# 英伟达-阿里云异构计算

# TensorRT 加速AI推理Hackathon 2021

# 项目报告书

**参赛主题：**

**参赛单位：阿里巴巴**

**团队名称：阿里安全AI平台**

**团队成员：孟通、刘彬、田惠舟**

## 摘要

（扼要描述，包括：

* 选择了什么模型，
* 进行TensorRT加速的技术路线，例如：onnx导入加plugin开放，INT8量化等。
* 取得的成果，例如，在batchsize=16时，达到了最高加速比，时延/吞吐时多少。
* 工作的难点和创新点是什么，例如，开发了3个plugin，支持模型变长输入等。
* 后续计划等

）

## 一．背景介绍

（建议使用计划书里面关于模型的介绍即可。）

目前TensorRT推理库已经广泛应用到各大厂商的产品中，在推理场景下，对比于Tensorflow、Pytorch和Paddle等深度学习框架，TensorRT有雄厚的优势。诸如图结构优化、离线量化以及对于硬件更加硬件亲和的编码都成为TensorRT能够进行高性能推理的基石。在上述背景需要为本次大赛选择一个合理的模型，该模型需要尽可能充分的利用TensorRT特性，并且能够探索在Tensor框架下更多的优化方法和手段。经过充分论证我们团队选择了PaddleDetection工程下的“YOLOv3 ResNet50 DCN”模型为参赛模型，选择理由如下：

1. 为什么选择Paddle框架？
2. 相比较于已经广泛应用的tensorflow和torch框架，paddle的框架在异构部署方面难度更高，因为paddle本身的定位是对算法更具亲和性，所以引入很多非工程常规的算子，例如yolo\_box, deformable\_conv, multiclass\_nms等。这些算子属于paddle中的原子粒度，无法通过普通的图优化或者转义方式进行面向TensorRT的映射，需要特殊处理，有不小的挑战。
3. Paddle目前的使用度快速上升，尤其是对于中国用户和算法工程人员，未来会成为一个主要趋势，其中提供了很多优秀的预训练模型，在算法工程师中有很大的影响力。
4. Paddle提供inference方案中支持了TensorRT的接入，Paddle的对于TensorRT的使用已经做了不少支持工作，选择Paddle作为性能优化的对比benchmark具有更大挑战性。
5. 为什么选YOLOv3 ResNet50 DCN网络？
6. Yolov3网络与传统目标检测方法对比推理速度达到接近2倍，在CV领域有很大的应用市场。
7. PaddleDectection的 Yolov3网络精度要优于传统的Yolov3网络，前者经过优化后mAP达到43.2比后者的mAP高10.2个点（注：mAP越高越好），PaddleDectection的Yolov3由于其超高的精度深受业界欢迎。
8. YOLOv3网络中存在TensorRT不支持的算子，可以利用TensorRT支持plugin的特性，实现高性能plugin。并且由于YOLO系列的网络后处理原理基本一致，这些plugin也可利用到其他YOLO系列网络。

## 二．开发内容

### 2.1 模型结构和基本情况

（介绍初始模型的训练框架，模型结构，模型精度，模型包括的主要网络层，模型在原框架中的性能（此处可简要提及，后边有详细对比）。）

深度学习训练框架：Paddle是百度出品的开源深度学习框架，是完全国产化的产品，集深度学习核心框架、基础模型库、端到端开发套件、工具组件和服务平台于一体，2016 年正式开源，是全面开源开放、技术领先、功能完备的产业级深度学习平台。飞桨源于产业实践，始终致力于与产业深入融合。

“YOLOv3 ResNet50 DCN”模型是传统yolov3模型的改进版，其主要改进如下：

1. 将DarkNet53的主干更换为ResNet50-VD
2. 引入Deformable Convolution V2代替主干网络中stage5部分3x3的卷积操作
3. FPN部分增加DropBlack模块，提高泛化能力

