推理引擎 - 模型转换与优化

计算图优化

ZOMI

Talk Overview

1. **推理系统介绍**
   - 推理系统架构
   - 推理引擎叫故
2. **模型小型化**
   - CNN小型化结构
   - Transform小型化结构
3. **离线优化压缩**
   - 低比特量化
   - 模型剪枝
   - 知识蒸馏
4. **模型转换与优化**
   - 架构与流程
   - 模型转换技术
   - 计算图优化
5. **Runtime与在线优化**
   - 动态batch
   - bin Packing
   - 多副本并行

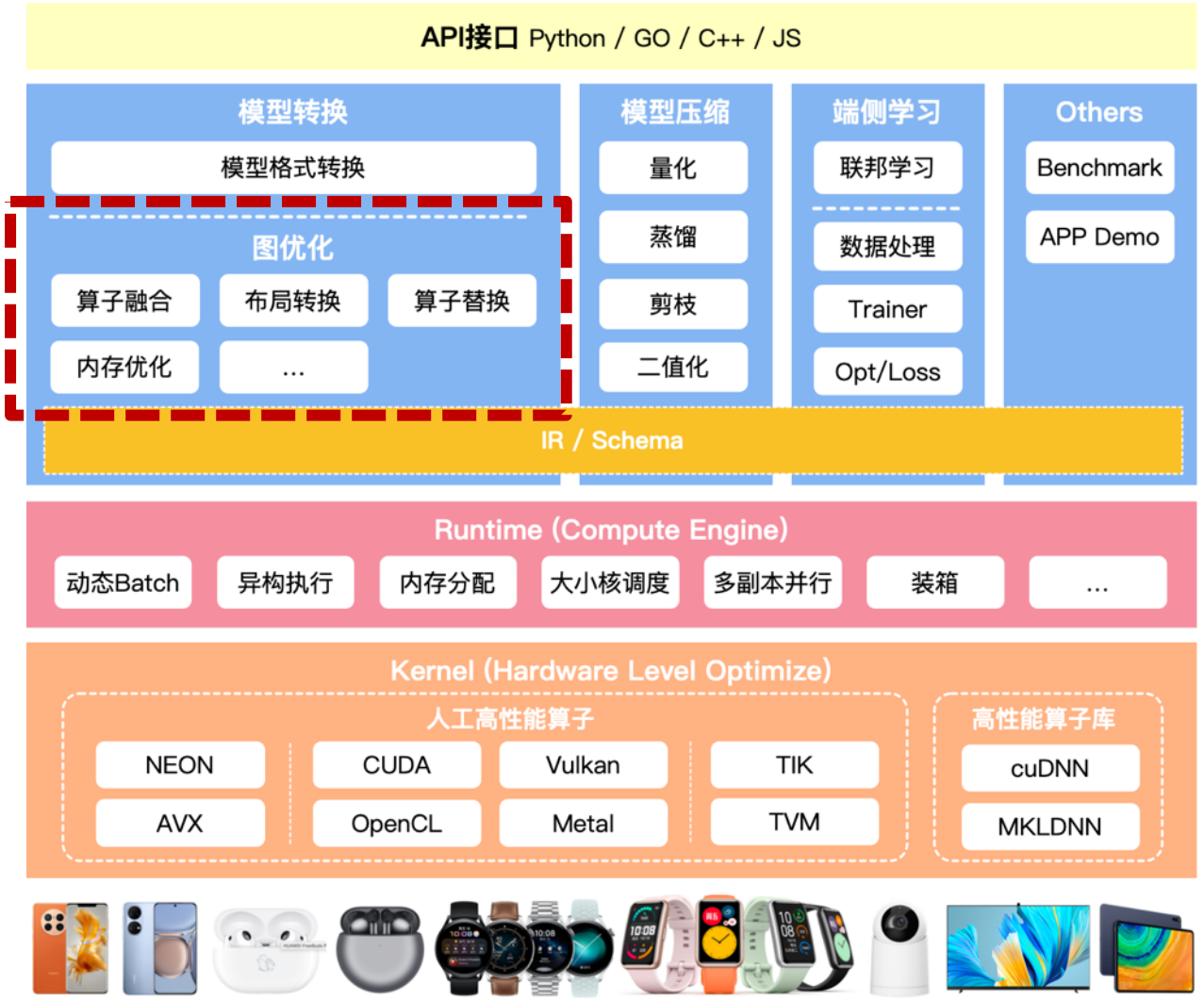
Talk Overview

1. 计算图优化

- Challenges and Architecture - 挑战与架构
- graph Optimization - 计算图优化
- Example - ONNX Runtime 图优化
- Optimize Details - 计算图优化详解

