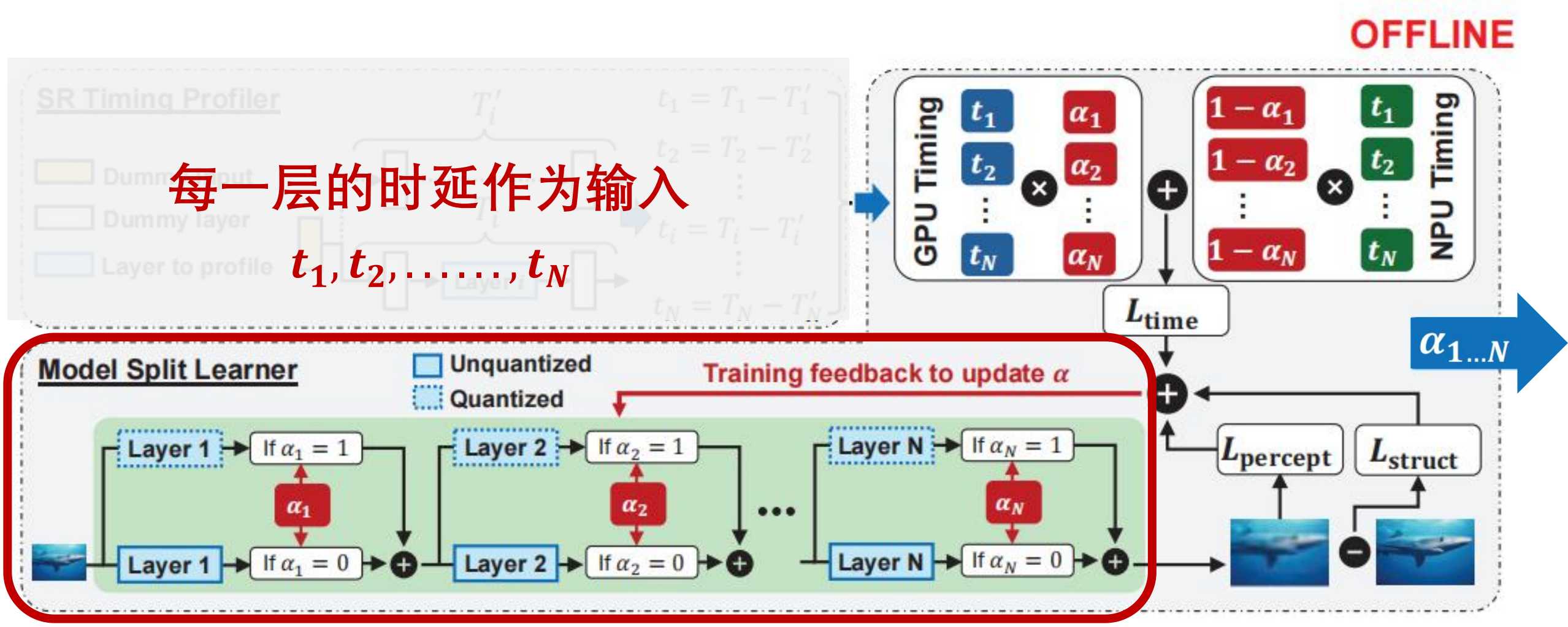
Challenge2: 如何在保证质量&效率下，实现SR模型的切分？

Design2—Model Split Learner

- 计算时延和感知质量的联合训练确定最优分割 $\min_{\mathcal{P}} (-Q(\mathcal{P}))$ s.t. $T(\mathcal{P}) \leq T_{obj}$.

