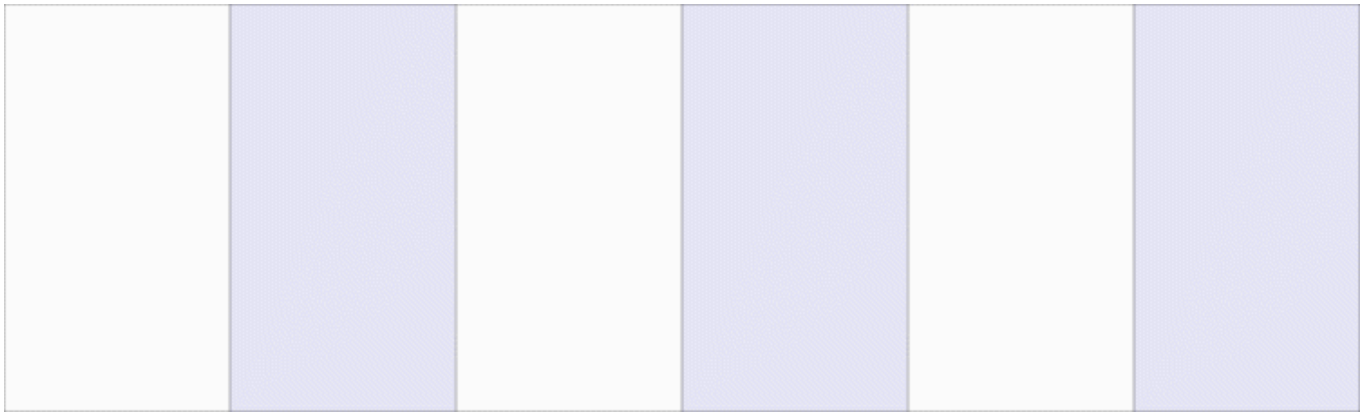
AI模型部署落地综述（ONNX/NCNN/TensorRT等）

极市平台 2023-01-14 22:00:20 发表于广东 手机阅读

以下文章来源于自动驾驶之心 ，作者逻辑牛



作者丨逻辑牛

来源丨自动驾驶之心

编辑丨极市平台

极市导读

模型部署指让训练好的模型在特定环境中运行的过程，是深度学习技术落地的重要流程，本文带你了解模型部署的方方面面，值得各位读者收藏阅读。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

导读

费尽心血训练好的深度学习模型如何给别人展示？只在服务器上运行demo怎么吸引别人的目光？怎么才能让自己的成果落地？这篇文章带你进入模型部署的大门。

0 前言

模型部署的步骤：

1. 训练一个深度学习模型；

2. 使用不同的推理框架对模型进行推理转换；

3. 在应用平台运行转换好的模型。

步骤看起来比较简单，但是牵扯的到的知识还是比较多。在实际应用过程中，我们使用的模型通常不会太简单，因为要确保模型的精度。但是，实际应用场景往往需要模型速度与精度能达到一个较好的平衡。因此这就需要在算法（剪枝，压缩等）与底层（手写加速算作）去优化模型。但是，我们现在可以站在巨人的肩膀上去眺望世界，因此，该文章会给大家介绍一些常用的开源推理框架，大家一起参考学习。毕竟大牛团队做出来的好用一些。

