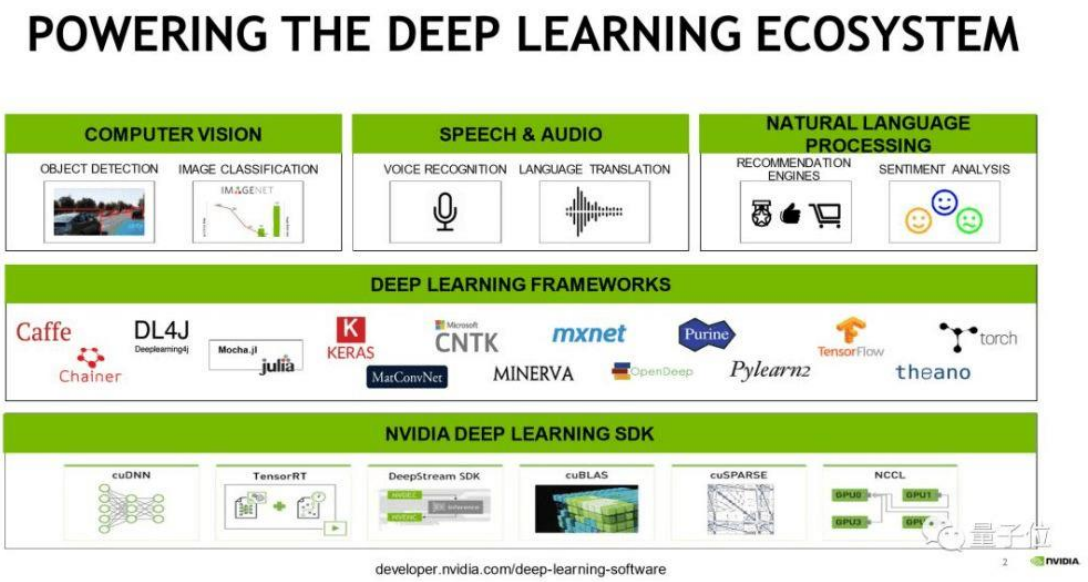
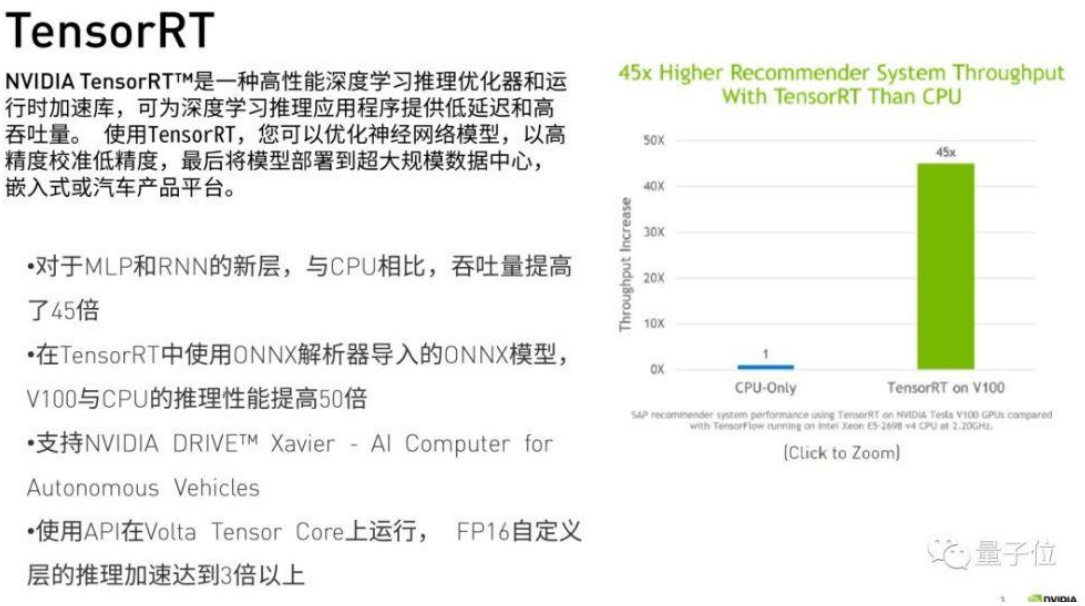
12 月 22 日，量子位联合 NVIDIA 英伟达举行了线下交流会，跟现场近百位开发者同学共同探讨了如何用 TensorRT 加速深度学习推理计算，详细讲解了 GPU 的推理引擎 TensorRT，以及如何配合 Deepstream 实现完整加速推理的解决方案。