每一层的时延作为输入

$t_1, t_2, \ldots\ldots, t_N$

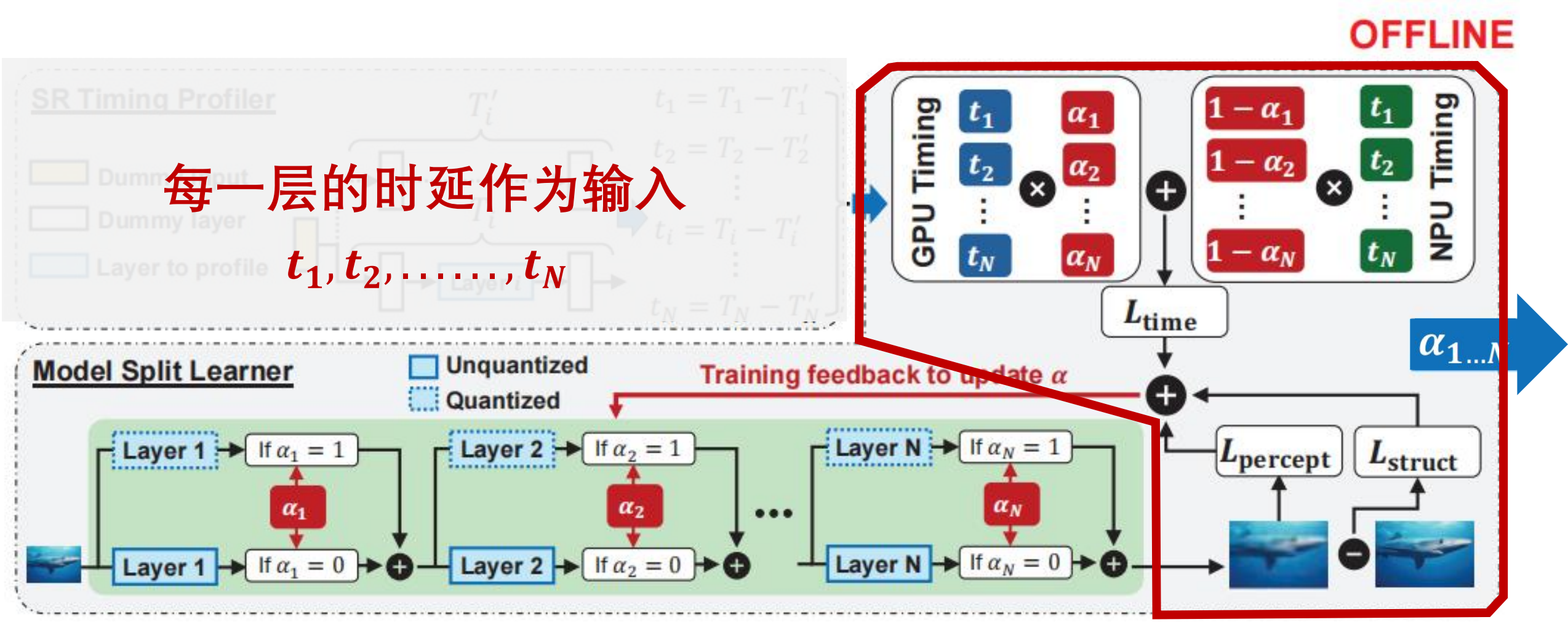


Design2—Model Split Learner

- 计算时延和感知质量的联合训练确定最优分割 $\min_{\mathcal{P}} (-Q(\mathcal{P}))$ s.t. $T(\mathcal{P}) \leq T_{obj}$.

每一层的时延作为输入

$t_1, t_2, \ldots\ldots, t_N$



Design2—Model Split Learner

- 计算时延和感知质量的联合训练确定最优分割

$$\min_{\mathcal{P}} \left(-Q(\mathcal{P})\right) \quad \text{s.t. } T(\mathcal{P}) \leq T_{obj}.$$

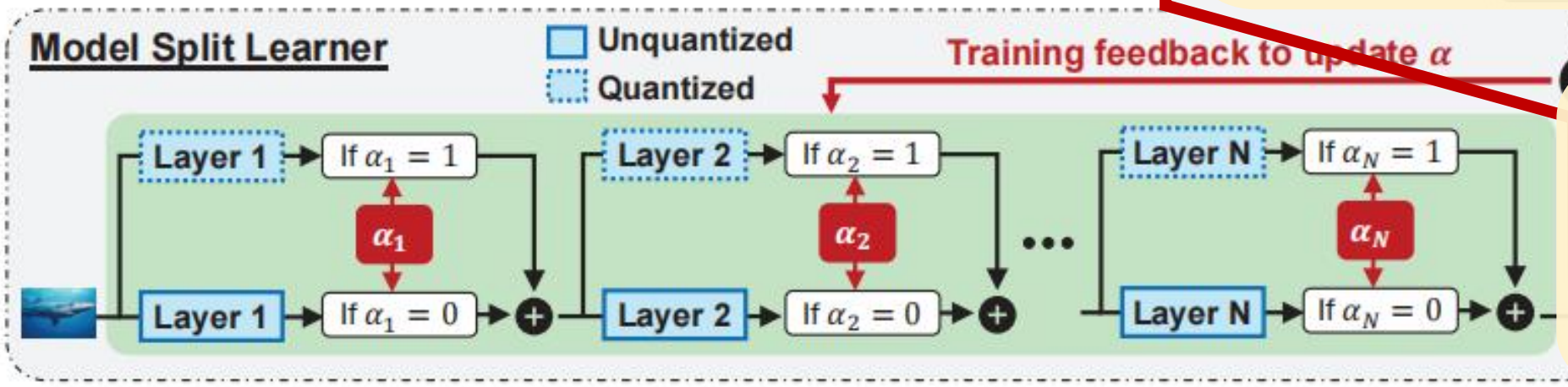
每一层的时延作为输入

$$T_{comp} = \sum_{i=1}^{M} \left(\alpha_i t_i^N + (1-\alpha_i) t_i^G \right),$$

$$T_{switch} = \sum_{i=1}^{M-1} \left((1-\alpha_i)\alpha_{i+1} t_i^{G\to N} + \alpha_i (1-\alpha_{i+1}) t_i^{N\to G} \right),$$

$$\begin{aligned} T(\boldsymbol{\alpha}) = {} & \boldsymbol{\alpha} \odot \mathbf{t}^N + (\mathbf{1}-\boldsymbol{\alpha}) \odot \mathbf{t}^G \\ & + [(\mathbf{1}-\boldsymbol{\alpha}_{1:M-1})\boldsymbol{\alpha}_{2:M}] \odot \mathbf{t}^{G\to N} \\ & + [\boldsymbol{\alpha}_{1:M-1}(\mathbf{1}-\boldsymbol{\alpha}_{2:M})] \odot \mathbf{t}^{N\to G} \end{aligned}$$

