TensorRT基础知识

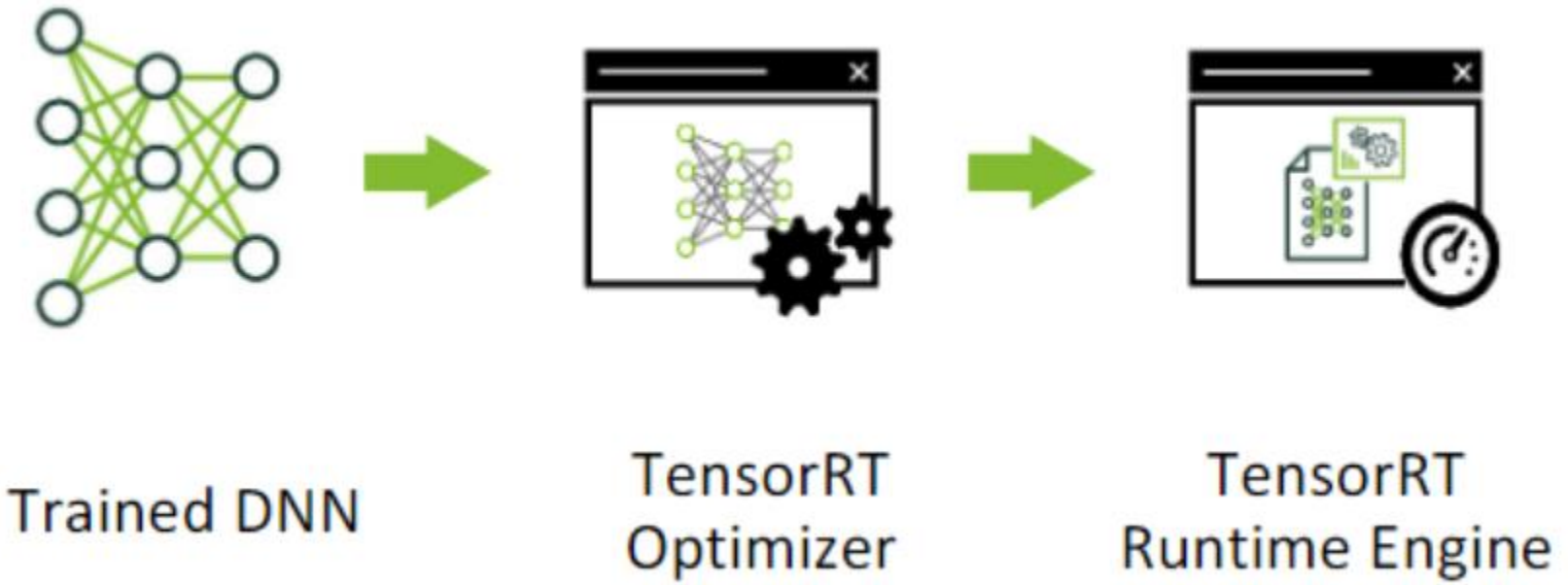
TensorRT是什么?

TensorRT是用于优化训练后的深度学习模型以实现高性能推理的**SDK**。

TensorRT包含用于训练后的深度学习模型的深度学习推理优化器，以及用于执行的runtime。

TensorRT能够以更高的吞吐量和更低的延迟运行深度学习模型。

TensorRT is a high-performance neural network inference optimizer and runtime engine for production deployment.