这张图是一个完整的现代深度学习应用，在硬件的应用方面，三个比较大的常用的领域是：视觉相关，包含医疗、自动驾驶、摄像头的处理等等；另外是语音和自然语言处理相关，包含语义理解和语音识别。
为了实现这些功能，会使用各种各样的框架，包括 Caffe、TensorFlow、PyTorch 等等。
框架不同，每个人的使用习惯也不同，但最终都是完成一件事情：深度学习模型的训练，以及深度模型的推理。这样一个工具。完成这两个任务最核心的内容还是计算。
今天主要核心介绍的就是 TensorRT 以及相关的 DeepStream SDK，他们如何来加速深度学习推理计算。



1、先介绍一下什么是 TensorRT。

Tensor，深度学习中的张量。一个一维 Tensor 的叫 Vector，二维的叫 Matrix，三维四维的 Tensor 都有。Tensor RT 的 Tensor 指的就是这个概念，而整个 TensorRT 的工具是指一个 GPU Inference Engine，也就是一个 GPU 推理引擎。

GPU 推理引擎是干什么的呢？我们在做推理的时候，比如拿 Caffe 框架训练出来一个模型，那么推理的时候就直接拿 Caffe 框架 Test。

但是上述场景只适用于做研究或者做课题，部署到产品端会更复杂。举个例子，比如部署到自动驾驶汽车上，框架都很常见，但这么大的框架塞到一个嵌入式的移动设备中非常不合适，里边的空间、计算的内存、CPU 的一些计算资源，不可能还有其余的空间给这些框架。

自动驾驶汽车上要求提高反应速度，在实验室里的话，一秒反应 1 帧、5 帧甚至 10 帧都没问题，但是在实际的应用中，需要反应很快，这时候就面临一个计算资源有限的问题，在计算资源有限的情况下，还要得到一个更好的加速效果。

硬件设备是没法调整的，那么就要从调整算法入手。TensorRT 就会在前端的 Inference 平台上，包括在服务器的一些显卡上，专门针对性地帮助推理。

2、加速推理的原理
TensorRT 加速推理的原理是什么呢？TensorRT 基本上支持现在所有的主流框架训练出来的模型，把模型输入进来，然后 TensorRT 帮助做解析，做网络层之间的优化。

另外是 TensorRT 的输入输出，TensorRT 的输入是一个已经训练好的 32 位的数据类型 FP32 的网络模型，而它的输出是可执行的一个推理引擎。这个推理引擎指的是输出这个推理引擎之后，可以直接在代码里被调用。

此外 TensorRT 还有一个很方便的功能：在已经建立好引擎之后，把它序列化出来，下次再重新调用这个引擎的时候，直接把序列化的文件反序列化，就可以使用了，不用再从头开始创建。

3、性能
这是 TensorRT 4.0 之后的一个更新，RNN 对 NLP 这块的支持变得越来越好。
除了 TensorFlow 和 Caffe 这两个训练出来的模型以外，基本上现在都支持 Onnx 的网络格式。Onnx 网络格式是一个通用的标准，通过 Onnx 的支持，不同的框架都可以转化到这个模型，相互之间就变得更加协调了。