- **基础知识**
- **整体架构**

推理引擎架构



离线优化模块挑战与架构

Optimizer Challenge 优化模块挑战

- **结构冗余**：深度学习网络模型结构中的无效计算节点、重复的计算子图、相同的结构模块，可以在保留相同计算图语义情况下无损去除的冗余类型；
- **精度冗余**：推理引擎数据单元是张量，一般为FP32浮点数，FP32表示的特征范围在某些场景存在冗余，可压缩到 FP16/INT8 甚至更低；数据中可能存大量0或者重复数据。
- **算法冗余**：算子或者Kernel层面的实现算法本身存在计算冗余，比如均值模糊的滑窗与拉普拉斯的滑窗实现方式相同。
- **读写冗余**：在一些计算场景重复读写内存，或者内存访问不连续导致不能充分利用硬件缓存，产生多余的内存传输。

Optimizer Challenge 优化模块挑战

- **结构冗余**：深度学习网络模型结构中的无效计算节点、重复的计算子图、相同的结构模块，可以在保留相同计算图语义情况下无损去除的冗余类型；
- **精度冗余**：推理引擎数据单元是张量，一般为FP32浮点数 存在冗余，可压缩到 FP16/INT8 甚至更低；数据中可能存大
- **算法冗余**：算子或者Kernel层面的实现算法本身存在计算冗余，比如均值模糊的滑窗与拉普拉斯的滑窗实现方式相同。
- **读写冗余**：在一些计算场景重复读写内存 分利用硬件缓存，产生多余的内存传输。