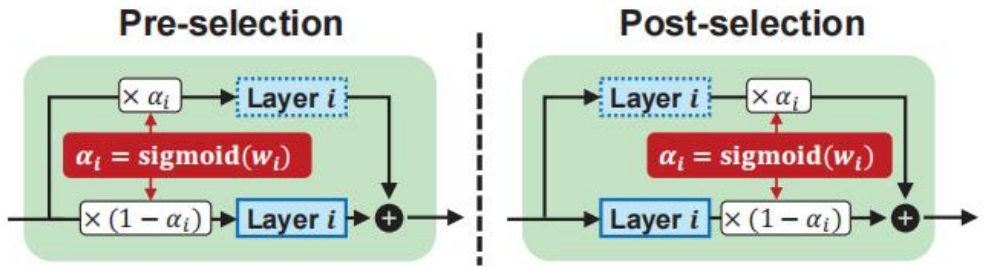
$\alpha_{1\ldots N}$



$$w \leftarrow w - \eta \cdot \left(\frac{\partial L_{time}}{\partial w} + \lambda_1 \frac{\partial L_{percept}}{\partial w} + \lambda_2 \frac{\partial L_{struct}}{\partial w} \right),$$

Design2—Model Split Learner

- 为了在训练中应用优化器（SGD /Adam ），采用α的连续表示



$$w \leftarrow w - \eta \cdot \left(\frac{\partial L_{time}}{\partial w} + \lambda_1 \frac{\partial L_{percept}}{\partial w} + \lambda_2 \frac{\partial L_{struct}}{\partial w}\right),$$

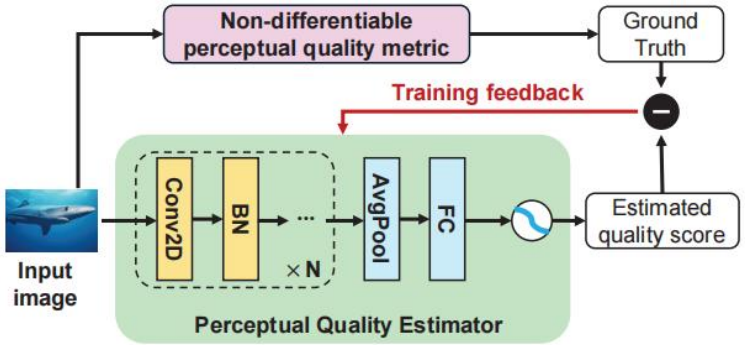
学习率

- 如何建模计算时间损失函数?

$$T_{comp} = \sum_{i=1}^{M} \left(\alpha_i t_i^N + (1-\alpha_i) t_i^G\right), \qquad T_{switch} = \sum_{i=1}^{M-1} \left((1-\alpha_i)\alpha_{i+1} t_i^{G\to N} + \alpha_i (1-\alpha_{i+1}) t_i^{N\to G}\right),$$

$$\begin{aligned} T(\boldsymbol{\alpha}) = {} & \boldsymbol{\alpha} \odot \mathbf{t}^N + (\mathbf{1}-\boldsymbol{\alpha}) \odot \mathbf{t}^G \\ & + [(\mathbf{1}-\boldsymbol{\alpha}_{1:M-1})\boldsymbol{\alpha}_{2:M}] \odot \mathbf{t}^{G\to N} \\ & + [\boldsymbol{\alpha}_{1:M-1}(\mathbf{1}-\boldsymbol{\alpha}_{2:M})] \odot \mathbf{t}^{N\to G} \end{aligned}$$

- 如何实现可差分的感知损失?

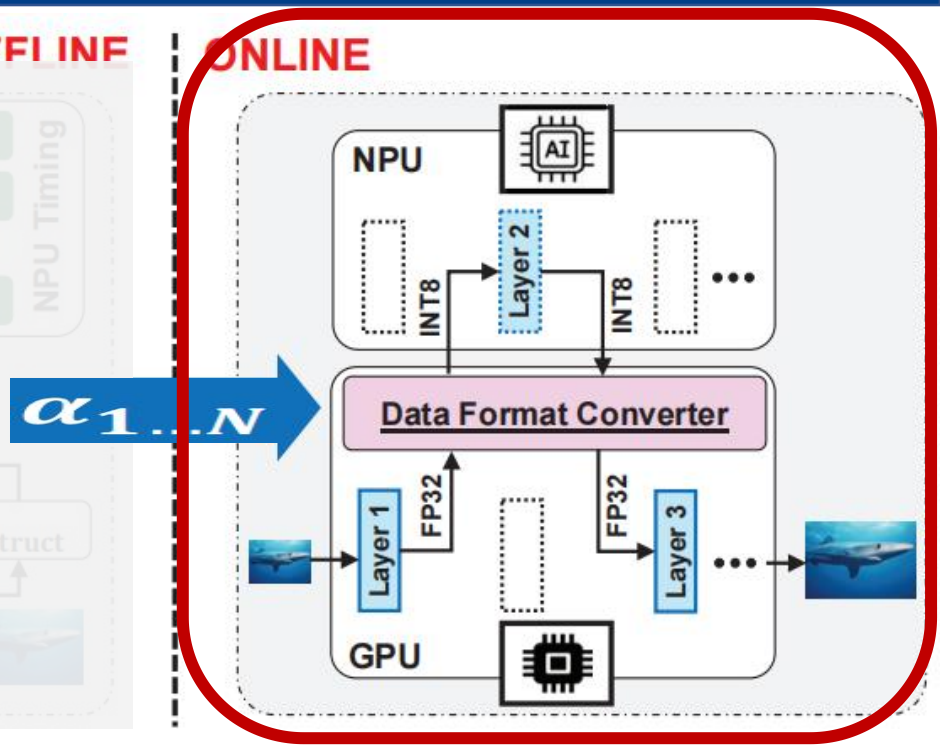


构建一个基于神经网络的感知质量估计器，模拟感知图像质量度量。

- 估计器由几个卷积层和BN层组成
- 从输入图像中提取全局和局部特征，并使用平均池化层+全连接层将提取的特征投影到一个标量中。
- 当训练这个估计器时，使用原始不可微度量给出的图像质量分数作为Ground Truth。

