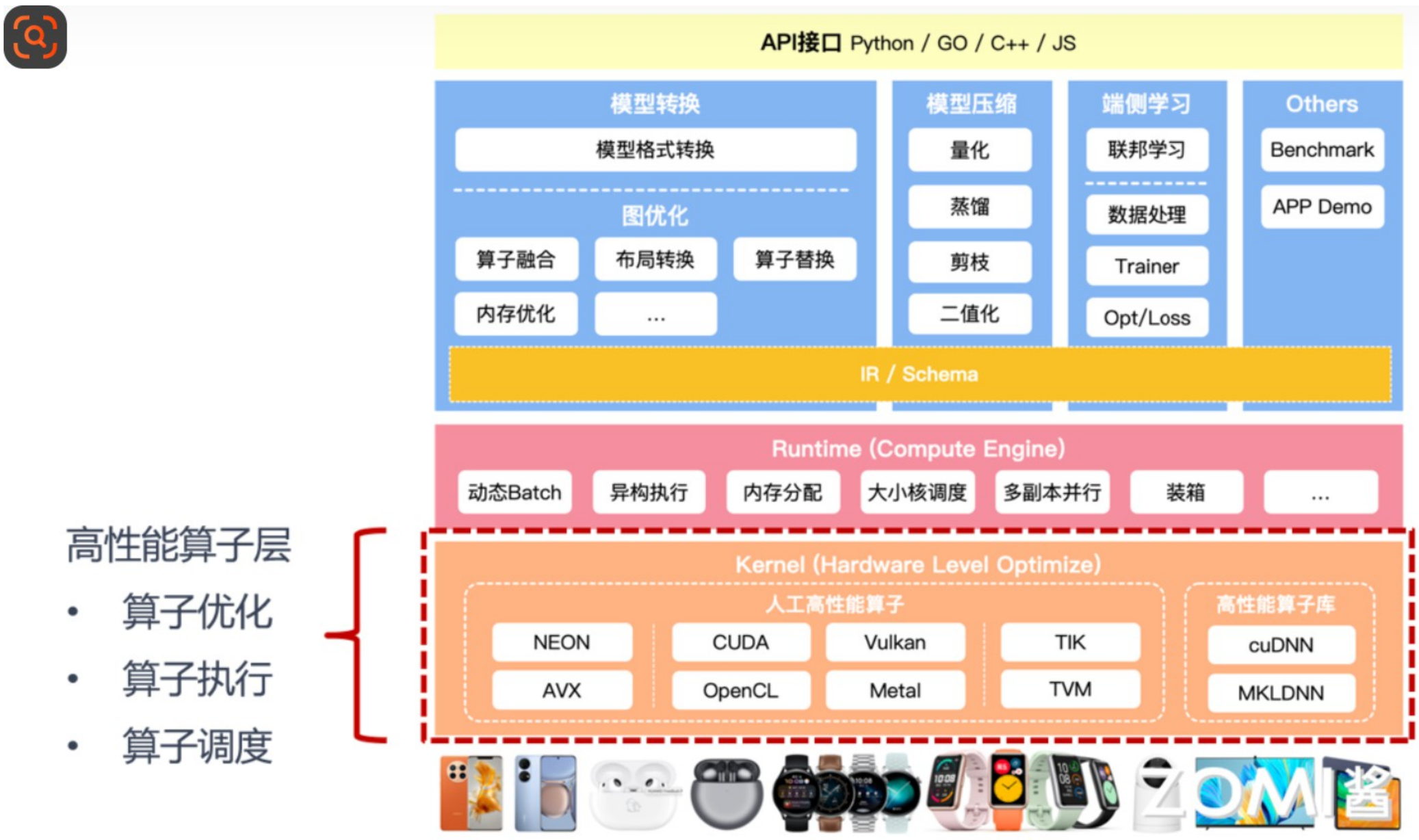


【AI系统】Kernel 层架构

ZOMI酱
AIInfra 制造机

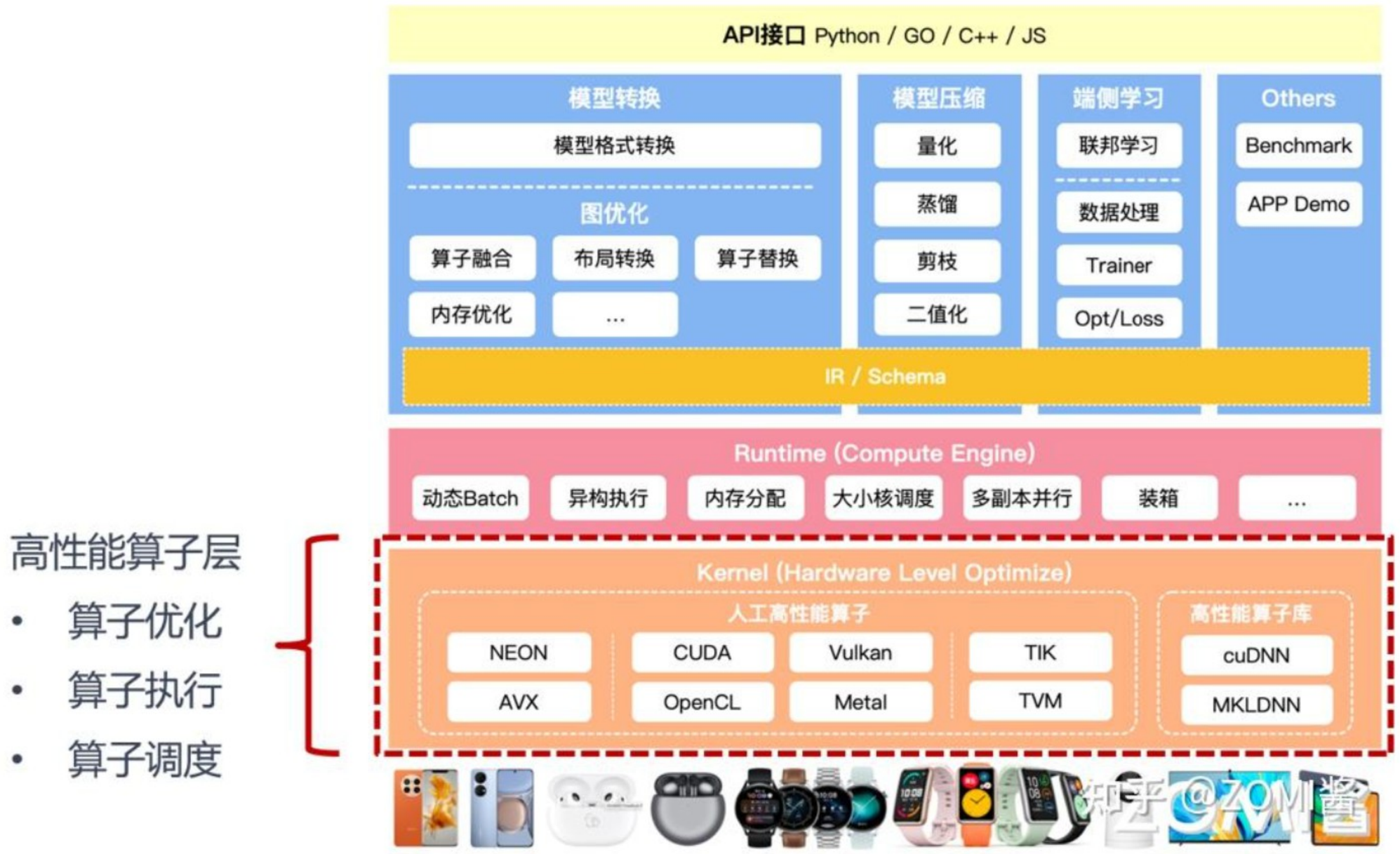
推理引擎的 Kernel 层通常是推理引擎中用于执行底层数学运算的组件。在神经网络模型推理过程中，需要对大量数据进行高效的数学运算，如矩阵乘法、卷积、池化等。Kernel 层就是实现这些运算的核心部分，它直接影响着推理引擎的速度和效率，因此本文将会重点介绍 Kernel 层相关的内容。