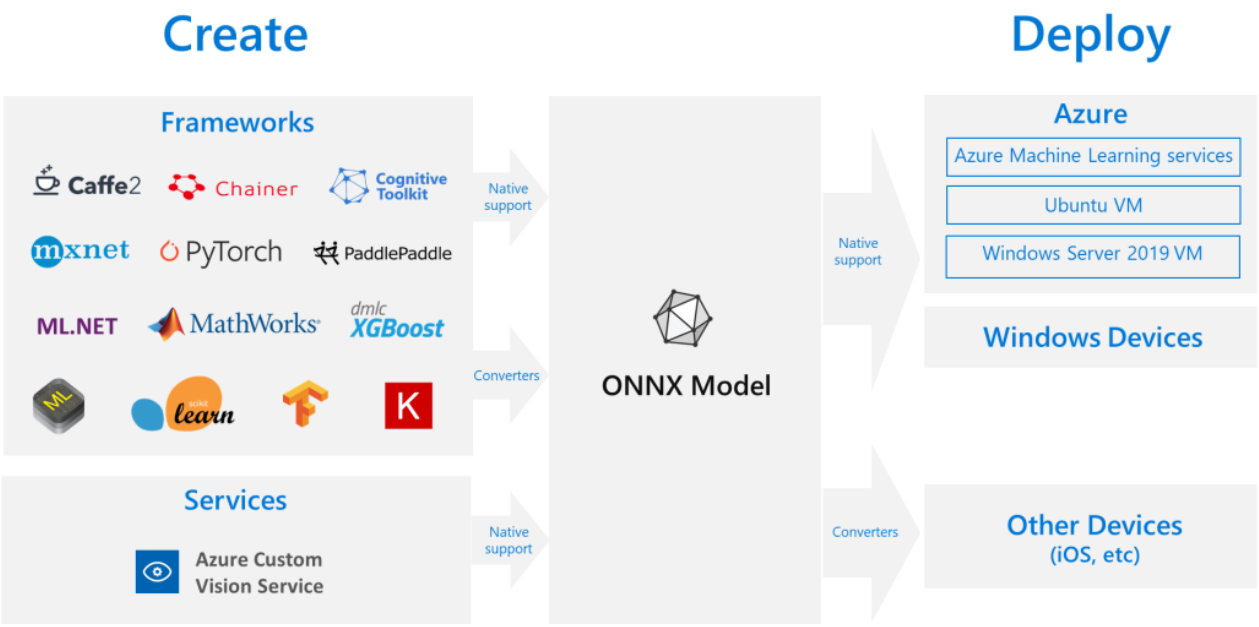
1 ONNX、NCNN、OpenVINO、 TensorRT、Mediapipe模型部署那家强?

1.1 ONNX

简介：

开放神经网络交换ONNX（Open Neural Network Exchange）是一套表示深度神经网络模型的开放格式，由微软和Facebook于2017推出，然后迅速得到了各大厂商和框架的支持。通过短短几年的发展，已经成为表示深度学习模型的实际标准，并且通过ONNX-ML，可以支持传统非神经网络机器学习模型，大有一统整个AI模型交换标准。ONNX定义了一组与环境和平台无关的标准格式，为AI模型的互操作性提供了基础，使AI模型可以在不同框架和环境下交互使用。硬件和软件厂商可以基于ONNX标准优化模型性能，让所有兼容ONNX标准的框架受益，简单来说，ONNX就是**模型转换的中间人。**

使用场景：



无论你使用什么样的训练框架来训练模型（比如TensorFlow/Pytorch/OneFlow/Paddle），你都可以在训练后将这些框架的模型统一转为ONNX存储。ONNX文件不仅存储了神经网络模型的权重，还存储了模型的结构信息、网络中各层的输入输出等一些信息。目前，ONNX主要关注在模型预测方面（inferring），将转换后的ONNX模型，转换成我们需要使用不同框架部署的类型，可以很容易的部署在兼容ONNX的运行环境中。

使用方法：

[代码示例]在 ONNX 模型上运行形状推理：https://github.com/onnx/onnx

```
import onnx
from onnx import helper, shape\_inference
from onnx import TensorProto

   \# 预处理：创建一个包含两个节点的模型，Y是未知的
node1 = helper.make\_node\("Transpose", \["X"\], \["Y"\], perm=\[1, 0, 2\]\)
node2 = helper.make\_node\("Trans

                          pose", \["Y"\], \["Z"\], perm=\[1, 0, 2\]\)

graph = helper.make\_graph\(
    \[node1, node2\],
    "two-transposes",
    \[helper.make\_tensor\_value\_info\("X", TensorProto.FLOAT, \(2, 3, 4\)\)\],
    \[helper.make\_tensor\_value\_info\("Z", TensorProto.FLOAT, \(2, 3, 4\)\)\],
\)
```

```
original\_model = helper.make\_model\(graph, producer\_name="onnx-examples"\)
    \# 检查模型并打印Y的信息
onnx.checker.check\_model\(original\_model\)
print\(f"Before shape inference, the shape info of Y is:\\n\{original\_model.graph.value\_
    \# 在模型上进行推理
inferred\_model = shape\_inference.infer\_shapes\(original\_model\)
    \# 检查模型并打印Y的信息
onnx.checker.check\_model\(inferred\_model\)
print\(f"After shape inference, the shape info of Y is:\\n\{inferred\_model.graph.value\_i
```

1.2 NCNN

简介：

ncnn 是一个为手机端极致优化的高性能神经网络前向计算框架，也是腾讯优图实验室成立以来的第一个开源项目。ncnn 从设计之初深刻考虑手机端的部署和使用，无第三方依赖，跨平台，手机端 CPU 的速度快于目前所有已知的开源框架。基于 ncnn，开发者能够将深度学习算法轻松移植到手机端高效执行，开发出人工智能 App。ncnn 目前已在腾讯多款应用中使用，如 QQ、Qzone、微信、天天P图等。

