**Tensorrt**

# 1 官方总文档阅读

# 2 NVIDIA TensorRT Installation Guide

<https://docs.nvidia.com/deeplearning/tensorrt/install-guide/index.html>

主要参考上述文档 3.2.3 小节

# 3 NVIDIA TensorRT Quick Start Guide

[https://docs.nvidia.com/deeplearning/tensorrt/quick-start-guide/index.html#ecosystem](https://docs.nvidia.com/deeplearning/tensorrt/quick-start-guide/index.html" \l "ecosystem)

## 3.1 instruction

### 3.1.1 Tensorrt是啥能干啥

1. NVIDIA® TensorRT™ is an SDK for optimizing trained deep learning models to enable high-performance inference. TensorRT contains a deep learning inference optimizer for trained deep learning models, and a runtime for execution.
2. After you have trained your deep learning model in a framework of your choice, TensorRT enables you to run it with higher throughput and lower latency

### 3.1.2 本文档包含几部分

* 1. tensorRT install
  2. tensorRT Ecosystem
  3. Example Deployment Using ONNX
  4. TF-TRT Framework Integration
  5. ONNX Conversion and Deployment
  6. Using the TensorRT Runtime API

## 3.2 installing tensorrt

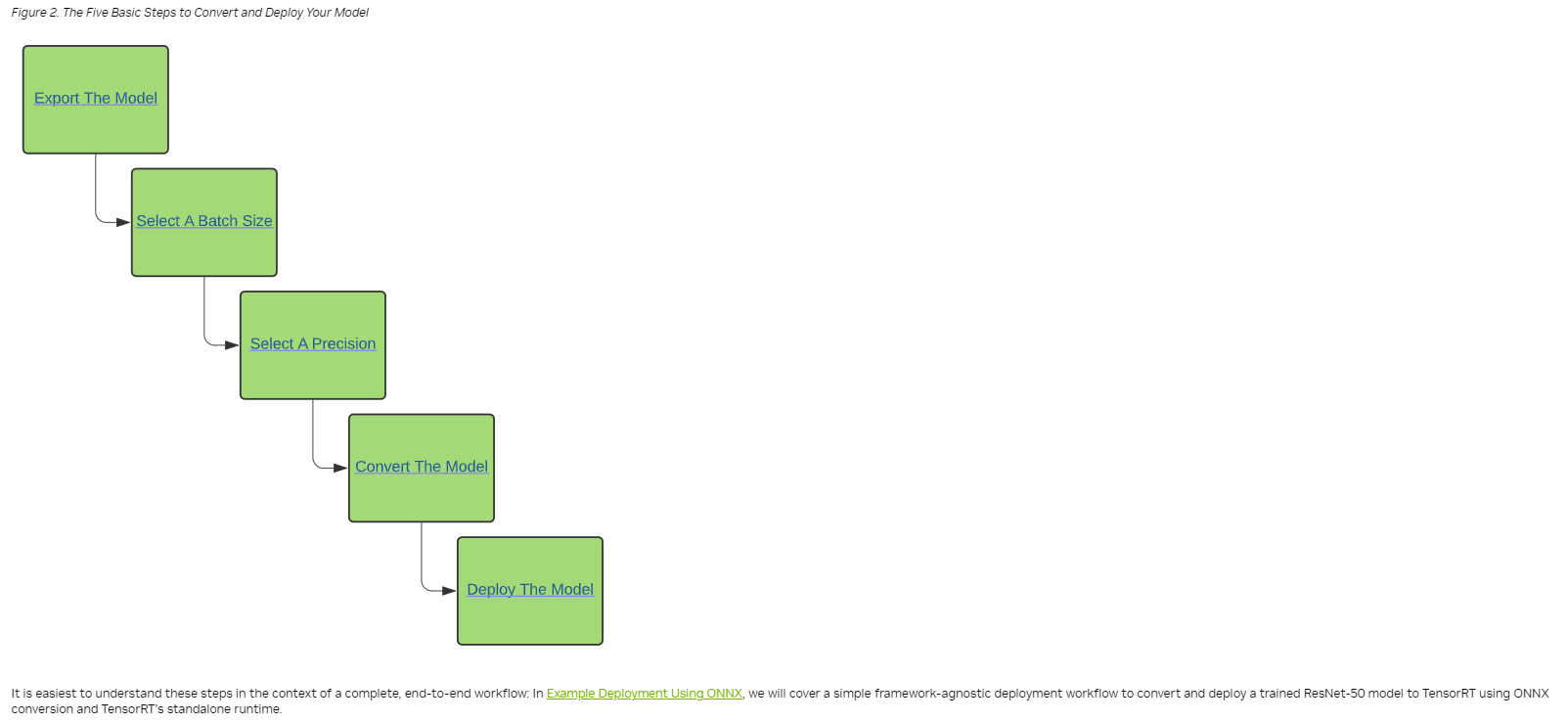
本部分直接以 2标题为主

## 3.3 tensorRT Ecosystem

Tensorrt提供了很多的开发选择，但是所有流程都需要包含一个模型转换（一种优化的表示），为自己的模型构建tensorRT workflow包含两部分内容：选择合适的部署选项 + 何使得引擎创建参数组合

TensorRT provides several options for deployment, but all workflows involve the conversion of your model to an optimized representation, which TensorRT refers to as an engine. Building a TensorRT workflow for your model involves picking the right deployment option, and the right combination of parameters for engine creation.