System overview

SR模型分割决策



两个阶段：

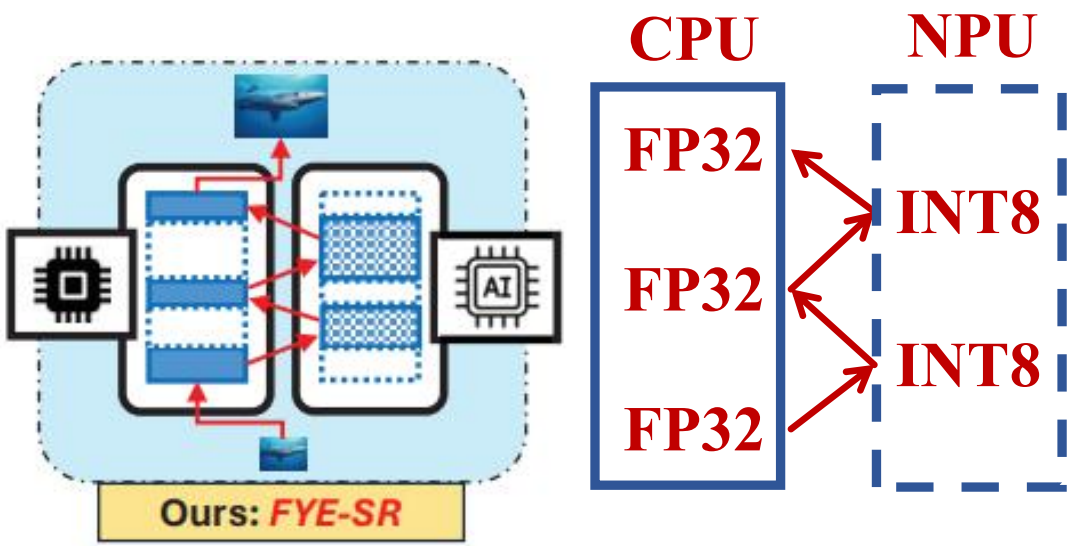
- **Offline:** SR模型在异构处理器上的切分方案
- **Online:** 在异构处理器上执行SR模型的切分方案

三个模块：

- **SR Timing Profiler：** 测量SR模型的不同NN层在传统处理器(例如GPU)和AI加速器(例如NPU)上的端到端延迟;
- **Model Split Learner：** 解决最优SR模型分割优化问题;
- **Data Format Converter：** 将中间特征图转换为正确的数据格式(例如INT8和FP32)，以便在异构处理器之间正确切换SR模型的计算。

Challenge3

- 异构处理器之间的数据格式转换计算开销大：FP32—>INT8—>FP32



不同层在异构处理器上数据格式转换的计算开销

数据格式转换的计算延迟(ms)

Method	640×360×64 quant./dequant.	640×360×32 quant./dequant.
Arithmetic	46.4/46.1	27.7/28.9
Table lookup	-/20.0	-/10.5

10~40ms

SR模型的计算延迟

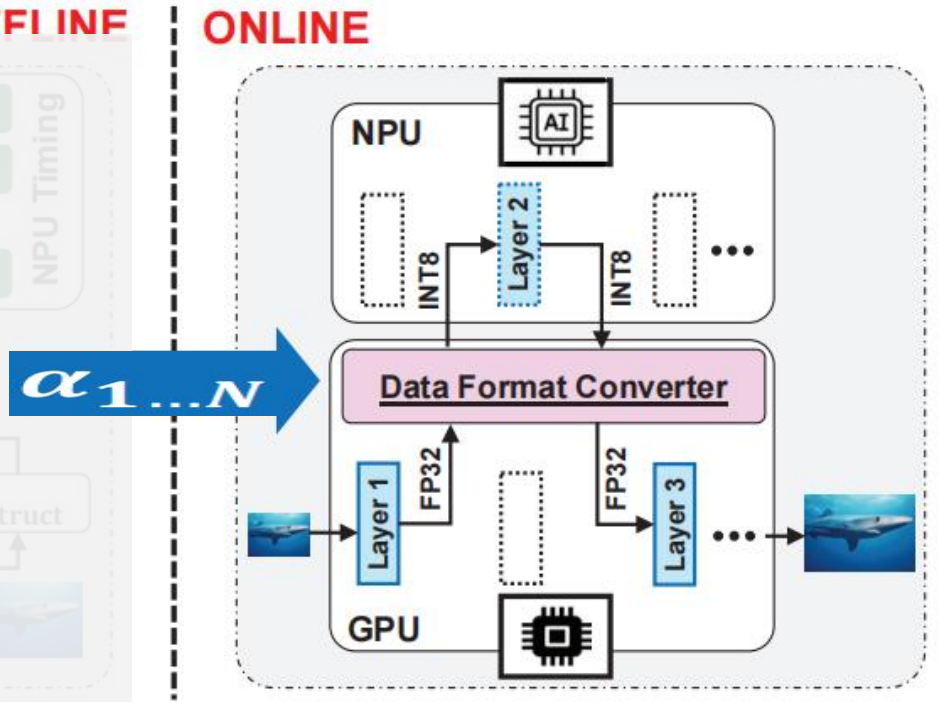
Device Model	GPU	NPU (FP16)	NPU (INT8)
Google Pixel 6	0.022s	-	0.006s
Galaxy S24 Ultra	0.019s	0.043s	0.005s

~5ms

Challenge3: 如何减少异构处理器之间数据格式转换的开销?

