综合实验手册

一、目标检测网络-YOLOv3

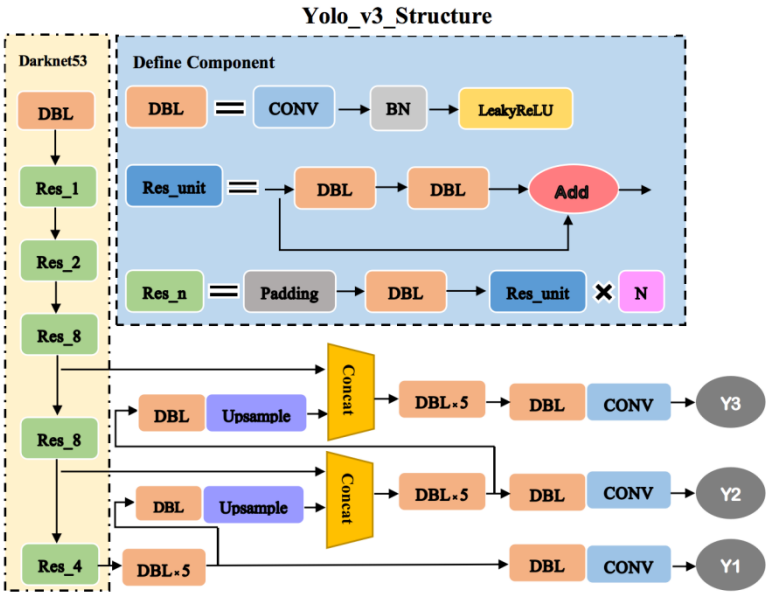
1.实验目的：通过用智能编程语言实现 YOLOv3 的部分后处理操作，深入掌握智能编程语言的算子开发和优化方法；通过在 TensorFlow 中添加大算子，掌握在 TensorFlow 框架中添加融合算子的方法；通过使用 TensorFlow 进行在线推理，掌握使用 TensorFlow 编写目标检测应用并在典型 DLP 上进行优化的方法。

2.实验描述：该实验难度中等，难度系数分为 1.1，学生最终得分为难度系数分乘以相应的评判标准得分。

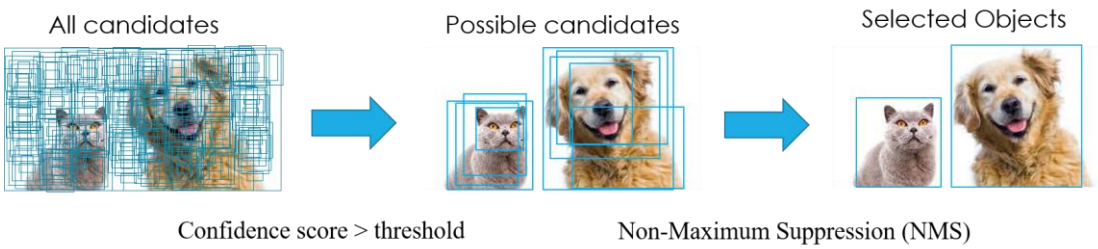
3.背景介绍：

a) 目标检测：目标检测是指针对一个或多个特定类别在数字图像中进行视觉对象的分类和定位。目标检测属于多任务学习，也就是说一个任务分支需要通过分类算法将物体根据类别划分，另一个任务分支是在判断类别后标记出每一个类别的具体位置信息。依据深度学习算法阶段不同，目标检测方法主要可以分为两类：一种是基于 Region Proposal 的两阶段检测，另一种是单阶段检测，仅使用卷积神经网络直接将目标定位框转为回归问题，预测不同类别的目标位置。

b) YOLOv3：本实验所使用的 YOLOv3 网络是经典的单阶段检测方法，其延续之前版本 YOLOv1 和 YOLOv2 主要特点，包括通过划分单元格来做检测，使用 Leaky ReLU 作为激活函数，实现端到端巡礼，使用 Batch Normalization 避免过拟合等。此外通过修改 Backbone 网络、使用多尺度训练等方式进一步改进了网络。下述结构图包含了从图片输入经过 Darknet-53，Concat 拼接到输出 3 个不同尺度 Feature Map 的过程。关于 YOLO 系列算法的介绍，在智能计算系统书中已有相信介绍，同学们可基于此阅读相关论文，了解更详细的内容。