计算图优化

算子融合、算子替换、常量折叠

统一算子/计算图表达

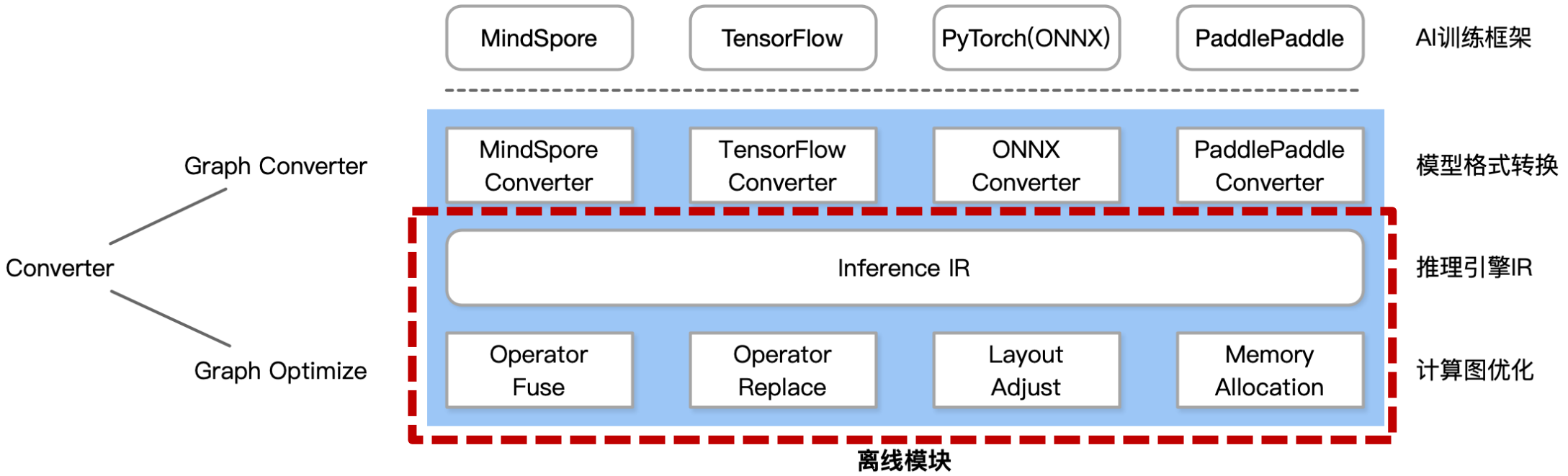
Kernel提升泛化性

数据排布优化

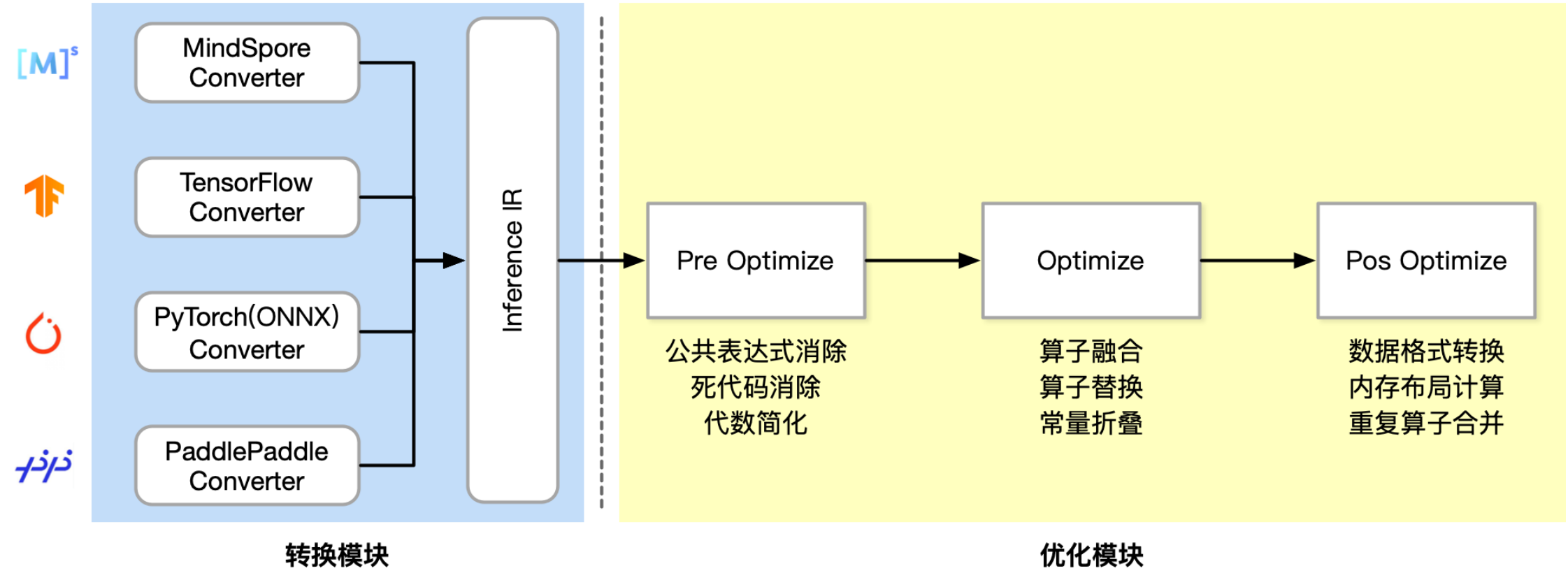
内存分配优化

转换模块架构

- Converter由Frontends和Graph Optimize构成。前者负责支持不同的AI 训练框架；后者通过算子融合、算子替代、布局调整等方式优化计算图：



转换模块的工作流程



离线优化模块

计算图优化

AI编译器之前端优化

播放全部

结构在快速演化，底层计算硬件技术更是层出不穷，对于广大开发者来说不仅要考虑如何在复杂多，还要应对计算框架的持续迭代。AI编译器就成了应对以上问题广受关注的技术方向，让用户仅...