Design3——Data Format Converter

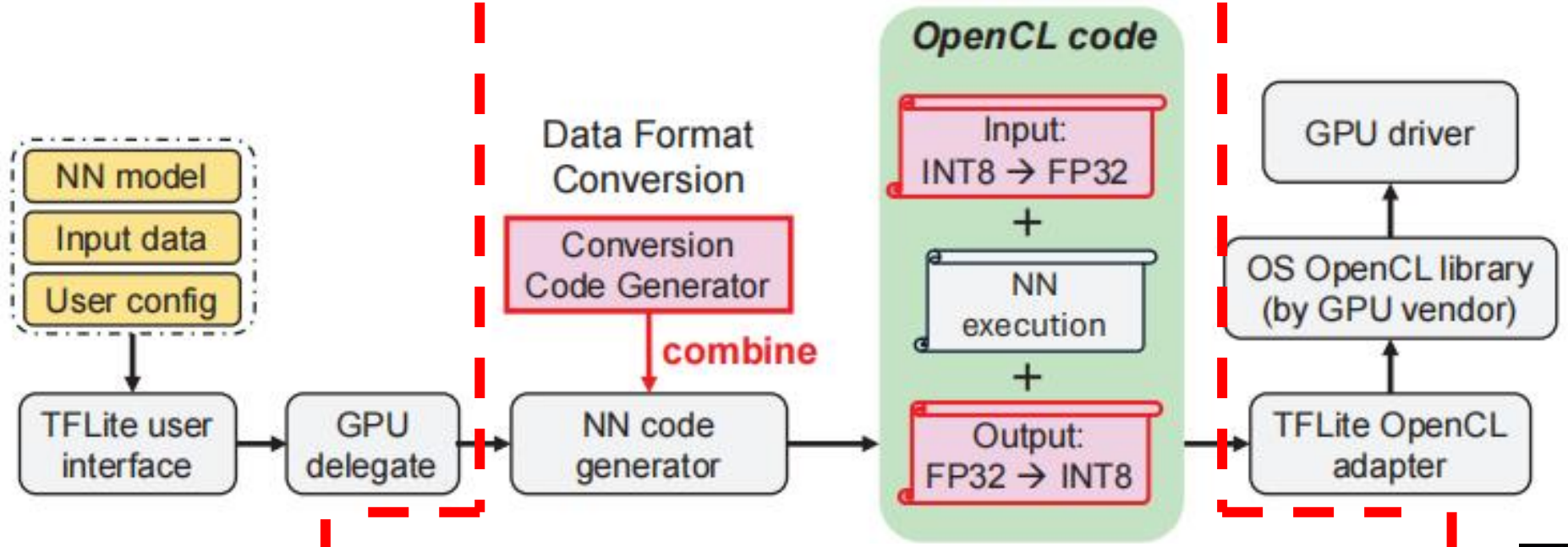
SR模型分割决策



为了降低开销:

- 在具有高并行性的移动GPU上进行这种转换。
- 当前的移动设备上的深度学习框架（例如TFLite）不支持这种转换。我们直接修改了处理移动NN执行的现成TFLite库，并在其上实现了我们的自定义GPU并行格式转换。

Design3——Data Format Converter




```
# convert a tf.Keras model to tflite model with INT8
converter = tf.lite.TFLiteConverter.from_keras_model(

# set optimization to DEFAULT and set float16 as the
# the target platform
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]
tflite_model = converter.convert()

# write the model to a tflite file as binary file
tflite_no_quant_file = TFLITE_MODEL_DIR + "mobilenet_
with open(tflite_no_quant_file, "wb") as f:
    f.write(tflite_model)
```

用户通过TFLite提供的接口，定义模型结构。

```
std::string c;
c += "__kernel void vector_f32toint8_gpu( __global const float* src_f,";
c += "                __global char* res, const int num ){";
c += "    const int idx = get_global_id(0);";
c += "    if (idx < num) {";
c += "        res[idx] = convert_char(src_f[idx]);";
c += "    }";
c += "}";
```

- 将Conversion Code添加进NN模型中；
- 将NN模型转化成更低级的中间表示(比如OpenCL)，适合在移动设备或嵌入式设备上执行。

在GPU上执行格式转换

Outline

- Background
- Motivation
- Design
- **Evaluation**
- Conclusion

Implementation

- **Offline Phase:**
  - SR Timing Profiler: A rooted smartphone with Android OS
    - .tflite models: TFLite C++ library
    - Test program: C++
  - Model Split Learner: Workstation
    - Optimization Toolkit (tfmot)
- **Online Phase：**
  - Android App
  - Modified TFLite library: C++ and OpenCL

三种基于NN的SR模型(模型复杂度小—>大)：

- EDSR [CVPR'17]
- ESRGAN [ECCV'18]
- ENet-PAT [ICCV'17]

