推理引擎 - 模型转换与优化

模型转换流程

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
   - 模型转换技术细节
   - 模型离线优化
5. **Runtime与在线优化**
   - 动态batch
   - bin Packing
   - 多副本并行

Talk Overview

I. 模型格式转换

- 转换模块挑战与架构
- 模型序列化/反序列化
- protobuffer / flatbuffer 格式
- 自定义计算图 IR
- 转换流程和技术细节

- **工程理论**
- **知识概念**

Talk Overview

I. 模型格式转换

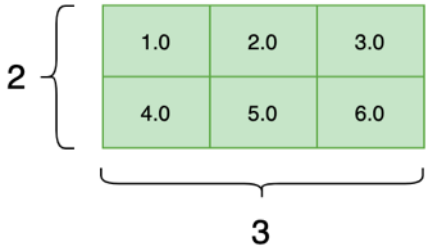
- 转换模块挑战与架构
- 模型序列化/反序列化
- protobuffer / flatbuffer 格式
- 自定义计算图 IR
- 转换流程和技术细节

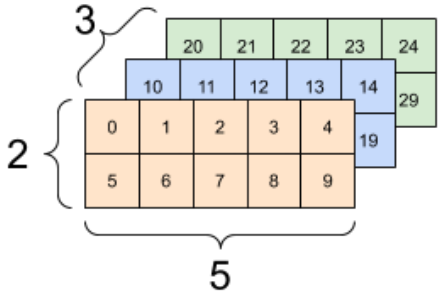
- **技术细节**
- **核心内容**

基于计算图的AI框架：基本组成

基本数据结构：Tensor 张量

- Tensor形状： [2, 3, 4, 5]
- 元素类型：int, float, string, etc.





基本运算单元：Operator 算子

- 由最基本的代数算子组成
- 根据深度学习结构组成复杂算子
- N个输入Tensor，M个输出Tensor

Add	Log	While
Sub	MatMul	Merge
Mul	Conv	BroadCast
Div	BatchNorm	Reduce
Relu	Loss	Map
Floor	Sigmoid	…..

推理引擎计算图：Tensor 张量的表示

Tensor 数据存储格式

```
// 定义 Tensor 的数
enum DataType : i
  DT_INVALID = 0,
  DT_FLOAT = 1,
  DT_DOUBLE = 2,
  DT_INT32 = 3,
  DT_UINT8 = 4,
  DT_INT16 = 5,
  DT_INT8 = 6,
  // ...
}
```

Tensor 数据内存排布格式

```
// 定义 Tensor 数据排布格
enum DATA_FORMAT : byte
    ND,
    NCHW,
    NHWC,
    NC4HW4,
    NC1HWC0,
    UNKNOWN,
    // ...
}
```

Tensor 张量的定义

```
// 定义 Tensor
table Blob {
    // shape
    dims: [int];
    dataFormat: DATA_FORMAT;

    // data type
    dataType: DataType = DT_FLOAT;

    // extra
    // ...
}
```

推理引擎计算图： Operator 算子的表示

算子列表

```
37
38  // 推理引擎算子
39  enum OpType :
40      Const,
41      Convolut:
42      Convolut:
43      Deconvolu
44      Deconvolu
45      MatMul,
46      Padding,
47      // ...
48  }
49
```

算子公共属性和特殊算子列表

```
49
50  // 算子的公共属性和特
51  union OpParameter
52      WhileParam,
53      IfParam,
54      PadParam,
55      Range,
56      Act,
57      // ...
58  }
59
```

算子的基础定义

```
59
60  // 算子基础定义
61  table Op {
62      inputIndexes: [int];
63      outputIndexes: [int];
64      main: OpParameter;
65      type: OpType;
66      name: string;
67      // ...
68  }
69
```

推理引擎计算图： 计算图的表示

定义网络模型子图

```
15  // 子图概念的定义
16  table SubGraph {
17      // Subgraph unique name.
18      name: string;
19      inputs: [int];
20      outputs: [int];
21
22      // All tensor names.
23      tensors: [string];
24
25      // Nodes of the subgraph.
26      nodes: [Op];
27  }
```