模型中使用到的层：

|  |  |  |
| --- | --- | --- |
| Paddle层类型 | Trt是否支持 | Paddle结局方案 |
| conv2d | 支持 |  |
| batch\_norm | 支持 |  |
| relu | 支持 |  |
| pool2d | 支持 |  |
| Elementwise\_add | 支持 |  |
| split | 不支持 | 插件支持 |
| sigmoid | 支持 |  |
| Deformable\_conv | 不支持 | 普通支持 |
| Leak\_relu | 支持 |  |
| Nearest\_interp | 不支持 | 普通支持 |
| concat | 支持 | 后处理concat层为paddle实现的 |
| Yolov\_box | 不支持 | 普通支持 |
| Transpose2 | 不支持 | 普通支持 |
| Multiclass\_nms | 不支持 | 普通支持 |
| scale | 不支持 | 普通支持 |

模型结构：模型由ResNet50-VD作为主干，加上改进后的fpn结构和paddle自定义实现的yolov\_box和nms层组成，其结构如图-2，图中可复用的结构已经做了抽象，称为“Bottleneck”，图-1描述了三种基本结构：

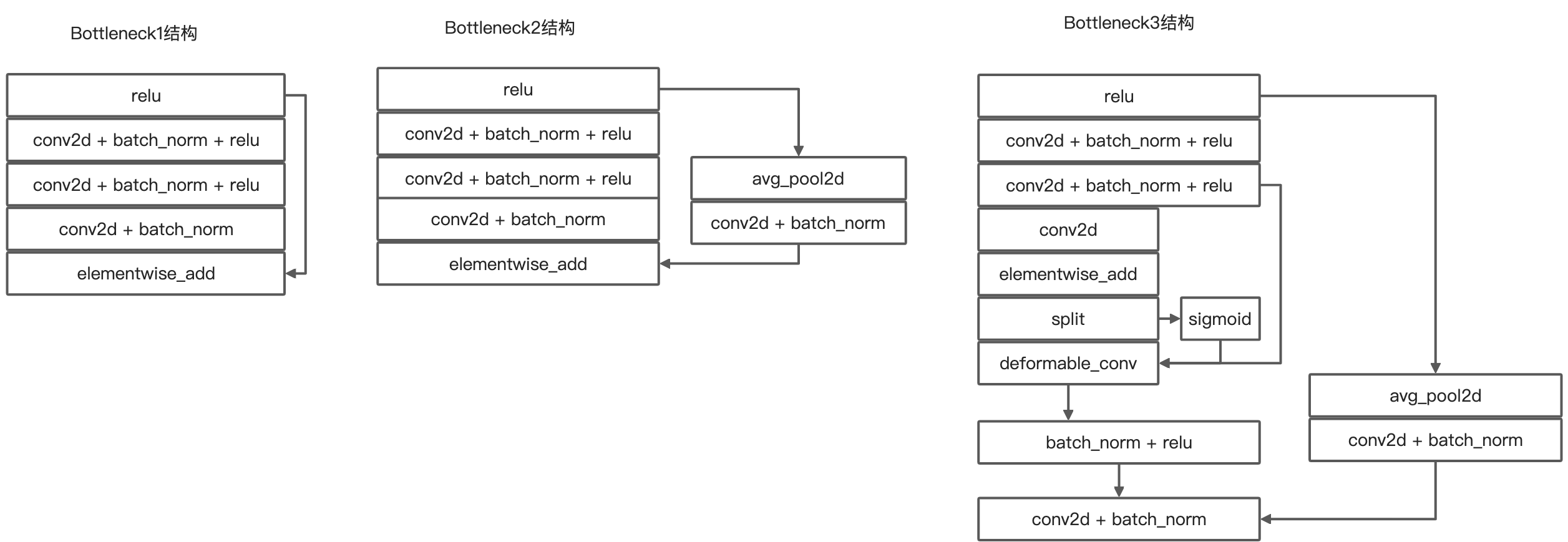


图-1 网络结构中基本结构描述

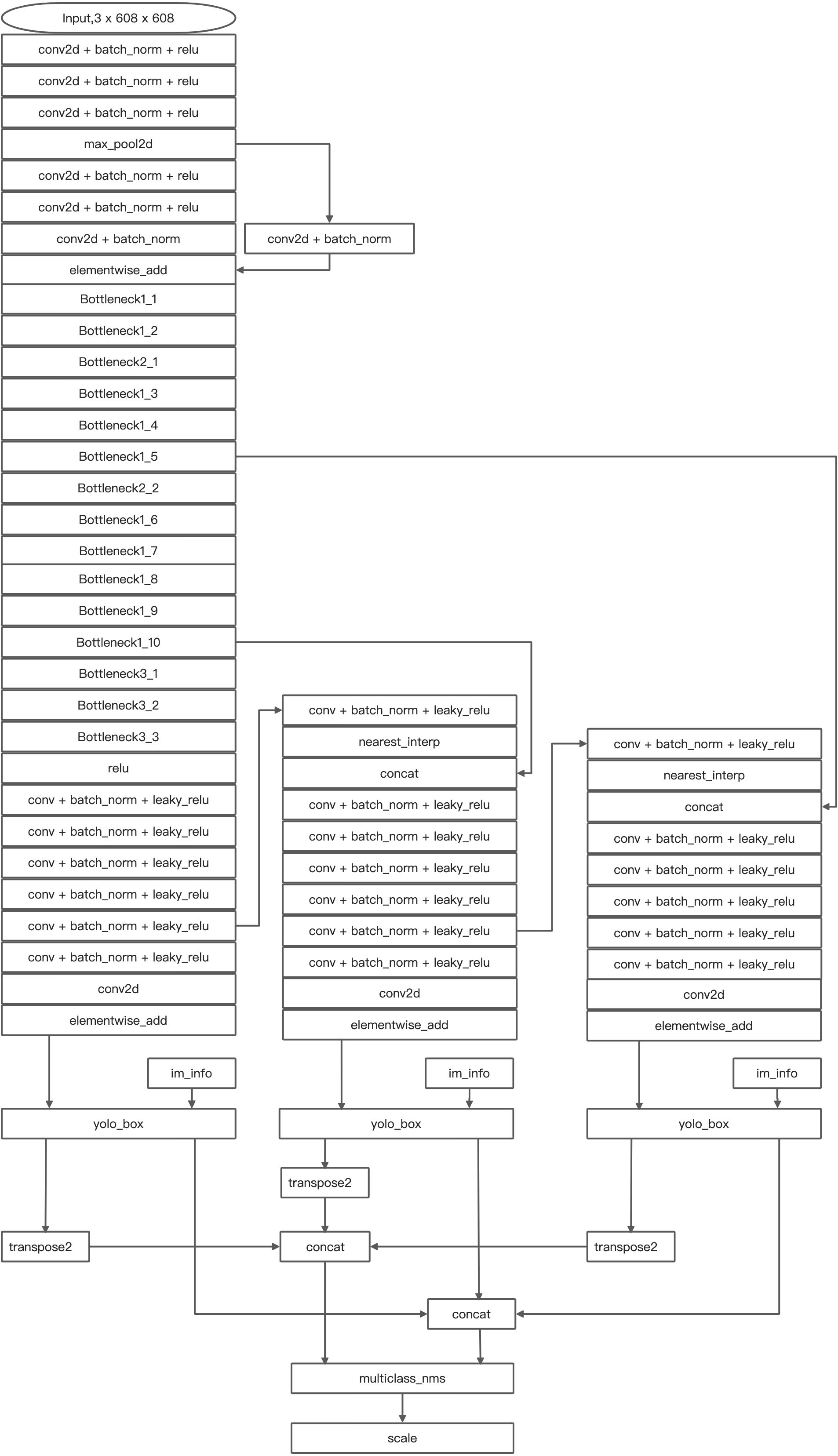


图-2 网络结构图

模型下载地址：

模型网络地址：

模型在原始框架中的精度和性能：

利用paddle框架和官方提供的评估接口，进行性能测试，性能数据如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| Batchsize | Latency(单位:ms) | Throughput(单位:fps) | Precision(单位:mAP) |
| 1 | 71 | 14 | 0.425 |
| 2 | 43 | 23 | 0.425 |
| 4 | 42 | 23.7 | 0.425 |
| 8 | 40 | 24.7 | 0.425 |
| 16 | 37.9 | 26.4 | 0.425 |
| 32 | 41.6 | 24 | 0.425 |
| 64 | 45 | 22.24 | 0.425 |