Kernel 层介绍

在推理引擎架构中，Runtime 层和 Kernel 层是紧密相连的两个组件。

Runtime 提供了一个执行环境，它管理整个推理过程，包括模型的加载、预处理、执行和后处理。它还负责资源管理，如内存分配和线程管理。其通常具有平台无关性，可以在不同的操作系统和硬件上运行，为上层应用提供 API 接口，使得用户能够轻松地集成和使用神经网络模型。

Kernel 层包含了一系列的低级函数，它们直接在硬件上执行数学运算，如卷积、矩阵乘法和激活函数。其通常是硬件特定的，针对不同的 AI 加速芯片有不同的实现。Kernel 层实现时，会进行各种优化，包括算法优化、内存访问优化、向量化、并行化和硬件特定的汇编优化。如下图所示为在推理引擎中的 Kernel 层。



推理引擎架构

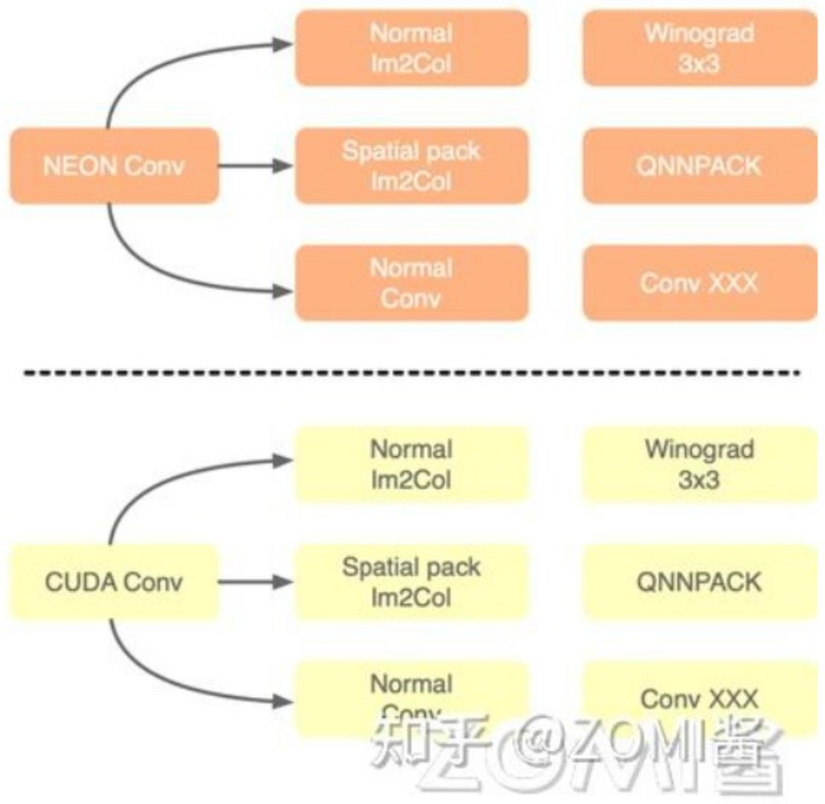
从层与层之间集成的角度来说，Runtime 层会根据模型的操作和目标硬件选择合适的 Kernel 层来执行计算。Runtime 的决策（如算子融合、内存管理等）会影响到 Kernel 层的性能表现。Kernel 层作为 Runtime 层的一部分，被集成到整个推理流程中。

而从交互和适配的角度来说，Runtime 层负责调用 Kernel 层提供的函数，传递必要的输入数据，并处理 Kernel 层的输出结果。它可能会提供一些适配层（adapter layer），以便在不同的硬件上运行相同的 Kernel 代码，或者将不同硬件的 Kernel 接口统一化。Runtime 层的优化和 Kernel 层的优化共同决定了整个推理引擎的性能。