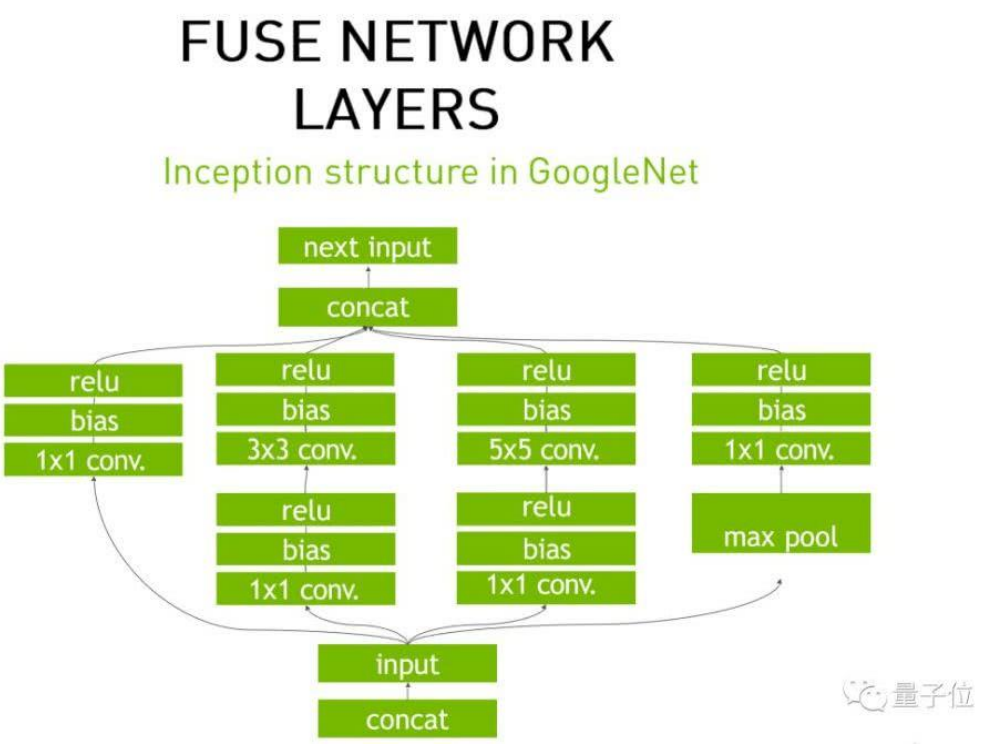
TensorRT 在里面做的主要是这五件事：

第一件事是权重精度的校正。比如现在通常用的是 FP32 的数据类型，如果在做 HPC 的时候，就可能都会用 double 类型的 float，我们为什么要 double 类型？因为精度需要非常大的动态范围。
但是深度学习在做训练的时候可能并不需要那么大的动态范围，它非常小，可能连 FP32 都用不了，这个时候就可以用一个更小的动态数据类型。

现在 TensorRT 提出一个 FP16 半精度的数据类型，还有 int8 以及最新出的 int4 的一些数据类型。用这个

工具可以帮助把一个较大范围的数据映射到一个较小的范围之内。这样的话在做计算的时候，转换数据的速度非常快，因为数据类型的原因，它所占用的资源也非常少，这个时候的计算速度就会变得很快。
第二件事是网络层的合并，包括一些张量的重新规划。
第三件事是内核调用。内核调用跟 GPU 底层相关，NVIDIA 所有的工具基本上都是基于 CUDA 这个生态所建立，CUDA 里核心的内容是两个方面。

一个方面是内存中各种显存的调用，多流的执行。并行计算最有一个难点是数据传输之间那部分很难消掉，硬件卡住的部分无法并行，而 CUDA 里边多流的执行把这些数据在传输的过程中进行计算，这样就把传输的部分给隐藏了。
另外一个方面是怎么调用 CUDA 核，怎么分配，每个 block 里边分配多少个线程，每个 grid 里边有多少个 block。这些内容都对 GPU 要求或者对 CUDA 的计算功底要求比较高，需要花更多时间，也需要对硬件底层更了解。而 TensorRT 里边调用了一些方法，以一个最合理的方式去调用、操作这些数据。



举一个网络融合的例子，GoogleNet 的 Inception 结构，很多网络模型也都借鉴这个格式。

数据出来之后，经过这么多步骤，TensorRT 怎么进行网络结构优化？数据出来之后，它要经过一个卷积层，一个 bias，一个 Relu，也就是三个网络层，而在底层就是在 CUDA 里边实现的。

数据到卷积层的时候，会把数据放到 GPU 里边去计算，然后再翻回来。我们的框架是这样

处理的，因为框架是需要所有的硬件平台都能够实现这功能，而 TensorRT 是针对 GPU 进行优化的，它是专门针对于前端推理的优化。

推理这样一个操作的优化的时候，TensorRT 把三个网络层结合到一起了，数据出来后分析网络结构。针对 Inference，把三个网络层结合到一起，数据出来后直接计算，然后返还给 CPU。这样就相当于减少了四次访问次数。