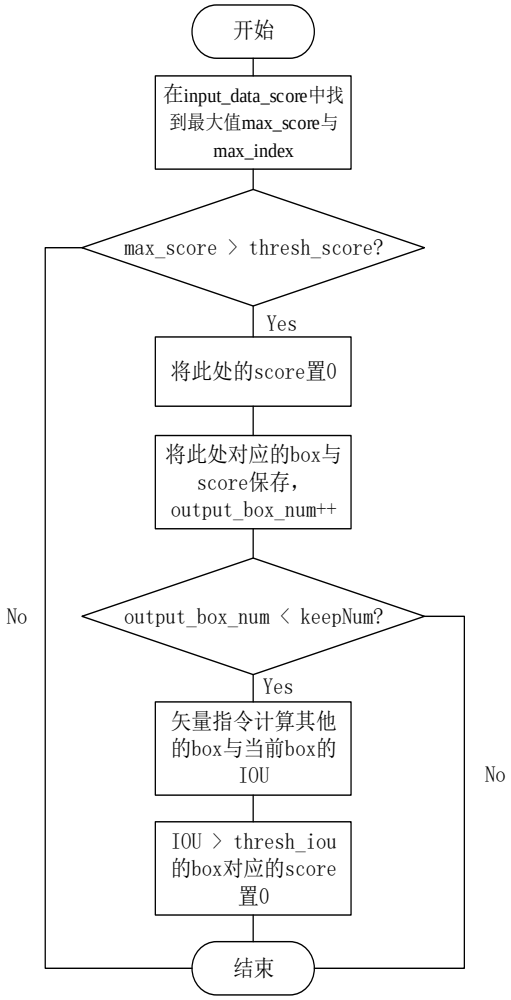
c) NMS：NMS(非极大值抑制)是一种针对目标框的多重性而提出的检测技术，是目标检测网络中一个重要环节，由于目标检测相邻的交叉窗口经常出现分数相近的情况，使用 NMS 可以去掉重复的检测框。



NMS 的计算过程可概括为以下四个步骤：(1)对所有预测框的置信度降序排序；(2)选出置信度最高的预测框，确认其为正确预测，并计算它与其他预测框的 IOU；(3)根据前一步中计算的 IOU 去除重叠度高的框，即 IOU> thresh_iou 就删除；(4)剩下的预测框返回第 1 步，直到没有置信度 > thresh_score 的框为止。

使用 BANGC 实现时，如果按照上述 NMS 原理进行计算，则会显得较为复杂，比如第(3)步中删除交并比大于阈值的框，使用 BANGC 语言实现这个操作需要来回拷贝数据，导致效率不高，因此推荐大家采用另外一种方式，即