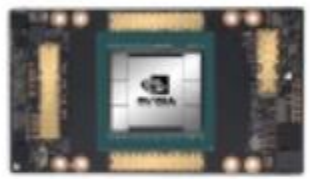
Embedded

Automotive

Data center

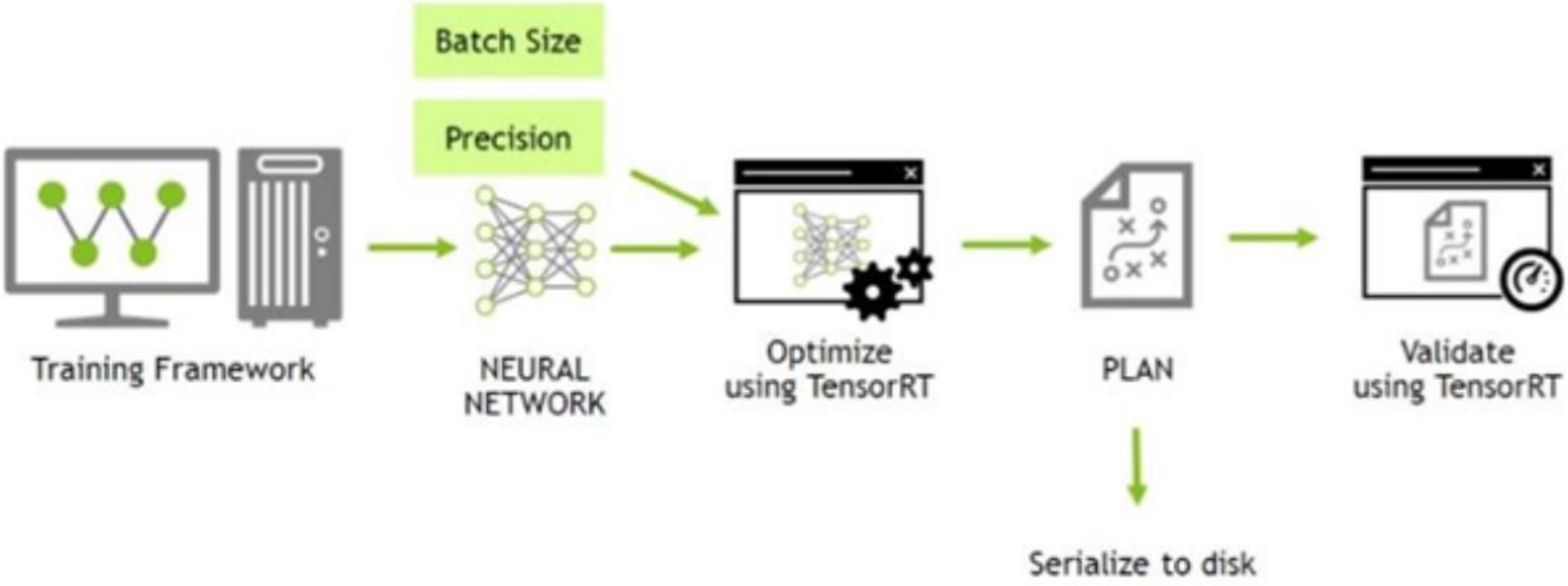
Jetson

Drive

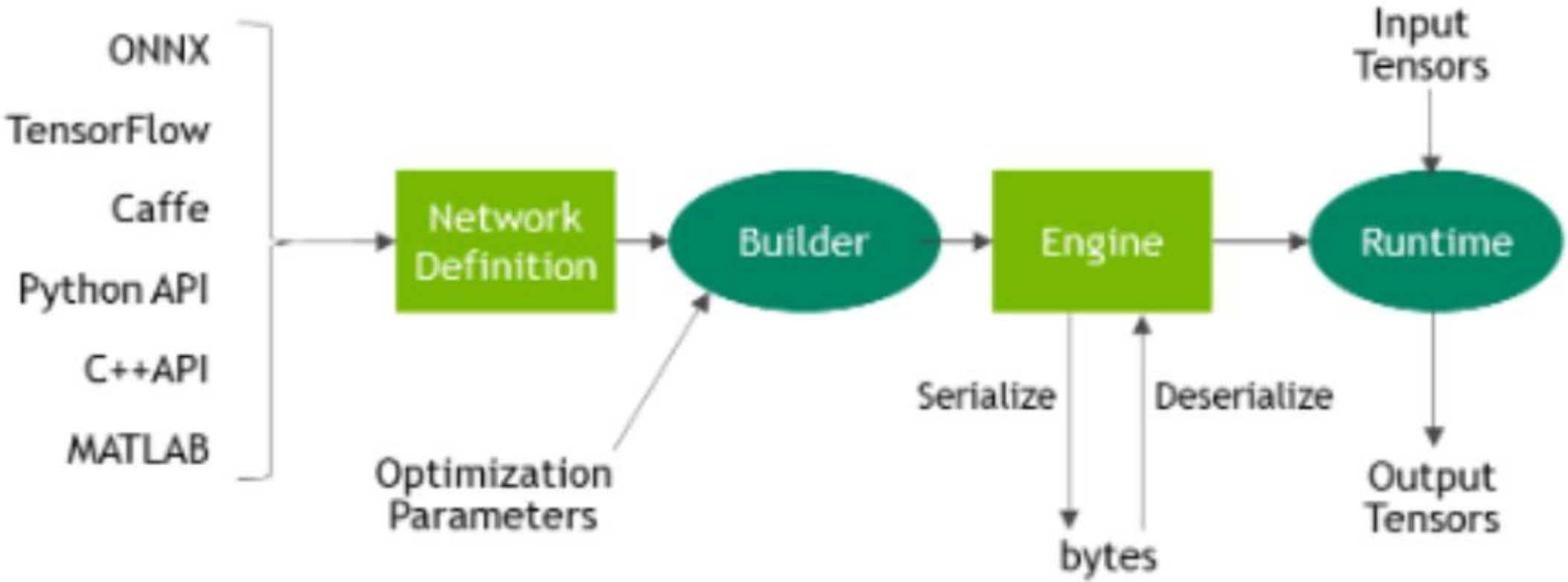


Data Center GPUs

Typical deep learning development cycle using TensorRT



wts


ONNX
TensorFlow
Caffe
Python API
C++API
MATLAB
Network Definition
Builder
Engine
Runtime
Input Tensors
Output Tensors
Optimization Parameters
Serialize
Deserialize
bytes

Builder TensorRT的模型优化器。 构建器将网络定义作为输入，执行与设备无关和针对特定设备的优化，并创建引擎。