注：超过batch = 128 触发内存空间不足错误

### 2.2 技术路线选择

（介绍选择了什么样的技术路线，为什么。可以从几个方面说明（选取1到若干个原因解释）：

1. 现有技术路线的比较，onnx，framework-trt，api等。考虑选择某一个路线的原因，例如，时间原因，可以复用已有公开成果，etc.
2. 创新性考虑，例如，重点放在量化，所以模型导入不是工作重点，etc.
3. 性能考虑，例如，进行了profiling，发现问题主要是在小batchsize情况性能不好，所以希望通过TensorRT改善小batch inference的性能等。
4. 可扩展性考虑，虽然参加比赛，但是考虑到未来工作的可扩展性等。
5. 实用性考虑，例如考虑到未来上线环境，c++环境更友好等。

）

### 2.3 开发工作和遇到的困难

（重点描述做了哪些工作，工作时间的安排，例如第一周做了什么，第二周做了什么，第三周做了什么。遇到了哪些困难，如何解决的（可以是workaround，或者是feature的扩展/plugin的编写等。可以用流程图，示意图等描述。）

## 三．性能结果

### 3.1 测试环境

硬件环境：

|  |  |
| --- | --- |
| 硬件类型 | 详细信息 |
| GPU | Tesla T4 |
| CPU | Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz |
| 内存 | 30G |

宿主机环境：

|  |  |
| --- | --- |
| 宿主机系统 | Ubuntu 7.5.0-3ubuntu1~18.04 |
| Docker | Docker version 20.10.6 |
| 测试镜像 | nvcr.io/nvidia/tensorrt:21.02-py3 |

软件环境（镜像环境）：

|  |  |
| --- | --- |
| 软件名称 | 版本 |
| Gpu驱动 | 460.32.03 |
| Cuda | 11.2 |
| Cudnn | 8.2 |
| TensrRT | 7.2.2 |
| Python | 3.8.5 |
| Gcc/g++ | 8.4.0 |

Python库环境：

|  |  |
| --- | --- |
| 库名称 | 版本 |
| Cython | 0.29.23 |
| matplotlib | 3.4.1 |
| numpy | 1.19.5 |
| nvidia-pyindex | 1.0.8 |
| opencv-python | 4.5.1.48 |
| paddlepaddle-gpu | 2.0.1 |
| Pillow | 8.1.0 |
| polygraphy | 0.22.0 |
| protobuf | 3.14.0 |
| pycocotools | 2.0 |
| pycuda | 2020.1 |
| PyGObject | 3.36.0 |
| python-dateutil | 2.8.1 |
| pytools | 2021.1 |
| PyYAML | 5.4.1 |
| pyzmq | 22.0.3 |
| requests | 2.25.1 |
| scipy | 1.6.2 |
| setuptools | 45.2.0 |
| six | 1.15.0 |
| tensorrt | 7.2.2.3 |
| tqdm | 4.60.0 |
| urllib3 | 1.26.4 |

（详细描述原始模型的运行环境，加速后模型的测试环境，包括软硬件环境）

### 3.2 正确性验证

（描述验证方法，可以是单层1对1比对，误差范围（相对误差）是多少；也可以是最终的预测效果的数据，如果是以最终预测效果为准，请说明测试数据集规模，和最终效果计算的方法，例如top5的accuracy等。如果涉及多个精度（并且您想强调该工作支持多种精度），请分开说明。）