不动之前的数据，被删除的框对应的 score 置 0，下一轮筛查的时候，删除的框必然不会对此轮筛查造成影响。流程图如下：



4.实验内容：

a) YOLOv3 DLP 运行：对原始模型进行量化，将推理代码移植在 DLP 上；

b) NMS 的 BCL 实现：本次实验的重点；

c) 框架集成：将 CNPlugin 中 YOLOv3 后处理大算子集成到 TensorFlow 编程框架中；

d) 完整框架推理测试：修改 pb 模型，添加后处理算子，执行 TensorFlow 的在线推理。

5.实验步骤：

a) 登录云平台：ssh xxx@120.236.247.203 -p xxx

b) 初始化环境：cd /opt/AICSE-demo-student/env；source env.sh

c) cd /opt/AICSE-demo-student/demo/yolov3/bangc/PluginYolov3DetectionOutputOp

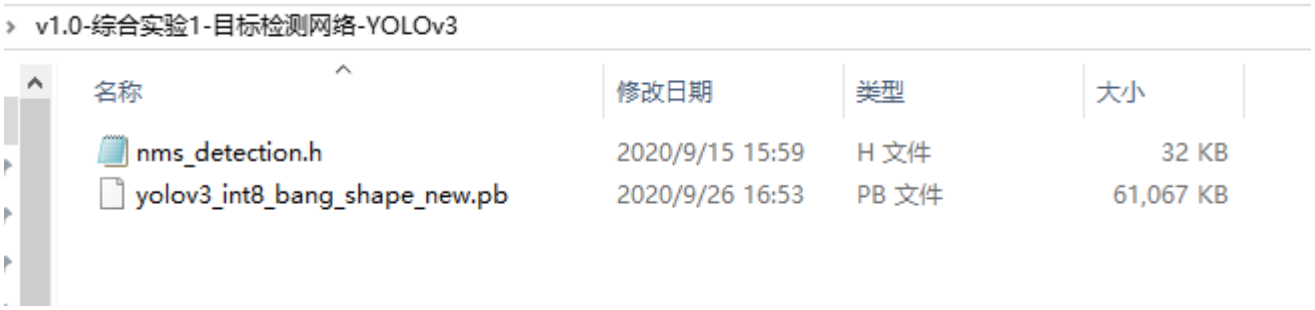
d) NMS BANGC 算子实现，补全 nms_detection.h 文件。

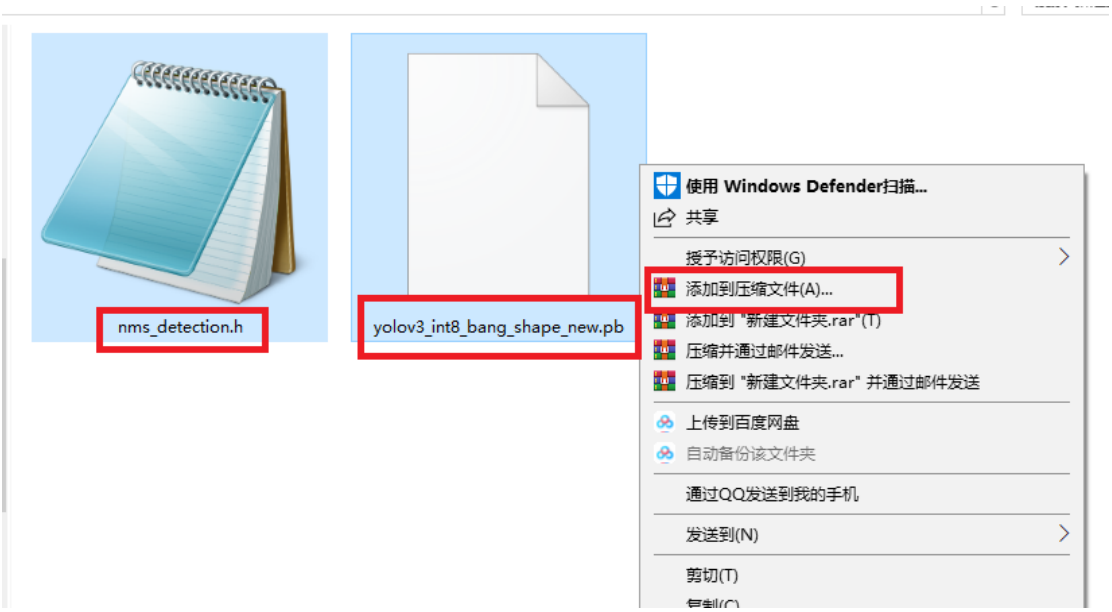
e) TensorFlow 算子集成，将下述文件夹中的文件依次添加到 TensorFlow 源码中（由于课程时间关系，该部分代码直接给出）并重新编译 TensorFlow：/opt/AICSE-demo-student/demo/yolov3/tf-implementation/tf-1.14-detectionoutput；/opt/AICSE-demo-student/env/tensorflow-v1.10

f) 框架推理测试，在提供的 pb 模型中添加后处理大算子，并将该节点命名为 Yolov3DetectionOutput。分别进入 /opt/AICSE-demo-student/demo/yolov3/ yolov3-bcl/demo；/opt/AICSE-demo-student/demo/yolov3/yolov3-release 文件夹下，执行：run_aicse.sh 比较精度性能差异。