使用场景：

Current building status matrix

System	CPU (32bit)	CPU (64bit)	GPU (32bit)	GPU (64bit)
Linux (GCC)	build passing	build passing	—	build passing
Linux (Clang)	build passing	build passing	—	build passing
Linux (ARM)	build passing	build passing	—	—
Linux (MIPS)	build passing	build passing	—	—
Linux (RISC-V)	—	build passing	—	—
Linux (LoongArch)	—	build passing	—	—
Windows	build passing	build passing	—	build passing
Windows (ARM)	build passing	build passing	—	—
macOS	—	build passing	—	build passing
macOS (ARM)	—	build passing	—	build passing
Android	build passing	build passing	build passing	build passing
Android-x86	build passing	build passing	build passing	build passing
iOS	build passing	build passing	—	build passing
iOS Simulator	build passing	build passing	—	—
WebAssembly	build passing	—	—	—
RISC-V GCC/Newlib	build passing	build passing	—	—

从NCNN的发展矩阵可以看出，NCNN覆盖了几乎所有常用的系统平台，尤其是在移动平台上的适用性更好，在Linux、Windows和Android、以及iOS、macOS平台上都可以使用GPU来部署模型。

框架特点：

- 支持卷积神经网络，支持多输入和多分支结构，可计算部分分支
- 无任何第三方库依赖，不依赖 BLAS/NNPACK 等计算框架
- 纯 C++ 实现，跨平台，支持 Android / iOS 等

- ARM Neon 汇编级良心优化，计算速度极快
- 精细的内存管理和数据结构设计，内存占用极低
- 支持多核并行计算加速，ARM big.LITTLE CPU 调度优化
- 支持基于全新低消耗的 Vulkan API GPU 加速
- 可扩展的模型设计，支持 8bit 量化和半精度浮点存储，可导入 caffe/pytorch/mxnet/onnx/darknet/keras/tensorflow(mlir) 模型
- 支持直接内存零拷贝引用加载网络模型
- 可注册自定义层实现并扩展

使用方法：

[代码示例]输入数据并推理输出：https://github.com/Tencent/ncnn/wiki

```
\#include \<opencv2/core/core.hpp>   \#include \<opencv2/highgui/highgui.hpp>   \#include

int main\(\)
\{
 // opencv读取输入图片
    cv::Mat img = cv::imread\("image.ppm", CV\_LOAD\_IMAGE\_GRAYSCALE\);
    int w = img.cols;
    int h = img.rows;

    // 减均值以及缩放操作，最后输入数据的值域为\[-1,1\]
    ncnn::Mat in = ncnn::Mat::from\_pixels\_resize\(img.data, ncnn::Mat::PIXEL\_GRAY, w, h
    float mean\[1\] = \{ 128.f \};
    float norm\[1\] = \{ 1/128.f \};
    in.substract\_mean\_normalize\(mean, norm\);

 // 构建NCNN的net，并加载转换好的模型
    ncnn::Net net;
    net.load\_param\("model.param"\);
    net.load\_model\("model.bin"\);

 // 创建网络提取器，设置网络输入，线程数，light模式等等
    ncnn::Extractor ex = net.create\_extractor\(\);
    ex.set\_light\_mode\(true\);
    ex.set\_num\_threads\(4\);
    ex.input\("data", in\);
 // 调用extract接口，完成网络推理，获得输出结果
    ncnn::Mat feat;
    ex.extract\("output", feat\);
```