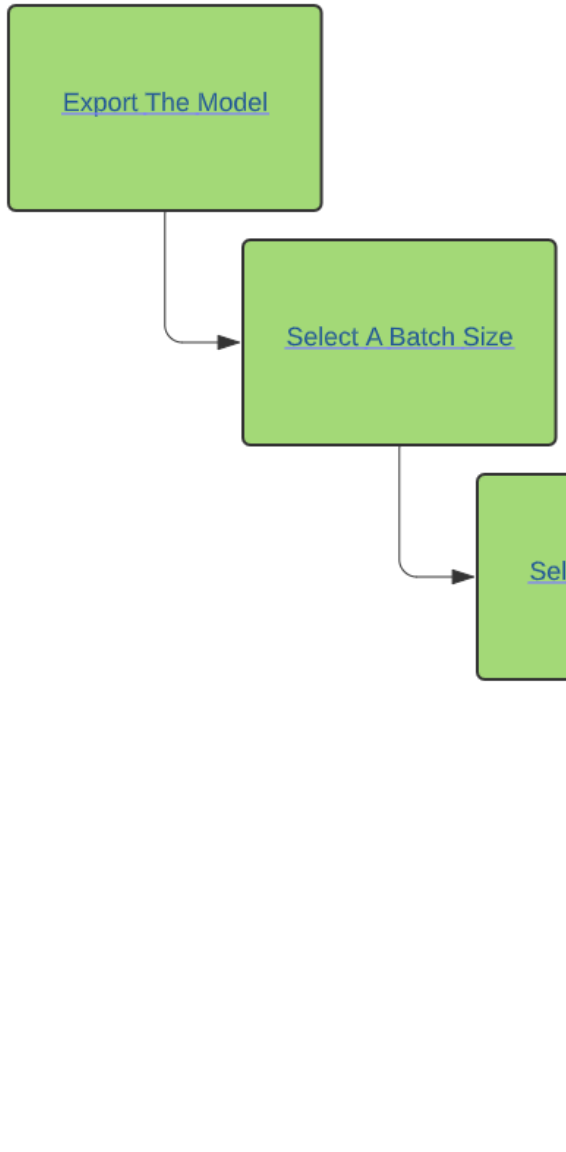
Network definition TensorRT中模型的表示。 网络定义是张量和运算符的图。

Engine 由TensorRT构建器优化的模型的表示。

Plan 序列化格式的优化后的推理引擎。 典型的应用程序将构建一次引擎，然后将其序列化为计划文件以供以后使用。要初始化推理引擎，应用程序将首先从plan文件中反序列化模型。

Runtime TensorRT的组件，可在TensorRT引擎上执行推理。

Basic TensorRT Workflow



推断时，当优先考虑延迟时选择小的batch，而优先考虑吞吐量时选择较大的batch。 较大的batch需要更长的处理时间，但可以减少每个sample的平均时间。

推理通常需要比训练少的数字精度。 较低的精度可以在不牺牲的精度的情况下提供更快的计算速度和更低的内存消耗。 TensorRT支持TF32，FP32，FP16和INT8精度。