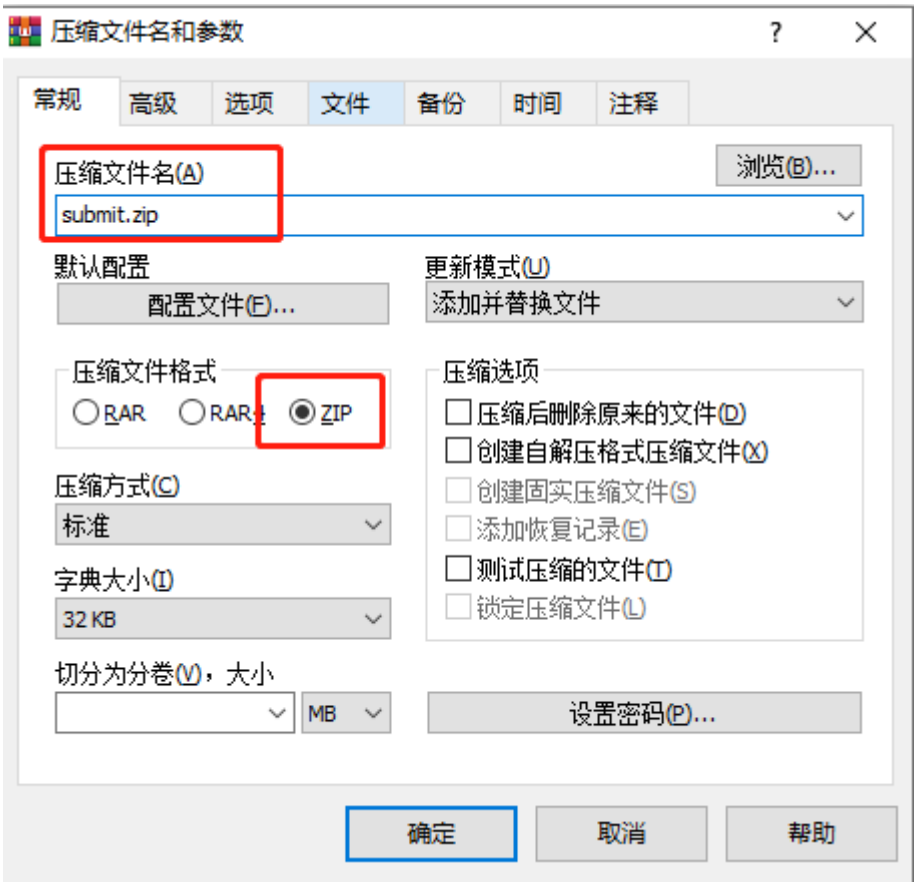
6. 上传自动评测平台

a) 将程序按以下文件列表整理，打包成压缩文件，请选中所有文件直接压缩，不要压缩文件夹。请注意文件名和图中保持一致。





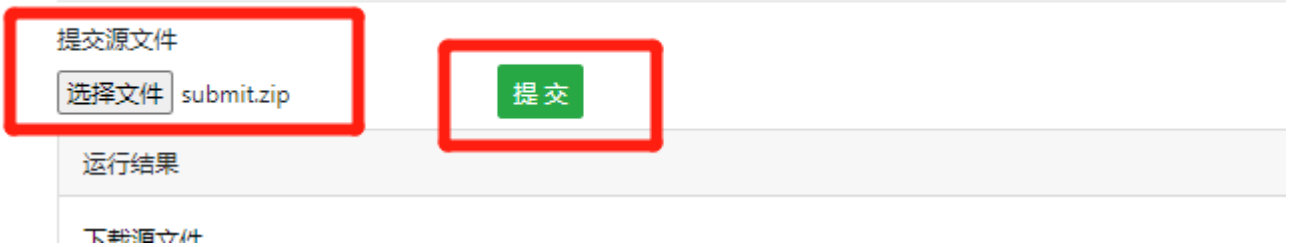
b） 压缩文件格式为 zip 格式。文件名命名为 submit.zip



c） 将 submit.zip 上传到评测平台。选择文件并点击提交。

5. 评分标准

60分 标准：补全nms_detection.h文件，cnplugin可正常编过；
70分 标准：在60分的基础上完成TensorFlow的集成编译，完成pb模型添 加后处理大算子的操作，执行测试脚本时
80分 标准：在70分的基础上，执行测试脚本时map值高于50%，单batch延 时(包含后处理)低于 100ms；
90分 标准：在80分基础上，执行测试脚本时map值高于54%，单batch延 时(包含后处理)低于 50ms；
100分标准：在90分基础上，执行测试脚本时map值高于56%，单batch延 时(包含后处理)低于 25ms。



d） 上传之后请等待系统评测。

e） 评测结束之后会自动显示评测结果。

90分 标准：在80分基础上，执行测试脚本时map值高于54%，单batch延 时(包
100分标准：在90分基础上，执行测试脚本时map值高于56%，单batch延 时(包

提交源文件

选择文件 submit.zip 提 交

运行结果

下载源文件

得分70.00 最后一次提交时间:2020-10-19 19:42:29