总体而言，Runtime 提供了一个高层次的抽象，管理模型的执行和资源，而 Kernel 层则是底层的实现，直接在硬件上执行数学运算。Kernel 层的优化主要体现在各种高性能算子和算子库的设计上，这些算子和算子库通常针对不同的 AI 加速芯片进行了优化，以提高计算速度和效率。

推理架构

如图所示，下面分别从 CPU 和 GPU 的角度介绍一下几种人工实现的高性能算子和封装的高性能算子库：



CPU 优化：

1. **NEON**：NEON 是 ARM 架构上的 SIMD（单指令多数据）扩展，用于提高多媒体处理和浮点运算的性能。推理引擎可以利用 NEON 指令集来优化 Kernel 层，特别是在移动设备和嵌入式设备上;
2. **AVX**：AVX（Advanced Vector Extensions）是 Intel 处理器上的 SIMD 指令集，用于提高浮点运算和整数运算的性能。推理引擎可以利用 AVX 指令集来优化 Kernel 层，特别是在 Intel CPU 上;
3. **Metal**：Metal 是苹果开发的低级图形和计算 API，用于优化在 Apple GPU 上的性能。推理引擎可以利用 Metal API 来优化 Kernel 层，特别是在 iOS 和 macOS 设备上;
4. **TVM**：TVM（Tensor Virtual Machine）是一个开源的深度学习编译器框架，用于优化神经网络模型在各种硬件上的性能。它支持 CPU、GPU、TPU 和其他类型的硬件。

GPU 优化：

1. **CUDA**：CUDA 是英伟达的并行计算平台和编程模型，用于在英伟达 GPU 上执行并行计算。推理引擎可以利用 CUDA 来优化 Kernel 层，特别是在大规模矩阵运算和卷积操作方面;
2. **OpenCL**：OpenCL 是一个开放的标准，用于编写在异构系统上运行的程序。它允许开发者利用 CPU、GPU 和其他类型的处理器来加速计算密集型任务。推理引擎可以利用 OpenCL 来优化 Kernel 层，特别是在 GPU 上；
3. **Vulkan**：Vulkan 是新一代的图形和计算 API，用于在各种 GPU 上执行并行计算。推理引擎可以利用 Vulkan API 来优化 Kernel 层，特别是在高性能计算和图形处理方面；
4. **Tensor Cores**：Tensor Cores 是英伟达 GPU 上的一种特殊类型的核心，专门用于加速矩阵乘法和卷积操作。推理引擎可以利用 Tensor Cores 来优化 Kernel 层，特别是在执行大规模的矩阵运算时。

此外，封装的高性能算子库有：

1. **cuDNN（CUDA Deep Neural Network Library）**：由英伟达开发，为 GPU 优化的神经网络算子库，包括卷积、池化、归一化、激活函数等；
2. **MKL-DNN（Intel Math Kernel Library for Deep Neural Networks）**：由 Intel 开发，为 CPU 优化的神经网络算子库，现在发展成为 oneDNN，支持多种 Intel 处理器；
3. **MIOpen**：由 AMD 开发，为 GPU 优化的深度学习算子库，特别针对 AMD 的 GPU 架构进行了