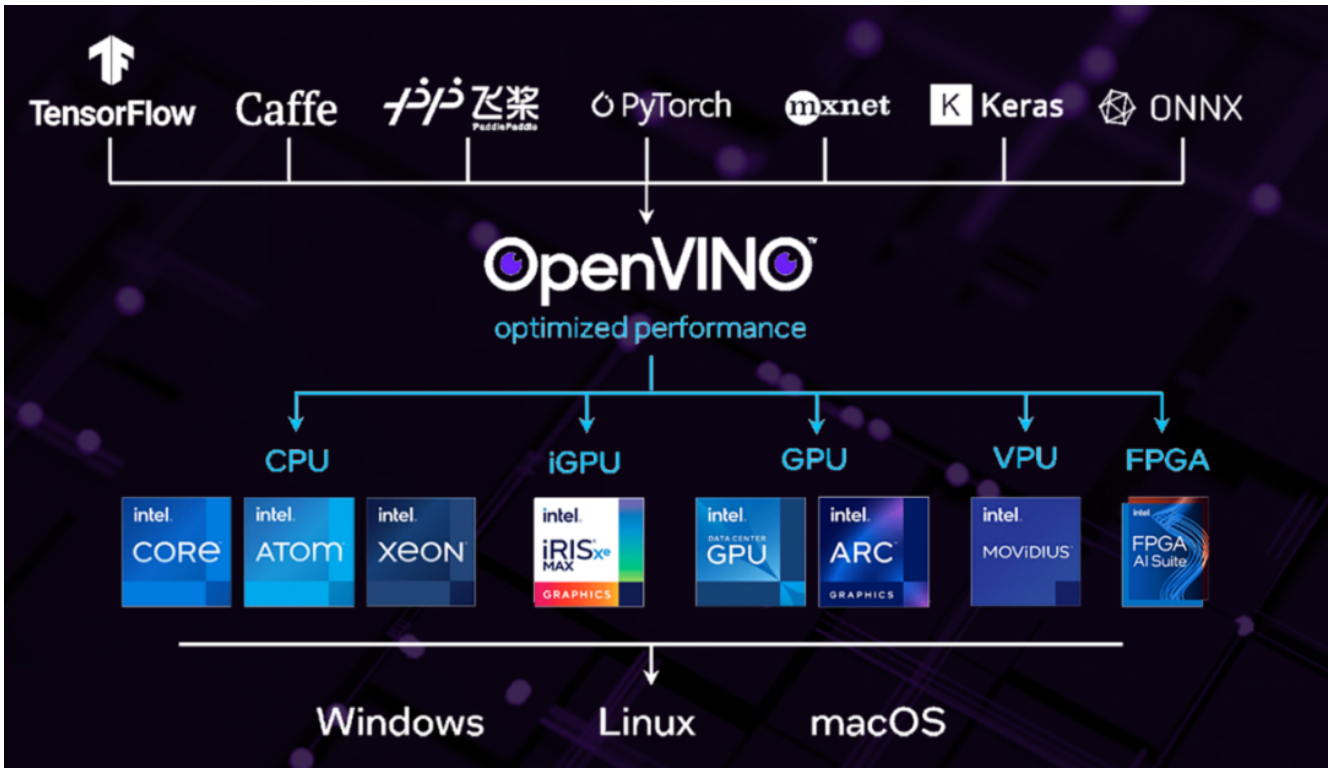
```
    return 0;
```

1.3 OpenVINO

简介：

OpenVINO是一种可以加快高性能计算机视觉和深度学习视觉应用开发速度的工具套件，支持各种英特尔平台的硬件加速器上进行深度学习，并且允许直接异构执行。OpenVINO™工具包是用于快速开发应用程序和解决方案的综合工具包，可解决各种任务，包括模拟人类视觉，自动语音识别，自然语言处理，推荐系统等。该工具包基于最新一代的人工神经网络，包括卷积神经网络（CNN），循环和基于注意力的网络，可在英特尔®硬件上扩展计算机视觉和非视觉工作负载，从而最大限度地提高性能。它通过从边缘到云的高性能，人工智能和深度学习推理来加速应用程序。

使用场景：



框架特点：

OpenVINO在模型部署前，首先会对模型进行优化，模型优化器会对模型的拓扑结构进行优化，去掉不需要的层，对相同的运算进行融合、合并以加快运算效率，减少内存拷贝；FP16、INT8量化也可以在保证精度损失很小的前提下减小模型体积，提高模型的性能。在部署方面，OpenVIVO的开发也是相对比较简单的，提供了C、C++和python3种语言编程接口。它最大的优

势呢，其实还是在Intel的不同硬件平台上进行部署的时候，移植会很方便。推理引擎对不同的硬件提供统一的接口，底层实现直接调用硬件指令集的加速库，应用程序开发人员不需要关心底层的硬件实现，即可在不同的硬件平台上加速模型推理。

- 在边缘启用基于CNN的深度学习推理
- 支持通过英特尔®Movidius™VPU在英特尔®CPU，英特尔®集成显卡，英特尔®神经计算棒2和英特尔®视觉加速器设计之间进行异构执行
- 通过易于使用的计算机视觉功能库和预先优化的内核加快上市时间
- 包括对计算机视觉标准（包括OpenCV *和OpenCL™）的优化调用