Accept

NMS评测结果	
单batch延时(包含后处理)(ms)	map值(%)
38.63787651062012	49.93

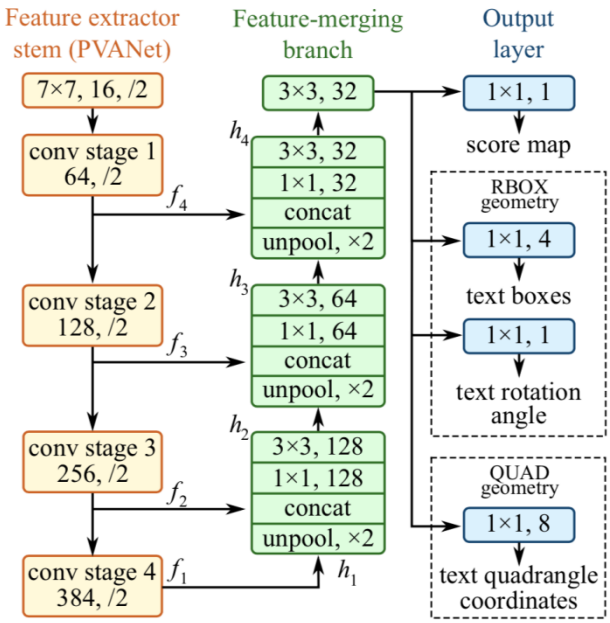
二、文本识别 OCR-EAST

1.实验目的：本实验主要完成光学字符识别的的代表性算法 EAST 网络，将其移植在智能处理器 DLP 中，并进行性能优化。通过使用智能编程语言实现 EAST 网络中的 Split+Sub+Concat 合并算子，深入掌握智能编程语言的算子开发和优化方法；通过在 TensorFlow 中添加合并算子，掌握在 TensorFlow 框架中添加融合算子的方法；通过使用深度学习框架进行在线推理，掌握使用 TensorFlow 编写文本识别应用并在典型 DLP 上进行优化的方法。

2.实验描述：该实验难度一般，难度系数分为 1.0，学生最终得分为难度系数分乘以相应的评判标准得分。

3.背景介绍：

a) OCR：光学字符识别(Optical Character Recognition,OCR)是指对文本资料的图像文本进行文字识别，获取文本资料版面信息的过程。OCR 过程大致可以分为：图像获取-> 图像预处理-> 文字检测-> 文字识别-> 输出。本次实验只涉及到文字检测，不涉及文字识别的任务。



b) EAST：算法全名是 Efficient and Accuracy Scene Text。其 pipeline 主要分为两个阶段: 第一阶段是基于 FCN 模型的文本框检测；第二阶段是对生成的文本框（旋转或矩形）经过非极大值抑制得到最终结果。FCN 网络结构如上图所示，主要分为三个部分：