优化；

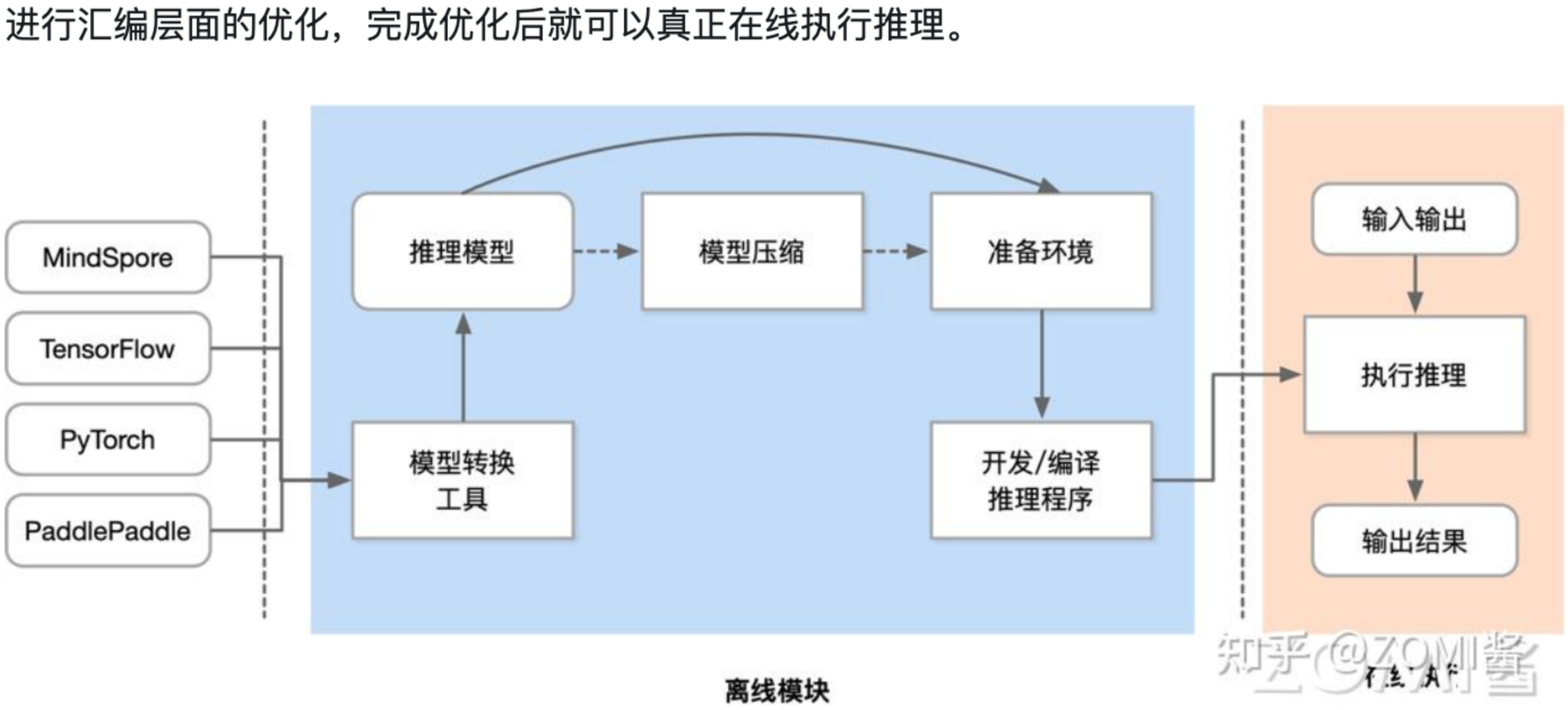
4. **TensorRT**：英伟达的深度学习推理优化器，它提供了 C++和 Python 接口，可以对模型进行优化并生成高性能的推理引擎；
5. **ONNX Runtime**：支持 ONNX 模型的跨平台推理引擎，包括了对多种硬件平台的高性能算子实现；
6. **ACL（ARM Compute Library）**：由 ARM 开发，为 ARM 架构的 CPU 和 GPU 提供优化的算子库，包括卷积、池化、全连接层等。

这些算子方面的优化方法和技术可以根据具体的硬件平台和模型需求来选择和组合，以提高推理引擎在 Kernel 层的性能。在实际应用中，开发者需要根据目标设备和性能要求来选择最合适的优化策略。

推理流程

推理引擎的推理流程是指从加载模型到输出推理结果的一系列步骤。首先加载预先训练好的模型文件，这些文件可能是以特定格式（如 ONNX、TensorFlow SavedModel、PyTorch TorchScript 等）保存的。