定义网络模型

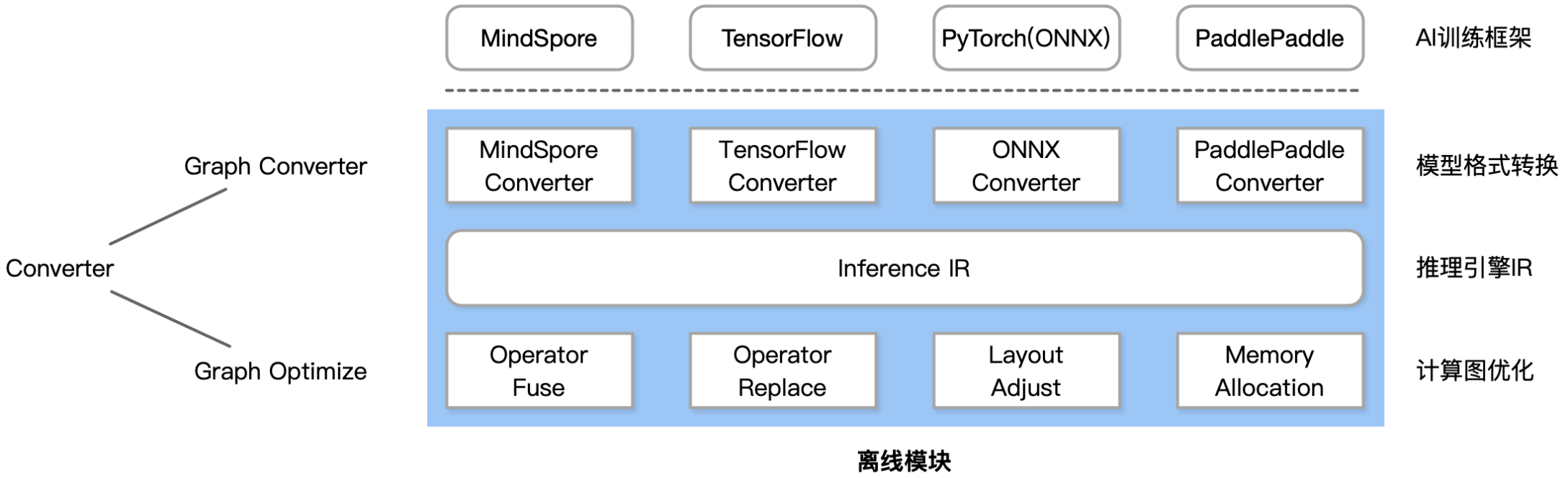
```
2   // 网络模型定义
3   table Net {
4       name: string;
5       inputName: [string];
6       outputName: [string];
7       oplists: [Op];
8       sourceType: NetSource;
9
10      // Subgraphs of the Net.
11      subgraphs: [SubGraph];
12      // ...
13  }
14
```

模型转换流程

技术细节

模型转换技术在设计思路

1. **直接转换**：直接将网络模型从 AI 框架转换为适合目标框架使用的格式；
2. **规范式转换**：设计一种开放式的文件规范，使得主流 AI 框架都能实现对该规范标准的支持；



直接转换

1. **内容读取**：读取 A 框架生成的模型文件，并识别模型网络中的张量数据的类型/格式、算子的类型和参数、计算图的结构和命名规范，以及它们之间的其他关联信息。
2. **格式转换**：将 step1 识别得到的模型结构、模型参数信息，直接代码层面翻译成推理引擎支持的格式。当然，算子较为复杂时，可在 Converter 中封装对应的算子转换函数来实现对推理引擎的算子转换。
3. **模型保存**：在推理引擎下保存模型，可得到推理引擎支持的模型文件，即对应的计算图的显示表示。

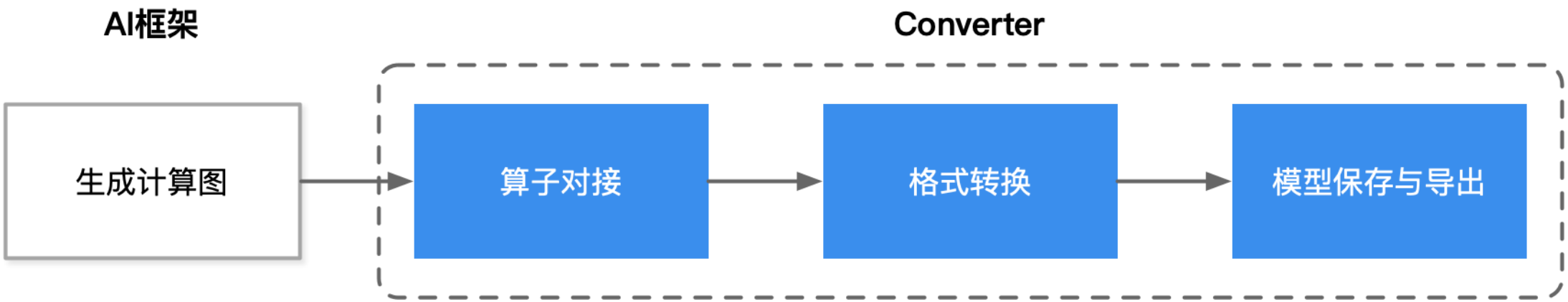
规范式转换 — 以 ONNX 为代表

- ONNX是一种针对机器学习所设计的开放式文件格式，用于存储训练好的网络模型。它使得不同的 AI 框架 (如Pytorch, MindSpore) 可以采用相同格式存储模型数据并交互。

- ONNX 定义了**一种可扩展的计算图模型、一系列内置的运算单元(OP)和标准数据类型**。每一个计算流图都定义为**由节点组成的列表**，并构建**有向无环图**。其中每一个节点都有一个或多个输入与输出，每一个节点称之为一个 **OP**。

模型转换通用流程

1. AI框架生成计算图（以静态图表示），常用基于源码 AST 转换和基于 Trace 的方式；
2. 对接主流通用算子，并重点处理计算图中的自定义算子；
3. 目标格式转换，将模型转换到一种中间格式，即推理引擎的自定义 IR；
4. 根据推理引擎的中间格式 IR，导出并保存模型文件，用于后续真正推理执行使用。



参考文献

1. Huawei Technologies Co., Ltd. "Huawei MindSpore AI Development Framework." *Artificial Intelligence Technology*. Singapore: Springer Nature Singapore, 2022. 137-162.
2. Jiang, Xiaotang, et al. "Mnn: A universal and efficient inference engine." *Proceedings of Machine Learning and Systems* 2 (2020): 1-13.
3. https://onnx.ai/supported-tools
4. https://github.com/onnx/onnx/blob/main/docs/IR.md
5. https://gitee.com/mindspore/mindspore
6. https://github.com/alibaba/MNN
7. https://onnxruntime.ai/