第一部分：Feature extractor stem(PVANet)。该部分是主干特征检测网络，引入多尺度检测的思想，提取不同尺寸卷积核下的特征并用于后期的特征组合，以适应文本行尺度的变化。作者采用了 PVANet 模型作为主干网络，实际使用的时候也可以采用 VGG16 或者 Resnet；

第二部分：Feature-merging branch。该部分运用了 Unet 的思想，主要是合并第一部分提取的特征，通过池化和 Concat 来恢复尺寸。其中，特征合并主要通过逐层合并的方式，首先将第一部分对应的特征图输入到一个池化层恢复尺度，然后与当前层特征图进行 Concat，再通过 1x1 卷积来减少通道数，最后利用 3x3 卷积将局部信息融合以最终产生该合并阶段的输出。

第三部分：Output Layer。该部分输出分为 3 类：score_map，RBOX geometry，QUAD geometry。Socre map 代表此处是否有文字的可能性；RBOX geometry 中的五个通道，分别代表每个像素点到文本框边线的距离，以及文本框的旋转角；QUAD geometry 中的八个通道，分别代表着每个像素点到文本框任意四边形的 xy 距离。

4.实验内容：

a) EAST DLP 运行：对原始模型进行量化，将单精度模型量化为 INT8 模型；并将推理代码移植在 DLP 上。
b) Split+Sub+Concat 合并算子的 BCL 实现：本次实验的重点。
c) 框架集成:将用 BCL 实现的 Split+Sub+Concat 合并算子集成在 CNPlugin 中，通过 CNPlugin 接口集成在 TensorFlow 框架中。
d) 完整框架推理测试：修改 pb 模型，添加合并算子，执行 TensorFlow 的在线推理，比较优化前后性能差异。

5.实验步骤：

a) 登录云平台：ssh xxx@120.236.247.203 -p xxx

b) 初始化环境：cd /opt/AICSE-demo-student/env；source env.sh

c) cd /opt/AICSE-demo-student/demo/east/cnplugin-SBC

d) 完成 BANGC 算子实现与测试，补全 spilt_sub_concat_kernel.h spilt_sub_concat _kernel.mlu，完成编译与测试工作。

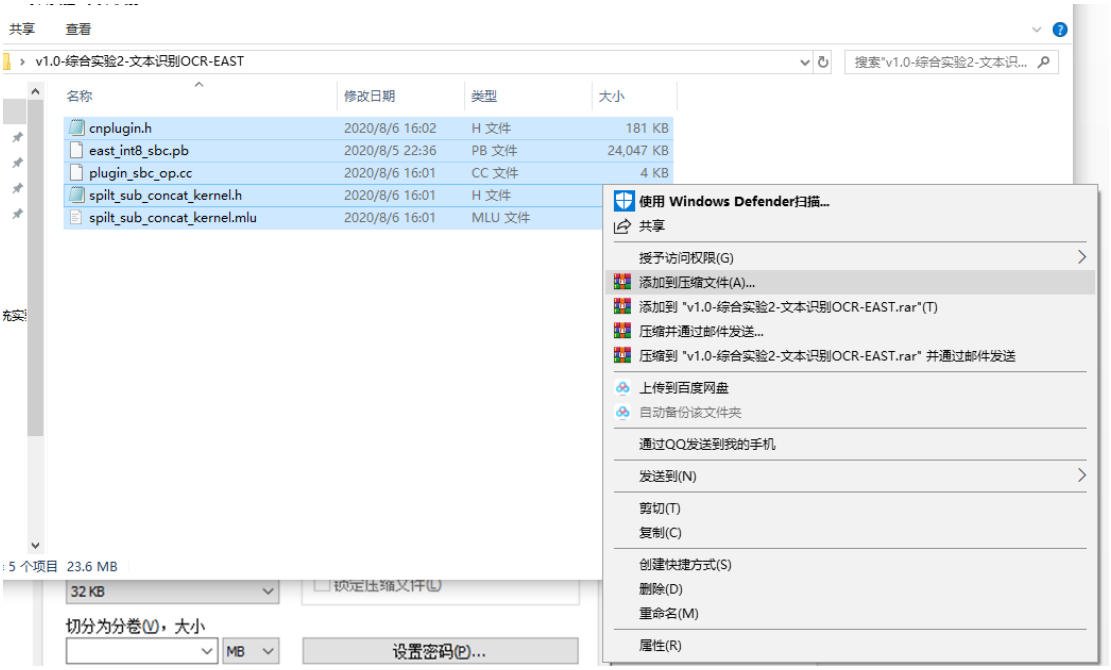
e) CNPlugin 算子集成，补全 plugin_sbc_op.cc 和 cnplugin.h 并编译新的 Cambricon-CNPlugin。