数据集(对HR图像进行双边降采样，获得成对的LR图像)：

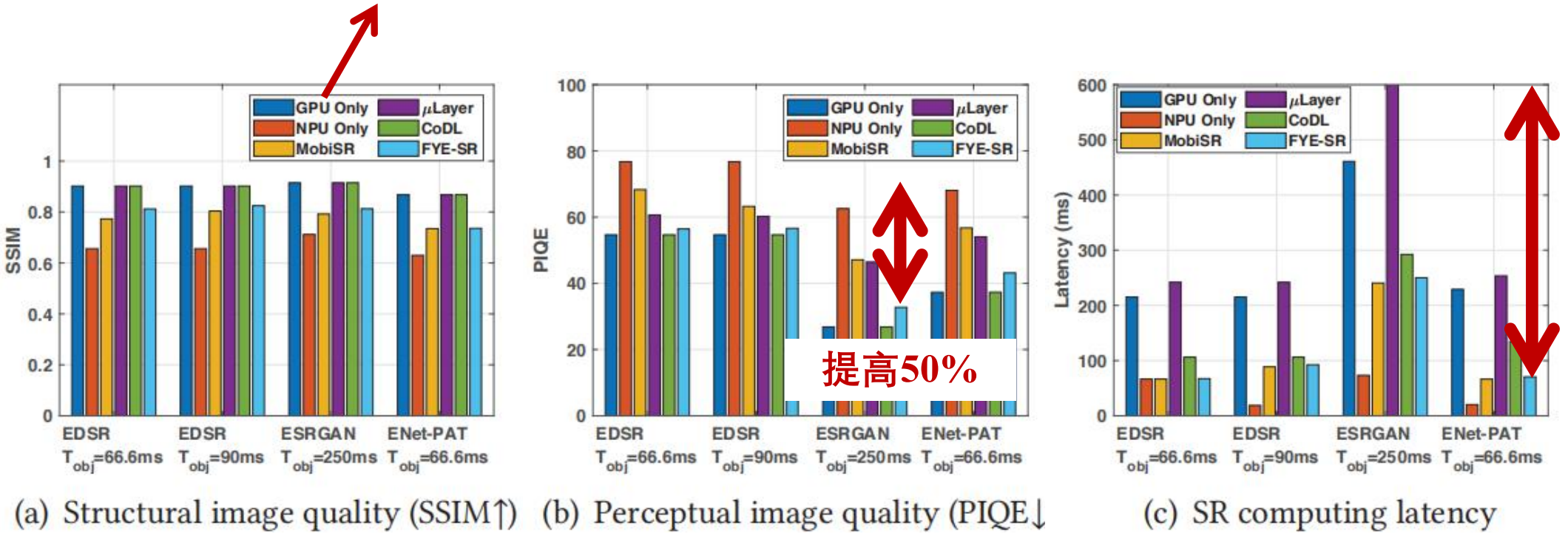
- DIV2K：
  - 2K分辨率，室外和室内照片
- OST300：
  - 户外场景，如建筑、天空和山脉
- Flickr2K：
  - 2K分辨率的城市和自然景观图像

对比算法：

- MobiSR [MobiCom'19]
  - 分割图像，CPU、GPU和AI加速器执行
- µLayer [EuroSys '19]
  - 分割每个NN层的channel，CPU、GPU和AI加速器执行。
- CoDL [MobiSys '22]
  - 分割图像，CPU、GPU执行

Evaluation—Image Quality and Computing Latency

GPU Only: FP16，最好的图像质量，更高的计算延迟。



(a) Structural image quality (SSIM↑) (b) Perceptual image quality (PIQE↓) (c) SR computing latency

在Pixel 6智能手机上使用EDSR、ESRGAN和ENet-PAT模型(小—>大)的SR图像质量和计算延迟

SSIM指标：FYE-SR
更接近GPU Only

PIQE指标：FYE-SR
比其他SOTA高出50%

Latency指标：计算延迟降低1.8×-5.6×

Evaluation—Image Quality and Computing Latency

GPU Only: FP16，最好的图像质量，更高的计算延迟。

(a) GPU Only (PIQE=26.84)

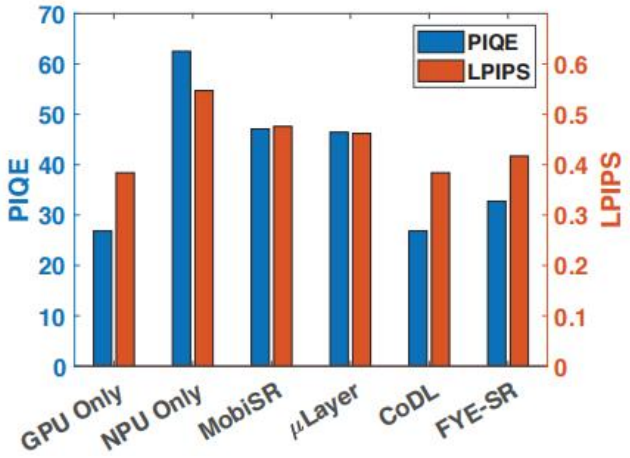
(b) NPU Only (PIQE=62.58)

(c) MobiSR (PIQE=47.10)

(d) FYE-SR (PIQE=32.75)