现在的网络都变得越来越宽，所以网络层也越来越多，那么减少数据传输的时间就会变得很重要。这也符合 CUDA 的核心思想：把零碎的数据整理到一起去传输，减传输次数。

三个网络层是一样的，可以放在一起。然后再往下，Concat 也不用了，因为 TensorRT 会自动帮你把它拼接好。

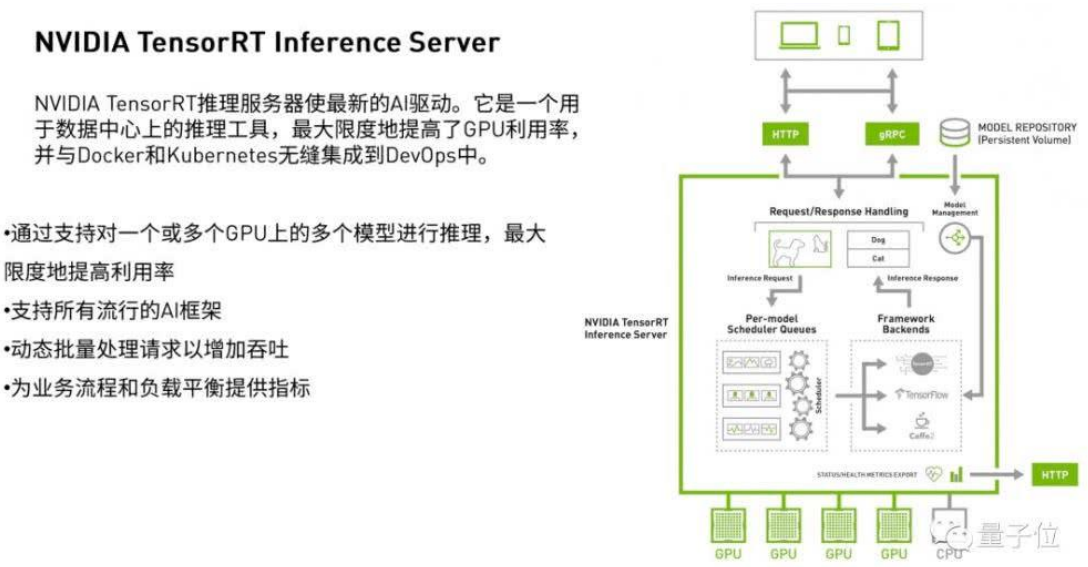
那么接下来怎样继续优化？到现在还能怎么去优化？之前提到过多流执行的概念。假设三个计算的时间是三秒，两个计算出来是两秒，那么如果我们把它分在两个流里边去执行，就可以节省两秒。

这些网络层现在是越来越深，网络层的个数越来越多，那么把这些融合到一起，减少读取数据访存的次数。于是就又把横向的时间减少了。

这就是 TensorRT 在这两个方向上优化的底层的原理。

TensorRT 现在能支持的网络层有很多，基本上所有的网络模型都是支持的。除了这些之外，还有一个重点，就是对不同的用户，对网络层的设计是不一样的，并不是通用的。