默认排序　升序排序　编辑

AI编译器前端"图层优化"内容概览！！【AI编译器】系列之前端优

600　2022-12-13

图层IR(Graph IR)是什么？AI编译器如何接收图层IR进行优化呢？【AI编

519　2022-12-13

算子融合了解下！AI编译器如何实现算子融合的？【AI编译器】系列

915　2022-12-16

编译器为什么要对数据布局转换呢Layout Transformations？【AI编

404　2022-12-17

详解AI编译器数据布局转换方法！华为昇腾处理器的数据布局格式！

225　2022-12-17

AI编译器内存优化算法！动态内存和静态内存区别！【AI编译器】前

328　1-16

常量折叠原理！AI编译器常量折叠跟传统编译器什么关系？计算图也

356　2022-12-18

编译器公共子表达式消除的方法！AI编译器消除公共子表达式【AI编译

270　2022-12-20

编译器死代码消除的原理！AI编译器死代码消除【AI编译器】系列之

375　2022-12-20

编译器的代数化简原理！AI编译器的代数化简来啦！【AI编译器】系

315　2022-12-21

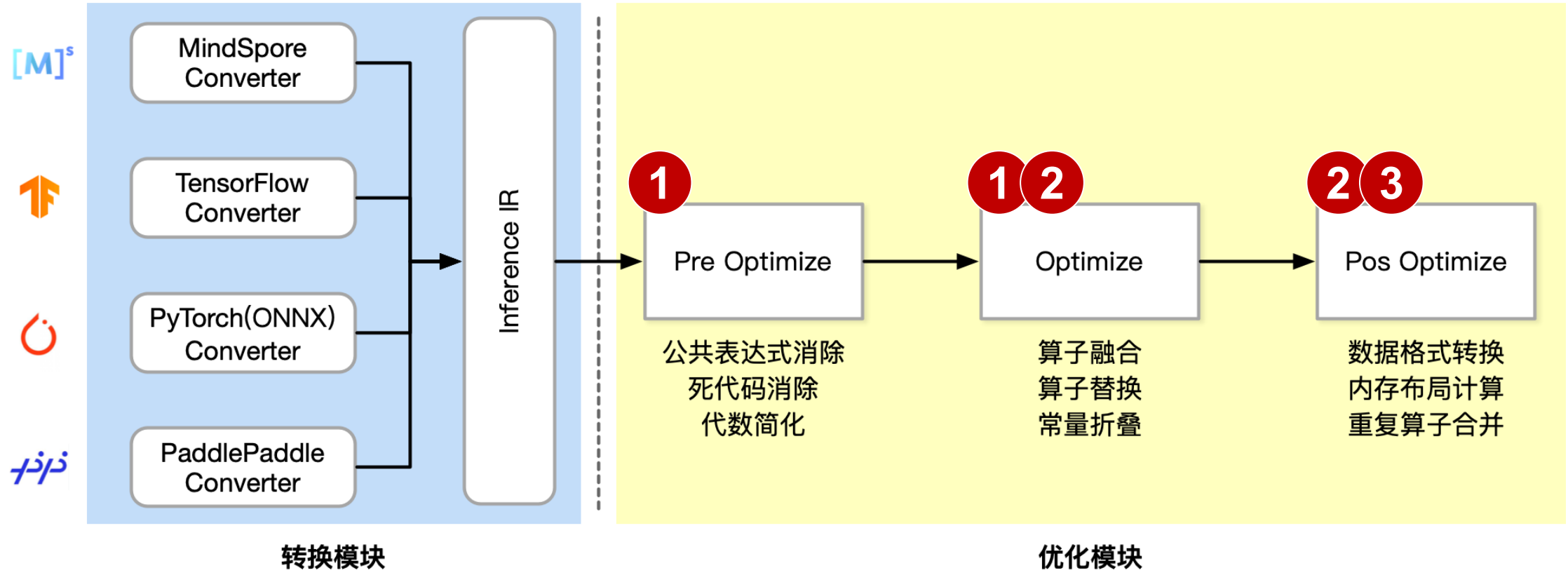
计算图优化 (a.k.a., 图优化)

- **图优化**：基于一系列预先写好的模板，减少转换模块生成的计算图中的冗余计算，比如 Convolution 与 Batch Normal / Scale 的合并，Dropout 去除等。图优化能在特定场景下，带来相当大的计算收益，但相当依赖根据先验知识编写的模板，相比于模型本身的复杂度而言注定是稀疏的，无法完全去除结构冗余。

图优化方式

1. **Basic:** 基础优化涵盖了所有保留计算图语义的修改，如：O1常量折叠、O2冗余节点消除和O3有限数量的算子融合。
2. **Extended:** 扩展优化仅在运行特定后端，如 CPU、CUDA、NPU 后端执行提供程序时适用。其针对硬件进行特殊且复杂的 Kernel 融合策略和方法。
3. **Layout & Memory:** 布局转换优化，主要是不同 AI 框架，在不同的硬件后端训练又在不同的硬件后端执行，数据的存储和排布格式不同。

工作流程



ONNX

Runtime图优化

ONNX Usage

- ONNX Runtime defines the GraphOptimizationLevel enum to determine which of the aforementioned optimization levels will be enabled. Choosing a level enables the optimizations of that level, as well as the optimizations of all preceding levels. For example, enabling Extended optimizations, also enables Basic optimizations. The mapping of these levels to the enum is as follows:

```
1
2   GraphOptimizationLevel::ORT_DISABLE_ALL -> Disables all optimizations
3   GraphOptimizationLevel::ORT_ENABLE_BASIC -> Enables basic optimizations
4   GraphOptimizationLevel::ORT_ENABLE_EXTENDED -> Enables basic and extended optimizations
5   GraphOptimizationLevel::ORT_ENABLE_ALL -> Enables all available optimizations including layout optimizations
6
```

ONNX Usage

- ONNX Runtime defines the GraphOptimizationLevel enum to determine which of the aforementioned optimization levels will be enabled. Choosing a level enables the optimizations of that level, as well as the optimizations of all preceding levels. For example, enabling Extended optimizations, also enables Basic optimizations. The mapping of these levels to the enum is as follows:

```
1
2   Ort::SessionOptions session_options;
3
4   // Set graph optimization level
5   session_options.SetGraphOptimizationLevel(GraphOptimizationLevel::ORT_ENABLE_EXTENDED);
6
7   // To enable model serialization after graph optimization set this
8   session_options.SetOptimizedModelFilePath("optimized_file_path");
9
10  auto session_ = Ort::Session(env, "model_file_path", session_options);
11
```