使用方法：

[代码示例]在应用程序中实现典型的 OpenVINO™ 运行推理：https://docs.openvino.ai/latest/openvino_docs_OV_UG_Integrate_OV_with_your_application.html

```
 \#include \<openvino/openvino.hpp>
// 1.创建 OpenVINO™ 核心以管理可用设备和读取模型对象
ov::Core core;
// 2.为特定设备编译模型
ov::CompiledModel compiled\_model = core.compile\_model\("model.onnx", "AUTO"\);
// 3.创建推理请求
ov::InferRequest infer\_request = compiled\_model.create\_infer\_request\(\);
// 4.设置输入
// 获取模型的输入端口
auto input\_port = compiled\_model.input\(\);
// 从外部存储器创建张量
ov::Tensor input\_tensor\(input\_port.get\_element\_type\(\), input\_port.get\_shape\(\),
// 为模型设置一个输入张量
infer\_request.set\_input\_tensor\(input\_tensor\);
// 5.开始推理
infer\_request.start\_async\(\);
infer\_request.wait\(\);
// 6.处理推理结果
// 通过tensor\_name获取输出张量
auto output = infer\_request.get\_tensor\("tensor\_name"\);
const float \\\*output\_buffer = output.data\<const float\>\(\);
// output\_buffer\[\] - 访问输出张量数据
// 7.释放分配的对象（仅适用于C）
ov\_shape\_free\(\&input\_shape\);
ov\_tensor\_free\(output\_tensor\);
ov\_output\_const\_port\_free\(input\_port\);
ov\_tensor\_free\(tensor\);
ov\_infer\_request\_free\(infer\_request\);
ov\_compiled\_model\_free\(compiled\_model\);
```

```
ov\_model\_free\(model\);
ov\_core\_free\(core\);


// 为项目创建结构
project/
   ├── CMakeLists.txt  - CMake file to build
   ├── ...             - Additional folders like includes/
   └── src/            - source folder
       └── main.cpp
build/                  - build directory
   ...


// 创建 Cmake 脚本
cmake\_minimum\_required\(VERSION 3.10\)
set\(CMAKE\_CXX\_STANDARD 11\)

find\_package\(OpenVINO REQUIRED\)

add\_executable\(\$\{TARGET\_NAME\} src/main.cpp\)

target\_link\_libraries\(\$\{TARGET\_NAME\} PRIVATE openvino::runtime\)


// 构建项目
cd build/
cmake ../project
cmake --build .
```

1.4 TensorRT

简介：

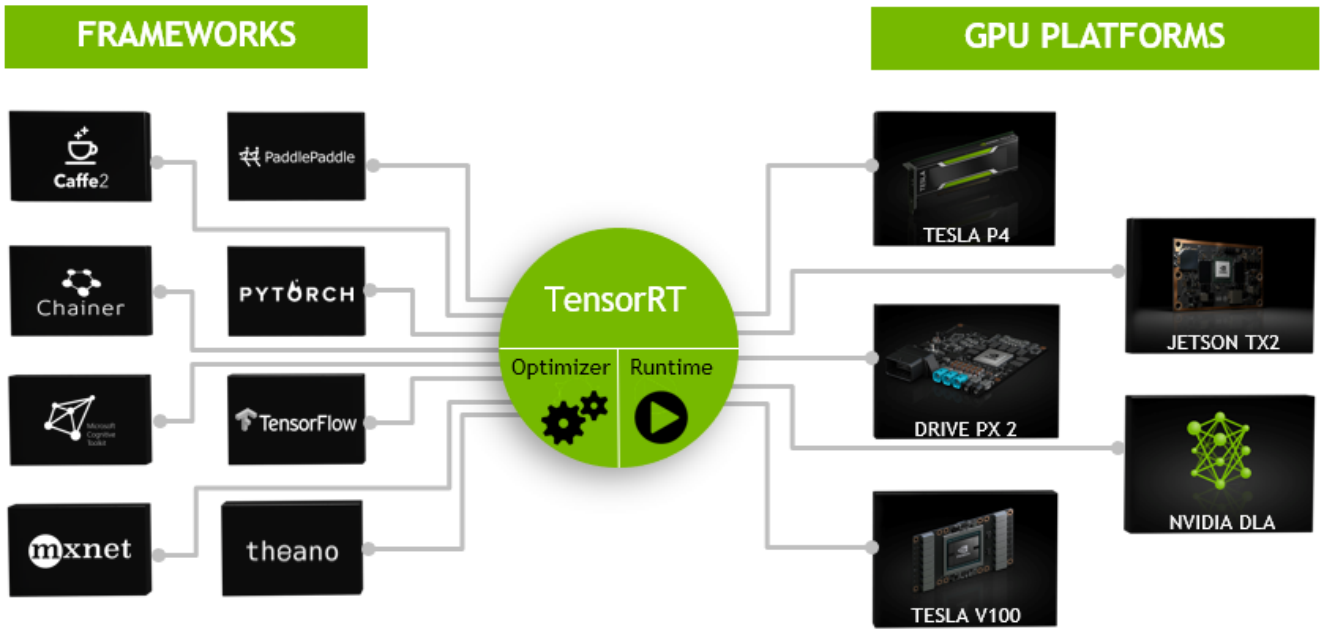
NVIDIA TensorRT™ 是用于高性能深度学习推理的 SDK。此 SDK 包含深度学习推理优化器和运行时环境，可为深度学习推理应用提供低延迟和高吞吐量。