f) TensorFlow 算子集成，将下述文件夹中的文件依次添加到 TensorFlow 源码中（由于课程时间关系，该部分代码直接给出）并重新编译 TensorFlow ： /opt/AICSE-demo-student/demo/east/tf-SBC ； /opt/AICSE-demo-student/env/ tensorflow-v1.10

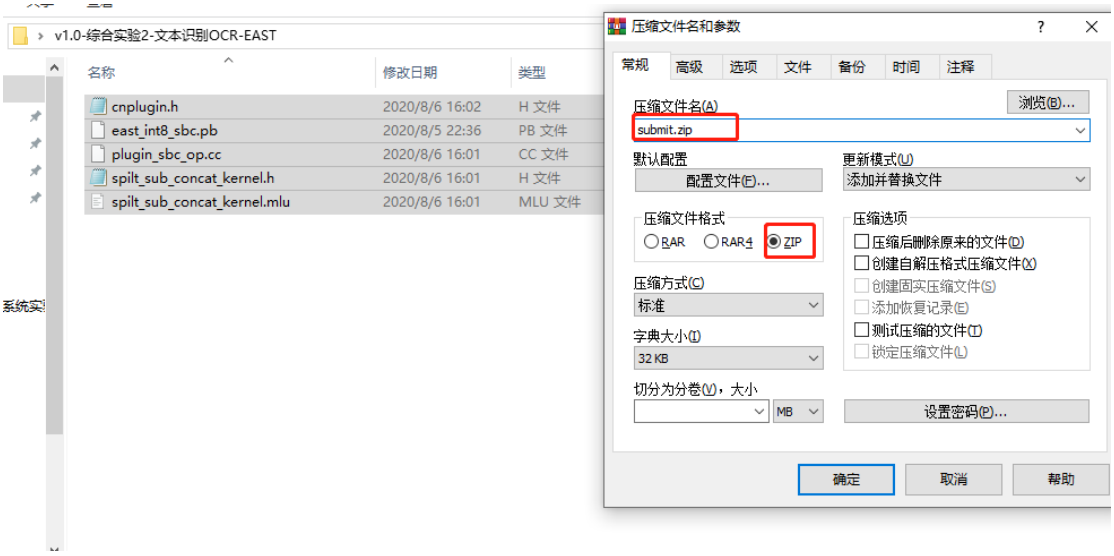
g) 框架推理测试，在提供的 pb 模型中添加合并算子。进入 /opt/AICSE-demo-student/demo/east/EAST 文件夹下，执行：run_aicse.sh 比较精度性能差异。

6.上传自动评测平台

a) 将程序按以下文件列表整理，打包成压缩文件。请注意文件名和图中保持一致。



b） 压缩文件格式为 zip 格式。文件名命名为 submit.zip



c） 将 submit.zip 上传到评测平台。选择文件并点击提交。

d） 上传之后请等待系统评测。

e） 评测结束之后会自动显示评测结果。

运行结果

下载源文件

得分90.00　最后一次提交时间:2020-10-14 21:37:50

Accept

BangC算子评测结果	
name	平均执行时间(ms)
BangC算子评测结果	2.556

Tensorflow集成评测结果				
name	Origin执行时间(ms)	Origin准确率(%)	SBC执行时间(ms)	SBC准确率(%)
Tensorflow集成评测结果	173.678	0.8363064008394544	138.558	0.8363064008394544