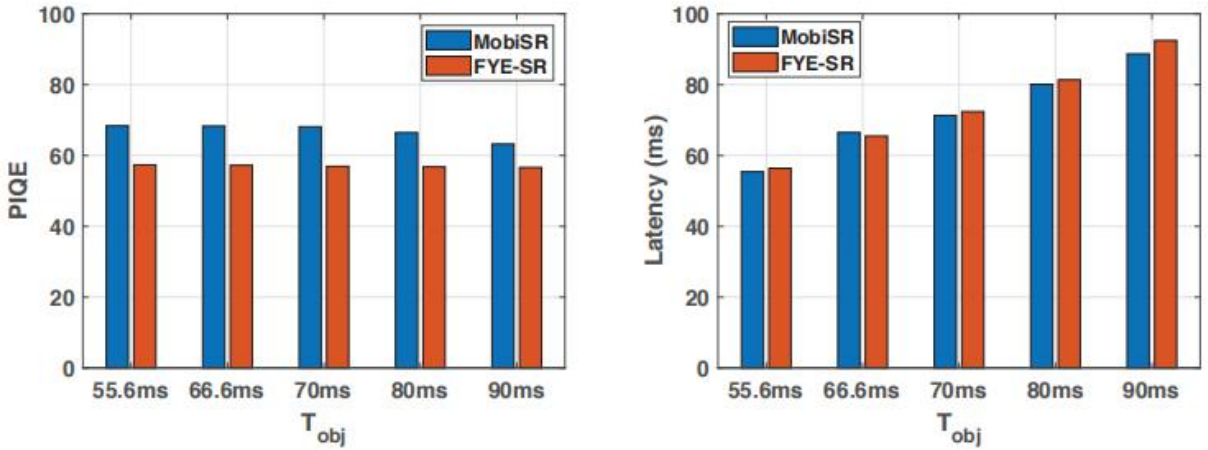
FYE-SR与SOTA利用ESRGAN超分模型的图像对比

更高的图像质量



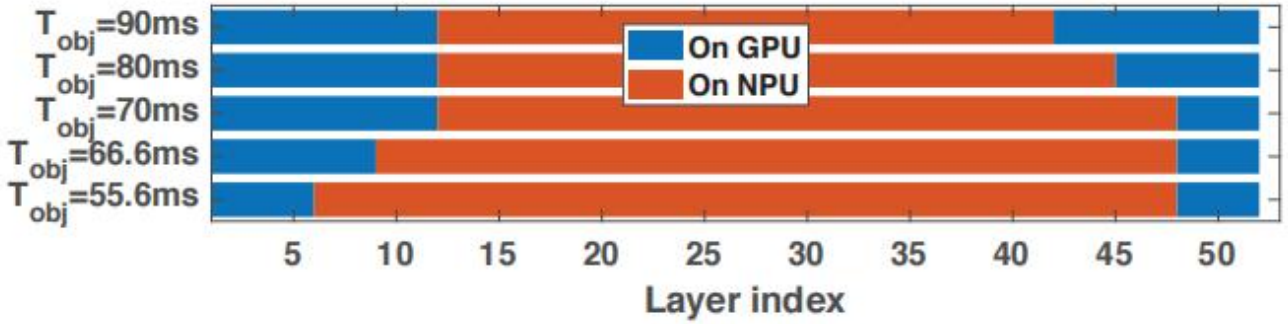
FYE-SR和SOTA在其他指标上的定量比较:
PIQE↓和LPIPS↓

Evaluation—Performance with Different T_{obj}



(a) Perceptual image quality (PIQE↓) (b) SR computing latency

FYE-SR与SOTA在不同的T_{obj}下图像质量和计算延迟的对比



不同的T_{obj}下，EDSR模型在GPU和NPU上的划分

当T_{obj}变化时，FYE-SR在图像质量上持续优于MobiSR，同时达到相同的SR计算延迟。

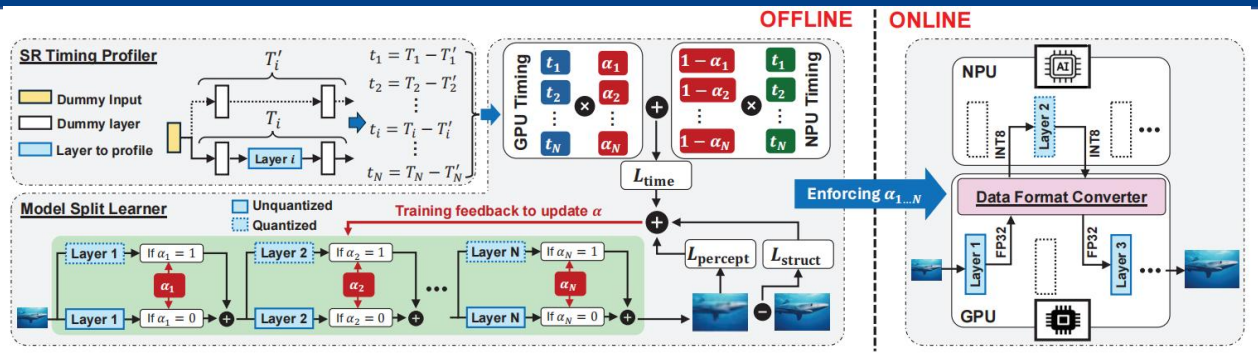
MobiSR根据像素变化跨处理器分割输入图像，这精度较低，低估了像素间的一致性。