在推理过程中，基于 TensorRT 的应用程序的执行速度可比 CPU 平台的速度快 40 倍。借助 TensorRT，您可以优化在所有主要框架中训练的神经网络模型，精确校正低精度，并最终将模型部署到超大规模数据中心、嵌入式或汽车产品平台中。

TensorRT 以 NVIDIA 的并行编程模型 CUDA 为基础构建而成，可帮助您利用 CUDA-X 中的库、开发工具和技术，针对人工智能、自主机器、高性能计算和图形优化所有深度学习框架中的推理。

TensorRT 针对多种深度学习推理应用的生产部署提供 INT8 和 FP16 优化，例如视频流式传输、语音识别、推荐和自然语言处理。推理精度降低后可显著减少应用延迟，这恰巧满足了许多实时服务、自动和嵌入式应用的要求。

使用场景：



框架特点：

1. 权重与激活精度校准

通过将模型量化为 INT8 来更大限度地提高吞吐量，同时保持高准确度

2. 层与张量融合

通过融合内核中的节点，优化 GPU 显存和带宽的使用

3. 内核自动调整

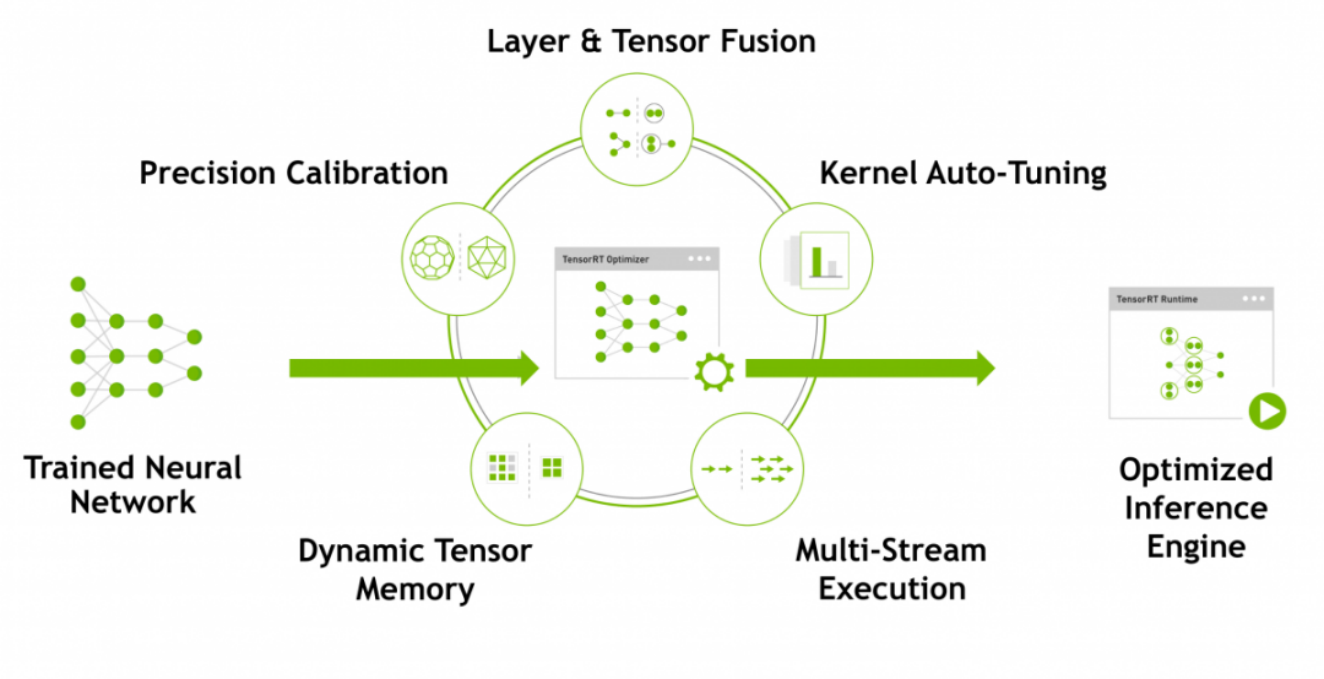
基于目标 GPU 平台选择最佳数据层和算法

4. 动态张量显存

更大限度减少显存占用，并高效地为张量重复利用内存

5. 多流执行

用于并行处理多个输入流的可扩展设计



图片取自TensorRT的官网，里面列出了TensorRT使用的一些技术。可以看到模型量化、动态内存优化、层的融合等技术均已经在TensorRT中集成了，这也是它能够极大提高模型推断速度的原因。总体来说TensorRT将训练好的模型通过一系列的优化技术转化为了能够在特定平台（GPU）上以高性能运行的代码，也就是最后图中生成的Inference Engine。

使用方法：

1.导出模型

2.选择批次大小

3.选择精度

4.转换模型：

- 使用 TF-TRT
- 从文件自动转换 ONNX
- 使用 TensorRT API 手动构建网络（C++或python）

5.部署模型：

- 在 TensorFlow 中部署
- 使用独立的 TensorRT 运行时 API
- 使用 NVIDIA Triton 推理服务器

具体模型转换部署方法详见：[Quick Start Guide :: NVIDIA Deep Learning TensorRT Documentation]： https://docs.nvidia.com/deeplearning/tensorrt/quick-start-guide/index.html