此外，对模型结构，包括层的类型、参数和拓扑结构进行解析。之后将模型转换或编译为推理引擎能够执行的格式，这其中可能包括优化模型结构、融合某些层以减少计算和内存访问、选择合适的 Kernel 实现等操作。对于支持硬件加速的推理引擎，这一步可能还包括为特定 AI 加速芯片生成优化的执行代码。在上述离线模块生成的推理模型经过模型压缩（或者不压缩）后送入编译模块再次进行汇编层面的优化，完成优化后就可以真正在线执行推理。

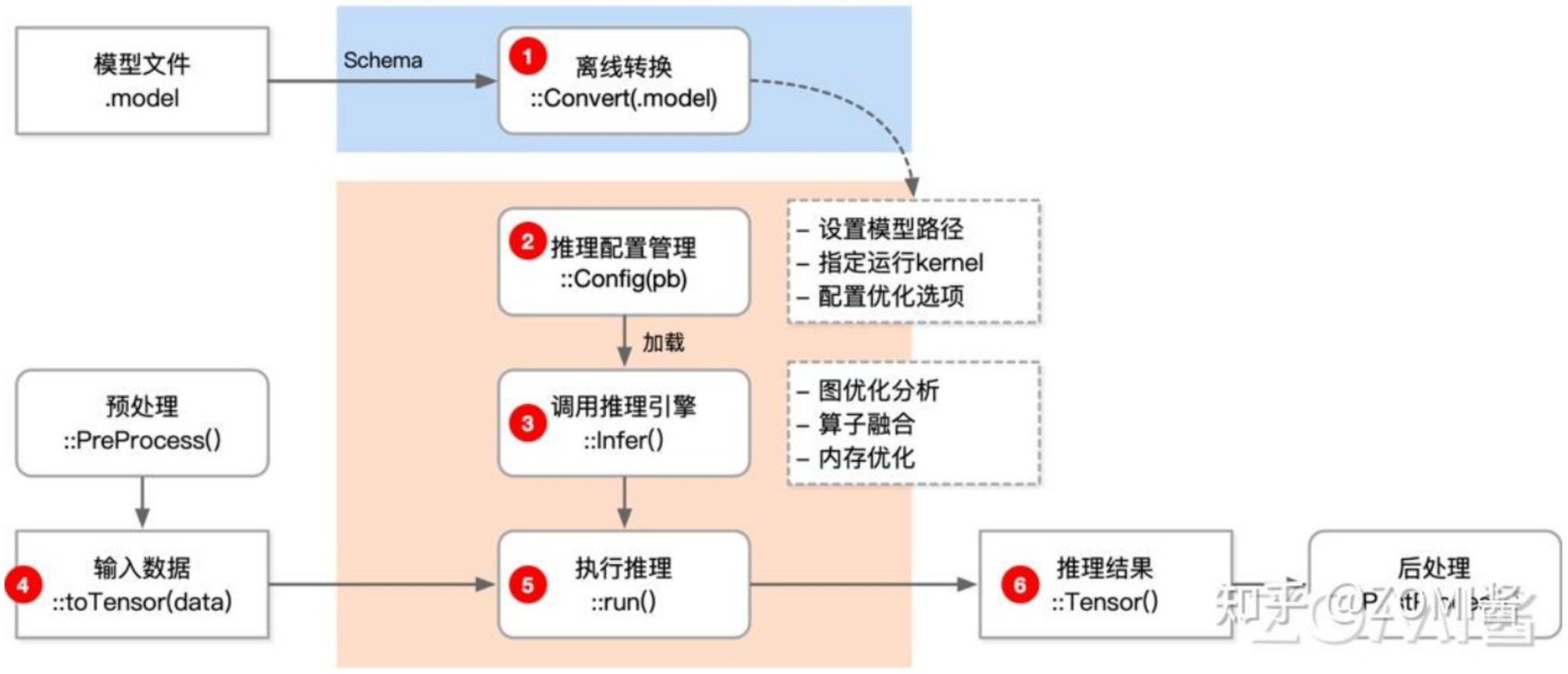


在线执行阶段，将输入数据（如图像、文本等）转换为模型所需的格式，将预处理后的输入数据复制到设备内存中，为推理做准备。在推理引擎中执行模型，最后将处理后的推理结果返回给用户或下游应用程序。