### 3.3.1 基本工作流 Basic TensorRT Workflow

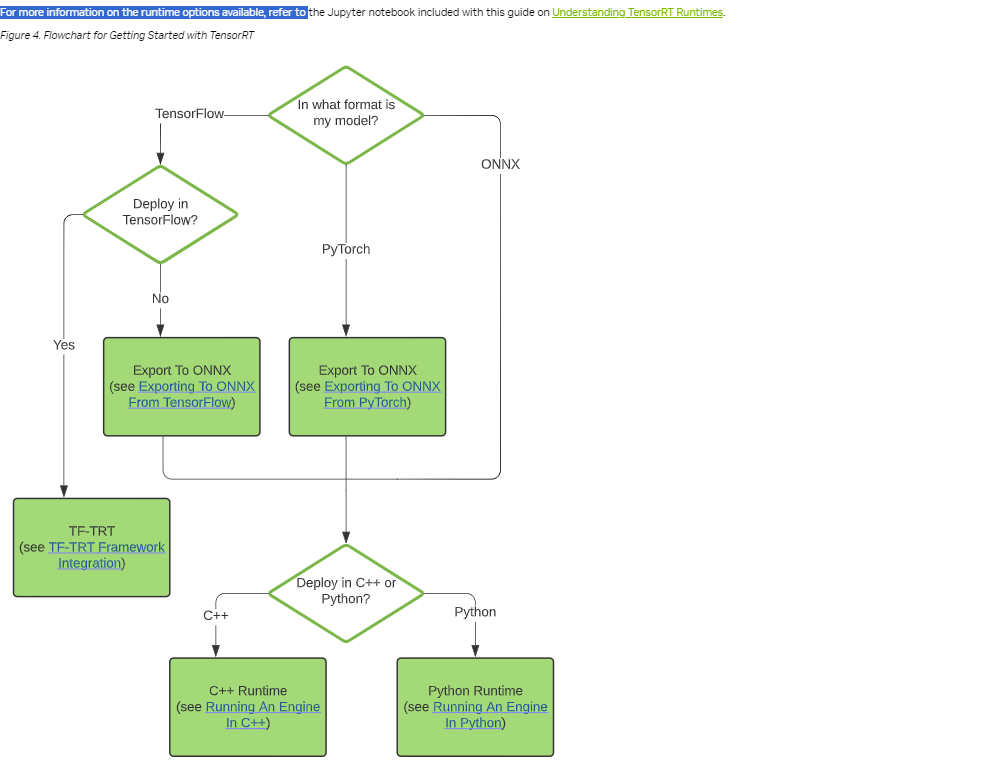


### 3.3.2 转化和开发选项Conversion and Deployment Options

1. Conversion（3种方式完成模型转换）
   1. using TF-TRT
   2. automatic ONNX conversion from .onnx files
      1. Onnx算子支持列表：[https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html#supported-ops](https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html" \l "supported-ops)
      2. manually constructing a network using the TensorRT API (either in C++ or Python)
2. Deployment（同样三种方式做模型的开发，对应章节有相应的开发文档介绍）
   * 1. deploying within TensorFlow
     2. using the standalone TensorRT runtime API
     3. using NVIDIA Triton Inference Server

### 3.3.3 选择合适的开发流 Selecting the Correct Workflow

1. 选择合适工作流的两个重要因素
   1. your choice of framework.
   2. your preferred TensorRT runtime to target.
      1. For more information on the runtime options available, refer to <https://github.com/NVIDIA/TensorRT/blob/main/quickstart/IntroNotebooks/5.%20Understanding%20TensorRT%20Runtimes.ipynb>
         1. 简单的列一下可选项
            1. TF-TRT
            2. TRITON Inference Server
            3. Python API
            4. C++ API



## 3.4 使用onnx的样例开发Example Deployment Using ONNX

### 3.4.1 模型导出

后续章节有专门的pytorch/tensorflow导出onnx的介绍

### **3.4.2 选择batch size**

1. 通用选择标准（延时 / 吞吐量）
   1. we pick a small batch size when we want to prioritize latency and a larger batch size when we want to prioritize throughput.
   2. a larger batch size when we want to prioritize throughput.
   3. Larger batches take longer to process but reduce the average time spent on each sample.
2. 动态batch
   1. 更多讯息请参考 [https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html#work\_dynamic\_shapes](https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html" \l "work_dynamic_shapes)

### 3.4.3 选择精度 Select a Precision

往往推理比训练所需的精度要求低，这就让我们有了精度选择的空间（trade off）

低精度为了争取：faster computation and lower memory consumption

### 3.4.4 使用tensorrt自带工具做onnx模型转换

We can run this conversion as follows:

trtexec --onnx=resnet50/model.onnx --saveEngine=resnet\_engine.trt

## 3.5 TF-TRT Framework Integration

暂时没必要看

## 3.6 转换样例ONNX Conversion and Deployment

这里有tensorflow和pythorch转onnx的案例代码

## 3.7 tensorRT runtime使用案例Using the TensorRT Runtime API

我主要关注C++的使用案例

### 3.7.1 更详细的说明了如何将onnx转成 tensorrt engine

Building an engine can be time-consuming, and is usually performed offline.

trtexec --onnx=fcn-resnet101.onnx --fp16 --workspace=64 --minShapes=input:1x3x256x256 --optShapes=input:1x3x1026x1282 --maxShapes=input:1x3x1440x2560 --buildOnly --saveEngine=fcn-resnet101.engine

kkkkkk