4、TensorRT Inference Server

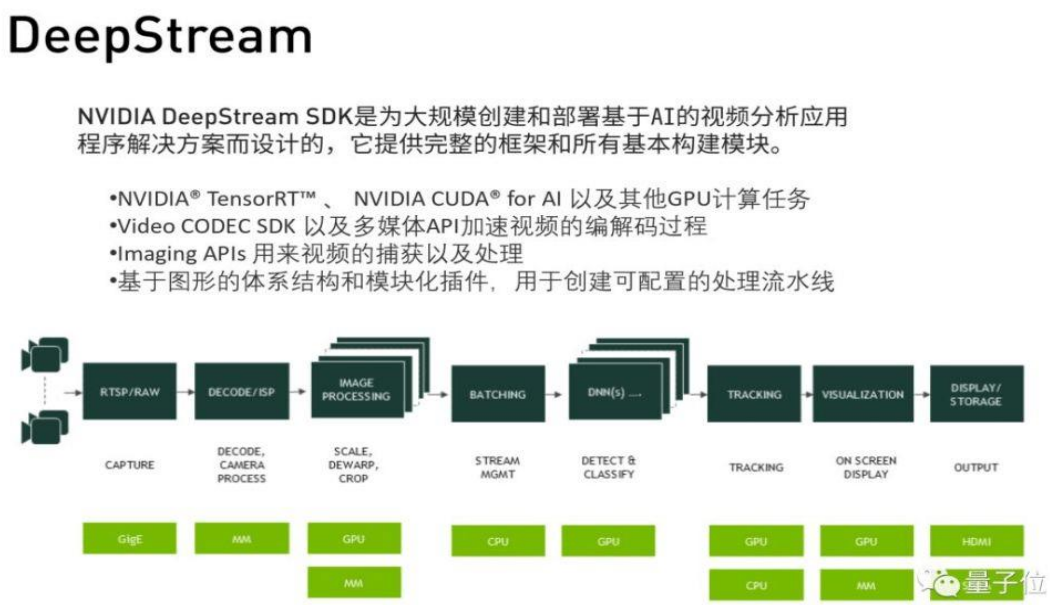


刚才介绍了 TensorRT 是一个加速工具，用 TensorRT 并不是把它放在一个课题实验里，核心目的是把 TensorRT 部署到产品上。

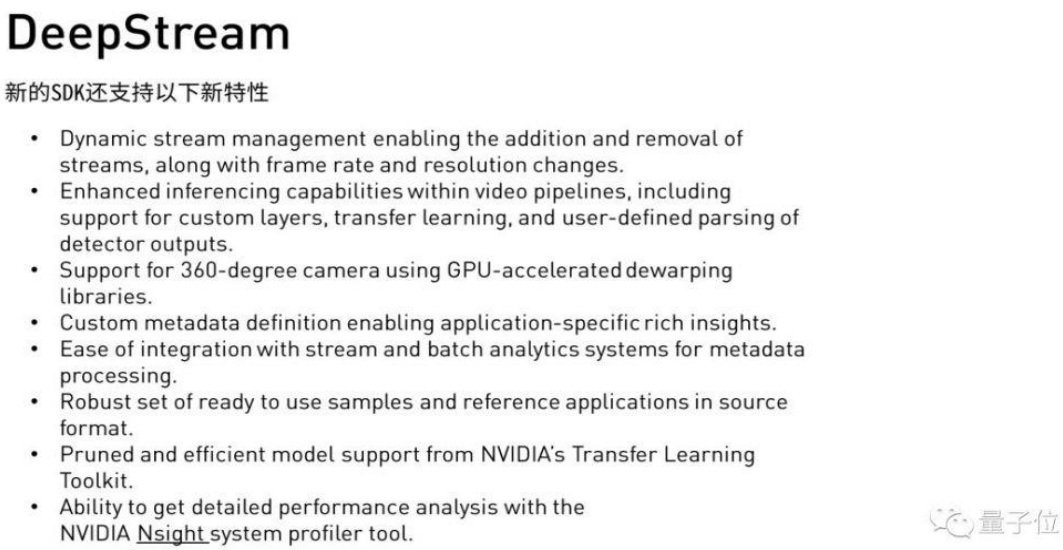
NVIDIA 提出了这样一个工具：TensorRT Inference Server，它针对于互联网企业。这个工具支持一个或者多个 GPU 的调用，支持所有的流行框架训练出来的模型，也能够动态处理。这就能帮助大家建立一个流程，特别是以核心算法为中心的这些公司，不用再因为使用率、利用率等等事情费神。

除此之外，把模型仓库和引擎分开，整个项目管理起来更方便。

二、Deepstream



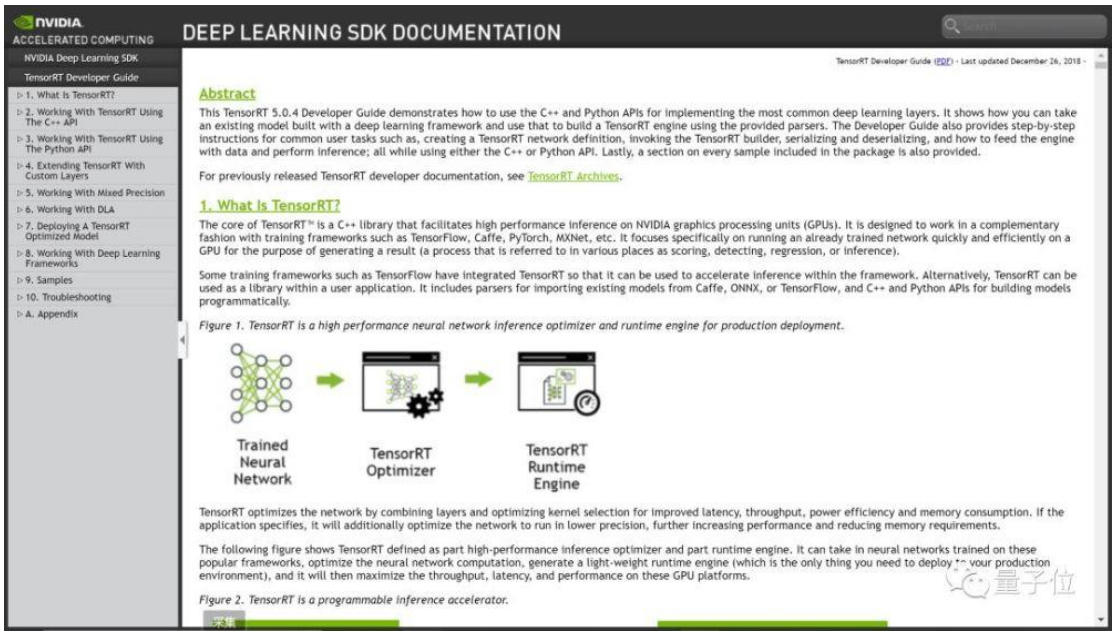
接下来介绍一个针对前端处理的工具叫 Deepstream。官网介绍 NVIDIA DeepStream SDK 例如前端摄像头数据出来之后，第一件事应该是编解码，拿着图片解码，再往下放到引擎里去处理。而 Deepstream 已经把这些都做好了，仅仅是视频的图像处理这部分，它处理的速度就比传统的方法就快很多。针对图片的大小不一样，帧率速度不一样，处理的速度也不一定成倍增加，但是 batch 越多，加速度比越高。