整个推理流程的目标是高效、准确地执行模型推理，同时尽可能减少计算和延迟。

开发推理程序

Kernel 层所进行的优化一般蕴含在下图中的⑤进行执行，Runtime 会根据模型的操作和目标硬件选择合适的 Kernel 来执行计算。如图所示，Kernel 层作为 Runtime 的一部分，被集成到整个推理流程中。



开发推理程序示意图

其负责调用 Kernel 层提供的函数，传递必要的输入数据，并处理 Kernel 的输出结果。总的来说，Runtime 层的优化和 Kernel 层的优化共同决定了整个推理引擎的性能。Runtime 的决策（如算子融合、内存管理等）会影响到 Kernel 层的性能表现。

Kernel 层优化方法

算法优化

算法优化主要针对卷积算子的计算进行优化。卷积操作是神经网络模型中计算密集且耗时的部分，因此对其进行优化能够显著提升推理性能。其中，对于卷积 Kernel 算子的优化主要关注 Im2Col、Winograd 等算法的应用。这些算法通过特定的数学变换和近似，减少了卷积操作的计算复杂度，从而提升了推理速度。其主要方法有：

1. **空间组合优化算法**：将大卷积分解为小卷积，减少内存访问次数，提高缓存利用率。
2. **Im2Col/Col2Im**：将输入图像和卷积核转换为列向量形式，使用矩阵乘法来实现卷积，可以利用高效矩阵乘法库。
3. **Winograd 算法**：通过预计算和转换，减少卷积中的乘法次数，特别适用于小尺寸的卷积核。
4. **快速傅里叶变换（FFT）**：对于大尺寸的卷积核，使用 FFT 将空间域的卷积转换为频域的点乘，提高计算效率。

内存布局

在内存布局方面，主要的方法有：

1. **权重和输入数据的重排**：重新组织权重和输入数据的内存布局，使得数据访问更加连续，减少缓存缺失。
2. **通道优先（Channel First/Last）**：根据硬件特性选择通道数据的存储顺序，例如 NCHW 或 NHWC。
3. **NC1HWC0 与 NCHW4**：NC1HWC0 布局通常用于支持 Winograd 算法或其他需要特定通道分组操作的卷积算法。这种布局是针对特定硬件（如某些 AI 加速器）优化的，其中 C 被分割为

C1 和 C0，C1 代表通道的分组数，C0 代表每个分组中的通道数。这种布局可以减少内存访问次数，提高缓存利用率，并可能减少所需的内存带宽。NCHW4 是一种特殊的内存布局，其中 C 被分割为 4 个通道，这种布局通常用于支持 4 通道的 SIMD 操作。NCHW4 布局可以在支持 4 通道向量化指令的硬件上提供更好的性能，例如某些 ARM 处理器。这种布局可以减少数据填充（padding）的需要，并提高数据处理的并行度。

汇编优化

从汇编优化的方面，主要的方法有：

1. **指令优化**：针对特定 CPU 指令集（如 AVX、AVX2、SSE 等）进行优化，使用向量化指令来提高计算效率。
2. **循环优化**：减少循环的开销，增加指令级的并行性。
3. **存储优化**：优化内存访问模式，使得数据访问更加连续，以提高缓存命中率和内存带宽利用率。

调度优化

从调度优化的方面，则主要有并行计算和自动调优等方法。根据操作间的依赖关系和执行时间，合理调度任务，减少等待时间和提高资源利用率。

推理引擎的 Kernel 层是整个推理系统的基础，对于实现高性能的深度学习推理至关重要。随着深度学习应用的普及和硬件技术的进步，推理引擎的 Kernel 层也在不断地发展和优化，以适应更加多样化的应用场景和性能要求。