将模型从ONNX（或WTS）转换为TensorRT引擎

两种类型的runtime：C++和Python绑定的独立的runtime，以及与TensorFlow的原生集成的runtime。

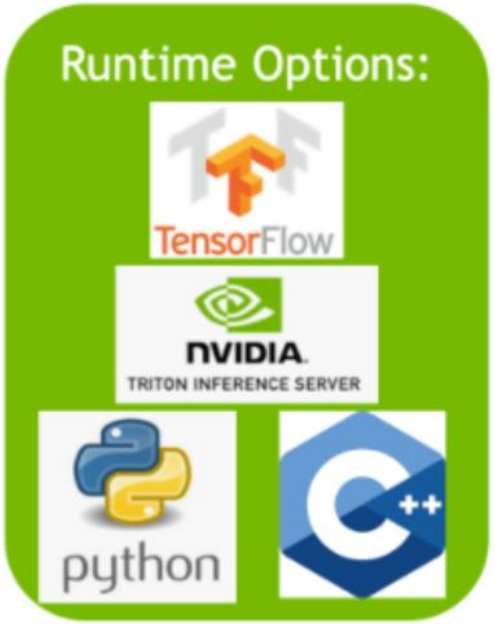
Conversion And Deployment Options

TensorRT生态系统分为两个部分：

1.用户可以遵循的各种路径将其模型转换为优化的TensorRT引擎。

2.部署优化的TensorRT引擎时，各种runtime用户可以使用TensorRT到不同的目标平台。

Main options available for conversion and deployment



Conversion

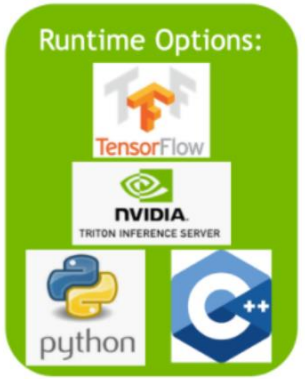
使用TensorRT转换模型有三个主要选择：

- using TF-TRT
- automatic ONNX conversion from .onnx files
- manually constructing a network using the TensorRT API (either in C++ or Python)

为了获得最佳性能和可定制性，还可以使用TensorRT网络定义API手动构建TensorRT引擎。 这涉及仅使用TensorRT操作按目标平台构建与原模型相同（或近似相同）的网络。

创建TensorRT网络后，可从框架中导出模型的权重，然后将其加载到TensorRT网络中。

Deployment



使用TensorRT部署模型有以下三种选择：

- deploying within TensorFlow
- using the standalone TensorRT runtime API
- using NVIDIA Triton Inference Server

TensorRT的runtimeAPI允许最低的开销和最细粒度的控制，但是TensorRT本身不支持的运算符必须实现为插件(plugin)。

TensorRT库将链接到部署应用程序，部署应用程序在需要推断结果时将调用该库。

要初始化推理引擎，应用程序将首先将模型从plan文件中反序列化为推理引擎。

TensorRT通常异步使用，因此，当输入数据到达时，程序将调用一个enqueue函数。

How Does TensorRT Work?

为了优化推理模型，TensorRT会采用你的网络定义，执行包括平台特定的优化，并生成推理引擎。 此过程称为**build阶段**。 build阶段可能会花费大量时间，尤其是在嵌入式平台上运行时。 因此，典型的应用程序将只构建一次引擎，然后将其序列化为plan文件以供以后使用。

构建阶段对图执行以下优化：

- 消除不使用其输出的层
- 消除等同于无操作的操作
- 卷积、偏置和ReLU操作的融合
- 使用完全相似的参数和相同的源张量（例如，GoogleNet v5的初始模块中的1x1卷积）进行的操作聚合(aggregation)
- 通过将层输出定向到正确的最终目的地来合并拼接层。

必要时，builder还可以修改权重的精度。 当生成8位整数精度的网络时，它使用称为calibration（校准）的过程来确定中间激活的动态范围，从而确定用于量化的适当缩放因子。

此外，build阶段还会在dummy数据上运行各层，以从其kernel目录中选择最快的文件，并在适当的情况下执行权重预格式化和内存优化。

使用TensorRT的必要步骤：

- 根据模型创建TensorRT的网络定义
- 调用TensorRT构建器以从网络创建优化的runtime引擎
- 序列化和反序列化引擎，以便可以在runtime快速重新创建它
- 向引擎提供数据以执行推理

Serializing and deserializing the TensorRT engine

Inference from network definition

Inference from Serialized Engine

Extending TensorRT With Custom Layers

用户可以使用C++和Python API的IPluginV2Ext类来实现自定义层，从而扩展TensorRT的功能。

自定义层（通常称为插件）由应用程序实现和实例化。