1.5 Mediapipe

简介：

MediaPipe是一款由 Google Research 开发并开源的多媒体机器学习模型应用框架。在谷歌，一系列重要产品，如 YouTube、Google Lens、ARCore、Google Home 以及 Nest，都已深度整合了 MediaPipe。作为一款跨平台框架，MediaPipe 不仅可以被部署在服务器端，更可以在多个移动端 （安卓和苹果 iOS）和嵌入式平台（Google Coral 和树莓派）中作为设备端机器学习推理 （On-device Machine Learning Inference）框架。

除了上述的特性，MediaPipe 还支持 TensorFlow 和 TF Lite 的推理引擎（Inference Engine），任何 TensorFlow 和 TF Lite 的模型都可以在 MediaPipe 上使用。同时，在移动端和嵌入式平台，MediaPipe 也支持设备本身的 GPU 加速。

使用场景：

	Android	iOS	C++	Python	JS	Coral
Face Detection	✅	✅	✅	✅	✅	✅
Face Mesh	✅	✅	✅	✅	✅	
Iris	✅	✅	✅			
Hands	✅	✅	✅	✅	✅	
Pose	✅	✅	✅	✅	✅	
Holistic	✅	✅	✅	✅	✅	
Selfie Segmentation	✅	✅	✅	✅	✅	
Hair Segmentation	✅		✅			
Object Detection	✅	✅	✅			✅
Box Tracking	✅	✅	✅			
Instant Motion Tracking	✅					
Objectron	✅		✅	✅	✅	
KNIFT	✅					
AutoFlip			✅			
MediaSequence			✅			
YouTube 8M			✅			

框架特点：

1. 端到端加速：内置快速 ML 推理和处理，即使在普通硬件上也能加速
2. 一次构建，随处部署：统一解决方案适用于安卓、iOS、桌面/云、Web 和物联网
3. 即用型解决方案：展示框架全部功能的尖端 ML 解决方案
4. 免费和开源：Apache 2.0下的框架和解决方案，完全可扩展和可定制