由于我们选用的模型是检测模型，目前对于检测模型评估主要以mAP的大小来衡量模型的精度，故我们对于正确性的验证设计的方案为，比较同一个数据集的mAP，来验证优化后模型的正确性。选用的数据集为coco val2017数据集，其中包含5**000张图片。**

精度数据如表-1，表中的数据为mAP的数值，数值越大越好，int类型的模型推理是需要经过量化，Paddle-TensorRT和Quake-TensorRT使用相同的量化图集，来自于coco train2017数据集的前500张。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 推理框架类型 | | |
| 推理类型 | Paddle | Paddle-TensorRT | Quake-TensorRT |
| Float32 | 0.425 | 0.409 | 0.406 |
| Float16 | 无 | 0.409 | 0.406 |
| Int8 | 无 | 0.35 | 0.396 |

表-1 mAP数据统计

结论：1. Paddle-TensorRT推理库在使用fp32和fp16时对比原始框架mAP降低，可能与TensorRT推理相关。

2. Paddle-TensorRT在int8的场景下，相比于Quake-TensorRT的mAP下降明显，需要进行详细分析。

### 3.3 性能测试

提供完整性能对比表格，并附上运行时“nvidia-smi“输出的gpu频率和memory 频率，确保测试性能时没有别的数据在运行，可以提供运行时GPU memory使用情况。

可以提供end-to-end的性能，也可以只提供模型预测部分的性能。例如，既可以是从图片输入，预处理，到最后结果统计的时间；也可以是raw data（例如RGB normalized数据）输入，到最后出inference结果的时间。

需要提供完整的latency vs throughput 表格和图形，可以按照实际情况取batchsize，计算方法如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Batchsize | Latency, ms | Throughput  (1000/latency\*batchsize) | Latency Speedup  (TRT latency / original latency) | Throughput speedup (TRT throughput / original thoughput) |
| PyTorch | 1 |  |  | 1x |  |
| 8 |  |  |  |  |
| 16 |  |  |  |  |
| 32 |  |  |  |  |
| TensorRT | 1 |  |  |  |  |
| 8 |  |  |  |  |
| 16 |  |  |  |  |
| 32 |  |  |  |  |

绘制Latency vs. throughput 图形，横坐标latency (ms)，纵坐标throughput。指出你认为最应该选择的batchsize并说明原因（可能是实际应用对latency有要求，可能是这个情况性能最好，等等）

性能测试方案说明：由于不同的图片推理耗时不同、硬件的状态每个时间点也不同，为了避免这些系统误差引起的性能耗时不准确，我们采用统计一个数据集的整体耗时，并且计算单张图片的平均耗时来衡量模型推理的性能，在进行正式测试前，需要预热100次，来降低系统误差，保证测试结果的有效性。会分别统计和对比FP32的、FP16和Int8的模型性能耗时。

Flaot32性能测试数据见表，其中每个测试数据对应的的GPU运行信息会以图片的形式展示在附件中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BatchSIze | Latency(单位:ms) | Throughput(单位:fps) | GPU运行信息 |
| Paddle | 1 | 71 | 14 | 图3 |
|  | 2 | 86 | 23 | 图4 |
|  | 4 | 84 | 23.7 | 图5 |
|  | 8 | 40 | 24.7 | 图6 |
|  | 16 | 37.9 | 26.4 | 图7 |
|  | 32 | 41.6 | 24 | 图8 |
| Paddle-TensorRT | 1 | 26.3 | 38 |  |
|  | 2 | 52.6 | 38 |  |
|  | 4 | 111.6 | 35.8 |  |
|  | 8 | 233 | 34.3 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Quake-TensorRT |  |  |  |  |

Float16性能数据见表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BatchSIze | Latency(单位:ms) | Throughput(单位:fps) | GPU运行信息 |
| Paddle-TensorRT | 1 | 21.8 | 45.9 |  |
|  | 2 | 38.6 | 51.8 |  |
|  | 4 | 70.8 | 56.5 |  |
|  | 8 | 117.4 | 68.1 |  |
|  | 16 | 194 | 82.5 |  |
|  |  |  |  |  |
| Quake-TensorRT |  |  |  |  |