Deepstream 支持个流的执行，Pipeline 的建立等等特性。比如车载的设备上面，要很多个摄像头，Deepstream 可以自动删减需要的摄像头；检测行人时，更多需要的是前项的摄像头，那 Pipeline 里边就只用这四个摄像头，Deepstream 会自动把所有需要的摄像头数据拿来调用。Deepstream 在一些不需要的 Pipeline 里，也可以自动删减。比如在检测行人和车道线的场景里，同样需要前面这四个摄像头，下一次不需要检测行人了，Deepstream 也能自动删减。

这两个工具：TensorRT Inference Server 和 Deepstream，一个是针对网络 request，一个是针对嵌入式的，或者说是连接的摄像头的 Server。

对于 TensorRT 这个工具，如果觉得很难使或者说使用比较复杂，可以试试从 Python 开始，现在 TensorRT 和 Deepstream 包括其他的一些工具都是支持 Python 框架的，从 Python 开始练起，脚本就不用编译，对大家也更简单一些。



在官网上有一个 Developer guide 的内容，是非常好的教材，里面有很多例子，把例子读一遍、写一遍，再读一遍，再写一遍，非常快。TensorRT 出来时间不长，我们自己也没有用太长时间，现在是把底层的一些好的方法集合起来，给使用者一个接口，用更好的方式帮助大家规避操作中的问题。

三、Q&A

1、Faster Rcnn 的 Pytorch 和 Caffe2 模型是否支持?

现在是支持检测，只要转化到 Onnx 模型应该都支持的。

Ft32，怎么转化成 Int8，用什么算法，怎么计算，能说明下原理吗?

Ft32 转化 Int8，首先 NVIDIA 里有一个工具，**Nvinfer**，在库里有一个专门矫正数据的类，直接调用就行。

2、在实际操作中，随便挑了几百张图，能转到 int8，实际的计算速度大概是 3.6 倍，但是精度下降了，这是为什么?

用一个高精度的网络模型训练出来，转化到一个低精度数据类型的网络模型，在低精度网络模型里精度下降一点，很正常。但是为什么精度下降了，我们还要做这个呢？就是因为在可接受的范围内提高了很多速度。

3、那该选哪些样本做计算呢？另外就是数量，官方建议是 500 到 1000 张图，不知道是不是数量越多精度下降越少?

可以给更多的图片，但是精度肯定是会下降的少一些，但是这个不会说差别非常大。选样本最好是选择特征照片，首先得涵盖各种特征，除了各个类别之外，有正样本负样本等等。

4、别人转后的 TensorRT 的模型，转给我用，会给好几个不同的 batch size，这几个会有区别吗？对速度有什么影响呢?

首先 TensorRT 转的并不是模型，转过来的是 TensorRT 的一个引擎，引擎需要调用模型，

这个引擎里边有不同 batch size，但速度应该是不变，更多的 batch 意味着这吞吐量不同，也就是同时的计算能力不用，关系到是否能处理更多数据，跟效率有关。

在量子位公众号（QbitAI）界面回复“190110”，即可获得本次分享的完整版 PPT。