使用方法：

[**代码示例**]以人脸检测为例：https://google.github.io/mediapipe/solutions/face_detection

```
import cv2
import mediapipe as mp
mp\_face\_detection = mp.solutions.face\_detection
mp\_drawing = mp.solutions.drawing\_utils
   \# 对于静态图像:
IMAGE\_FILES = \[\]
with mp\_face\_detection.FaceDetection\(
    model\_selection=1, min\_detection\_confidence=0.5\) as face\_detection:
  for idx, file in enumerate\(IMAGE\_FILES\):
    image = cv2.imread\(file\)
    \# 将BGR图像转换为RGB并使用MediaPipe人脸检测对其进行处理.
    results = face\_detection.process\(cv2.cvtColor\(image, cv2.COLOR\_BGR2RGB\)\)

    \# 绘制每张人脸的人脸检测.
    if not results.detections:
      continue
    annotated\_image = image.copy\(\)
    for detection in results.detections:
      print\('Nose tip:'\)
      print\(mp\_face\_detection.get\_key\_point\(
          detection, mp\_face\_detection.FaceKeyPoint.NOSE\_TIP\)\)
      mp\_drawing.draw\_detection\(annotated\_image, detection\)
    cv2.imwrite\('/tmp/annotated\_image' + str\(idx\) + '.png', annotated\_image\)
   \# 用于网络摄像头输入:
cap = cv2.VideoCapture\(0\)
with mp\_face\_detection.FaceDetection\(
    model\_selection=0, min\_detection\_confidence=0.5\) as face\_detection:
  while cap.isOpened\(\):
    success, image = cap.read\(\)
    if not success:
      print\("Ignoring empty camera frame."\)
      \# 如果加载视频，请使用“中断”而不是“继续”.
      continue

    \# 若要提高性能，可以选择将图像标记为不可写以通过引用传递.
    image.flags.writeable = False
    image = cv2.cvtColor\(image, cv2.COLOR\_BGR2RGB\)
    results = face\_detection.process\(image\)

    \# 在图像上绘制人脸检测注释.
    image.flags.writeable = True
    image = cv2.cvtColor\(image, cv2.COLOR\_RGB2BGR\)
    if results.detections:
      for detection in results.detections:
        mp\_drawing.draw\_detection\(image, detection\)
    \# 水平翻转图像以获得自拍视图显示.
```

```
    cv2.imshow\('MediaPipe Face Detection', cv2.flip\(image, 1\)\)
    if cv2.waitKey\(5\) \& 0xFF == 27:
      break
cap.release\(\)
```

2.框架对比

应用平台：

模型推理部署框架	应用平台
NCNN	移动端
OpenVINO	CPU，GPU，嵌入式平台都可以使用，尤其是在CPU上首选OPenVINO。DepthAI嵌入式空间AI平台。
TensorRT	只能用在NIVDIA的GPU上的推理框架。NIVDIA的Jetson平台。
Mediapipe	服务端，移动端，嵌入式平台，TPU。

研发单位：

- 腾讯公司开发的移动端平台部署工具——NCNN；
- Intel公司针对自家设备开开发的部署工具——OpenVINO；
- NVIDIA公司针对自家GPU开发的部署工具——TensorRT；
- Google针对自家硬件设备和深度学习框架开发的部署工具——Mediapipe；
- 由微软、亚马逊 、Facebook 和 IBM 等公司共同开发的开放神经网络交换格式——ONNX；

如何选择：

- ONNXRuntime 是可以运行在多平台 (Windows，Linux，Mac，Android，iOS) 上的一款推理框架，它接受 ONNX 格式的模型输入，支持 GPU 和 CPU 的推理。唯一不足就是 ONNX 节点粒度较细，推理速度有时候比其他推理框架如 TensorRT 较低。
- NCNN是针对手机端的部署。优势是开源较早，有非常稳定的社区，开源影响力也较高。
- OpenVINO 是 Intel 家出的针对 Intel 出品的 CPU 和 GPU 友好的一款推理框架，同时它也是对接不同训练框架如 TensorFlow，Pytorch，Caffe 等。不足之处可能是只

支持 Intel 家的硬件产品。

- TensorRT 针对 NVIDIA 系列显卡具有其他框架都不具备的优势，如果运行在 NVIDIA 显卡上， TensorRT 一般是所有框架中推理最快的。一般的主流的训练框架如TensorFlow 和 Pytorch 都能转换成 TensorRT 可运行的模型。当然了，TensorRT 的限制就是只能运行在 NVIDIA 显卡上，同时不开源 kernel。
- MediaPipe 不支持除了tensorflow之外的其他深度学习框架。MediaPipe 的主要用例是使用推理模型和其他可重用组件对应用机器学习管道进行快速原型设计。MediaPipe还有助于将机器学习技术部署到各种不同硬件平台上的演示和应用程序中，为移动、桌面/云、web和物联网设备构建世界级ML解决方案和应用程序。

3.小结

本文主要介绍了5种推理框架，目的是使大家更加直观的了解这几种框架的特点，应用场景以及如何选择，为大家之后的学习提供有限的帮助，不足之处请大家多多指正。

参考资料

1. https://learn.microsoft.com/zh-cn/windows/ai/
2. https://github.com/Tencent/ncnn
3. https://zhuanlan.zhihu.com/p/344442534
4. https://github.com/google/mediapipe
5. https://www.zhihu.com/question/346965029/answer/2395418101

实操教程：GPU多卡并行训练总结（以pytorch为例）｜CUDA WarpReduce 学习笔记｜卷积神经网络压缩方法总结

阅读原文

ICCV 2023 | Pixel-based MIM: 简单高效的多级特征融合自监督方法

极市平台