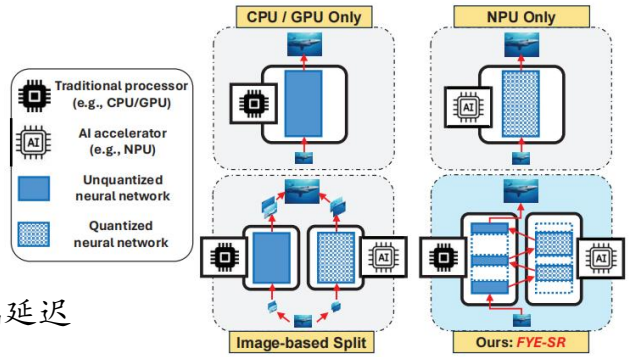
当T_{obj}增加时，FYE-SR能够自适应地调整SR模型的分割，在GPU上执行更多的SR计算，从而导致图像质量的提高。

Outline

- Background
- Motivation
- Design
- Evaluation
- **Conclusion**

Conclusion





- **Problem:** 如何实现超分模型在移动端的低时延&高质量？
  - **Input (Optional):** 低质量图像, SR模型
  - **Output:** 高质量图像
  - **Significance:** 提升图像质量，节约计算资源
- **SoA & Limitation:**
  - **L1:** 过度压缩超分辨率模型会严重损害图像质量
  - **L2:** 模型切分时忽略了AI加速器运算时可能造成的精度损失
- **Opportunities:**
  - **L1 -> O1:** 超分辨率模型在相同参数大小的情况下需要更多的计算量，AI加速器可以加速SR模型的运算。
  - **L2 -> O2:** SR产生的超分图像质量可以通过不同的指标进行评估，不同的指标评估了不同方面的图像质量。
  - **L2 -> O3 (Insight):** SR模型中不同层对量化精度的敏感性不同。
- **Challenges:**
  - **O1 -> C1:** SR模型每一层的端到端时延难以计算。
  - **O1&O2 -> C2:** 如何在保证精度&速度下，实现SR模型的layer-level切分？
  - **O3 -> C3:** 如何减少异构处理器之间数据格式转换的开销？
- **Model:**
  - **C1 -> M1:** SR Timing Profiler: 测量SR模型的不同NN层在传统处理器(例如GPU)和AI加速器(例如NPU)上的端到端延迟
  - **C2 -> M2:** Model Split Learner: 解决最优SR模型分割优化问题
  - **C3 -> M3:** Data Format Converter: 将中间特征图转换为正确的数据格式(例如INT8和FP32)，以便在异构处理器之间正确切换SR模型的计算。
- **3 Contributions:**
  - Conceptually: The first work for 最小化超分模型的端到端延迟并最大化超分图像的感知质量。
  - Technically: SR Timing Profiler + Model Split Learner + Data Format Converter
  - Experimentally: Scale: 3种SR模型，多种时延约束条件，3个HR数据集
    - Baselines:
      - MobiSR [MobiCom'19], µLayer [EuroSys'19], CoDL [MobiSys'22]
    - Metrics: Latency, both structural metrics (PSNR and SSIM) and perceptual metrics (PIQE and LPIPS)

Conclusion

- **这个paper有什么问题，基于这个paper还能做什么？**
  - **(优)这篇论文虽然想法很朴素，但是给我的感觉：**
    - 每一个设计点都做的很扎实、很深入
    - Motivation部分、design部分的图很多，也包含了一些实验结果
    - Evaluation部分，实验分析多
  - **(缺)模型拆分策略：**
    - 在我看来，本文是第一个将NN的切分位置以及感知图像质量一起建模到Loss函数中，之前都是将NN的切分位置建模到Loss函数中(在我认知中)
      - 还有没有其他的指标可以建模进去？一起训练？比如客户端负载动态变化、运行时处理器状态（如温度、负载、功耗）等
    - 之前做CV里面的模型切分，比如ResNet/VGG这种backbone网络的切分，没有一个人做过超分模型的切分? 然后FYE-SR观察到了超分模型上进行模型切分的独立Insight，并展开研究。
  - **(缺)Mobile device部署：**
    - 模型切分有的是用在边缘端(Nvidia Xaiver)，感觉FYE-SR没有充分挖掘出mobile device的Insight(没有说出来Mobile比edge区别在哪里，难在哪里)
      - 比如mobile上散热更差，建模Loss时考虑功耗;
      - mobile上计算能力更差，XXX

Conclusion

- **这个paper提到的idea，能不能用在自己的方向/project上面？**
  - multi/early-exiting
    - 退出出口 + 指标(精度/时延)的联合训练
    - 异构处理器：在哪个处理器上退出？
  - XAI可解释性学习
    - 选择XAI的哪一层对结果贡献更大 + 指标(精度/时延)
- **这个paper能不能泛化，需要较为熟悉这个小方向？**
  - 该论文只是切割了超分的模型，模型拆分方法能否泛化到计算更复杂的模型？
    - 如LLM中的Transformer架构？
  - 能否将该方法适配到资源更受限的物联网设备？例如，低功耗智能家居设备如何高效完成边缘任务
    - [MobiCom23] {NeuriCam} Key-Frame Video Super-Resolution and Colorization for IoT Cameras

TMC候选框过滤：
有些重要的框过滤掉后，会有风险；我们能不能学习一个小的分类模型（0：筛掉，1：选择），Loss设计为精度+时延

Q&A

2024年12月27日

刘天恩

东南大学智慧物联网实验室