Int8性能数据见表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BatchSIze | Latency(单位:ms) | Throughput(单位:fps) | GPU运行信息 |
| Paddle-TensorRT | 1 | 21.6 | 46.2 |  |
|  | 2 | 38 | 52.6 |  |
|  | 4 | 69.3 | 57.7 |  |
|  | 8 | 117.8 | 68 |  |
|  | 16 | 191 | 83.8 |  |
|  | 32 | 386.1 | 83 |  |
| Quake-TensorRT |  |  |  |  |

## 四．其他结果

（例如开发了通用工具，提交了trt bug等，或者详细的profiling过程等。你认为除了性能以外需要特别提及的亮点）

## 五．未来计划

（如果后续进一步工作，或者有更长时间，还有什么进一步优化的计划）

## 六．代码位置

（具体提供开源代码的地址，branch。注意，比赛结果提交的时候需要代码开源，并且创建专门的tag：”TRT2021”，后续你可以继续对你的代码进行优化，我们的评奖按照这个tag来。添加Readme文件， 说明编译步骤和运行环境， 比如使用了哪个docker image。性能测试/evaluation的代码也需要放在repo里面。）

## 七．TensorRT改进建议

（可以列举一些目前还不支持的特性或是能够使得工具更具易用性和通用性的技术反馈。）

## 参考文献

（如果有的话）

# 附件：评分标准，和对应报告的部分（不用包含到报告之中）

1. ⽂档描述清晰。20分 => 对应整个文档
2. 代码整⻬，逻辑清晰。30分 => 对应最后公开的代码，参见 “六.代码位置”部分
3. 模型顺利运⾏，有加速效果。30分 => 参见“三.性能结果”部分
4. 如果模型开发过程中发现TensorRT 的bug，提交bug，得到确认。5分每个bug。 => 在 “2.3 开发工作和遇到的困难” ，“四. 其他结果”和“七. TensorRT 改进建议” 中体现。
5. 开发了相应 Plugin 或 CUDA 代码。20分 => 在代码和“2.2 技术路线选择” 和“2.3 开发工作和遇到的困难”中描述。
6. 进⾏了 Profiling，对性能优化提出进⼀步的意⻅。20分 => 在“2.2 技术路线选择”，“2.3 开发工作和遇到的困难”和“四. 其他结果”中体现。
7. 进⾏了 INT8 量化的⼯作。20分 => 代码，“三.性能结果”， “2.2 技术路线选择” 和“2.3 开发工作和遇到的困难”中描述。

附件：

GPU运行状态信息截图：

Paddle框架Float32模型推理时GPU信息

|  |  |  |
| --- | --- | --- |
| Batch | GPU状态信息 | 分析 |
| 1 |  |  |
| 2 |  |  |
| 4 |  |  |
| 8 |  |  |
| 16 |  |  |
| 32 |  |  |

Paddle-TensorRT推理Float32模型时GPU运行时信息

|  |  |  |
| --- | --- | --- |
| Batch | GPU状态信息 | 分析 |
| 1 |  |  |
| 2 |  |  |
| 4 |  |  |
| 8 |  |  |

Quake-TensorRT推理Float32模型时GPU运行时信息

|  |  |  |
| --- | --- | --- |
| Batch | GPU状态信息 | 分析 |
| 1 |  |  |
| 2 |  |  |
| 4 |  |  |
| 8 |  |  |

Paddle-TensorRT推理Float16模型时GPU运行信息

|  |  |  |
| --- | --- | --- |
| Batch | GPU状态信息 | 分析 |
| 1 |  |  |
| 2 |  |  |
| 4 |  |  |
| 8 |  |  |
| 16 |  |  |

Paddle-TensorRT推理In8模型时GPU运行信息

|  |  |  |
| --- | --- | --- |
| Batch | GPU状态信息 | 分析 |
| 1 |  |  |
| 2 |  |  |
| 4 |  |  |
| 8 |  |  |
| 16 |  |  |
| 32 |  |  |