三、自然语言处理-BERT

1.实验目的：本实验主要完成将自然语言处理的代表性算法 BERT 网络移植到智能处理器 DLP 上，并进行性能优化与比较，使读者可以借助 DLP 处理器完成完整的自然语言处理任务。通过用智能编程语言实现 BERT 模型中的 BatchMatMulV2 算子，深入掌握智能编程语言的算子开发和优化方法；通过在 TensorFlow 中添加大算子,掌握在 TensorFlow 框架中添加融合算子的方法;通过使用 TensorFlow 进行在线推理，掌握使用 TensorFlow 编写 NLP 应用并在典型 DLP 上进行优化的方法；通过与 BANGC 大算子实现的比较，体会 DLP 相比 GPU 和 CPU 的优势。

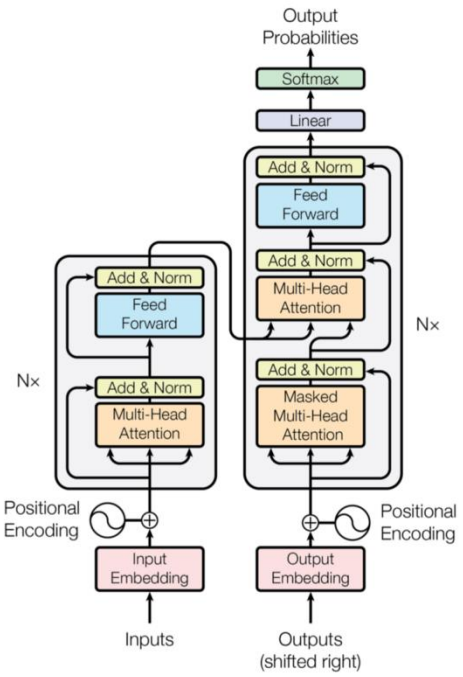
2.实验描述：该实验难度偏高，难度系数分为 1.2，学生最终得分为难度系数分乘以相应的评判标准得分。

3.背景介绍：

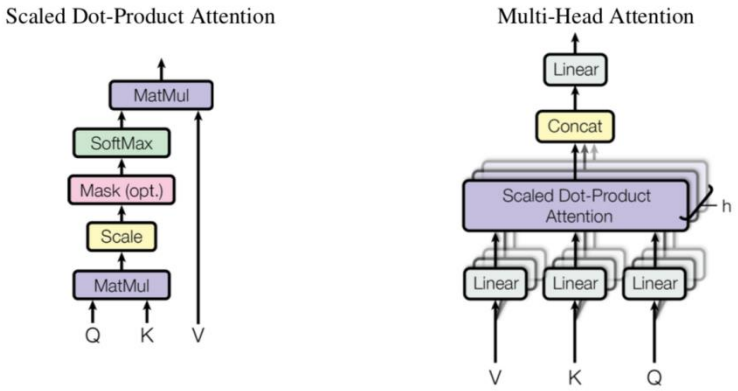
a) 注意力机制：Attention 是一种用于提升基于 RNN(LSTM 或 GRU)的 Encoder+Decoder 模型效果的的机制，一般也称为注意力机制。注意力机制现已广泛应用于机器翻译、语音识别、图像标注(Image Caption)等领域。Attention 之所以受欢迎的原因在于其给模型赋予了区分辨别的能力，例如，在机器翻译、语音识别应用中，为句子中的每个词赋予不同的权重，使神经网络模型的学习变得更加灵活。常见的注意力机制可分为许多种类，在 BERT 中使用的是 Self Attention。

b) Self Attention：与传统的 Attention 机制不同，传统的 Attention 是基于输入端和输出端的隐层单元计算 Attention 的；而 Self Attention 是分别对输入端与输出端自身进行计算，再把输入端得到的 Self Attention 加入到输出端得到的 Attention 中。

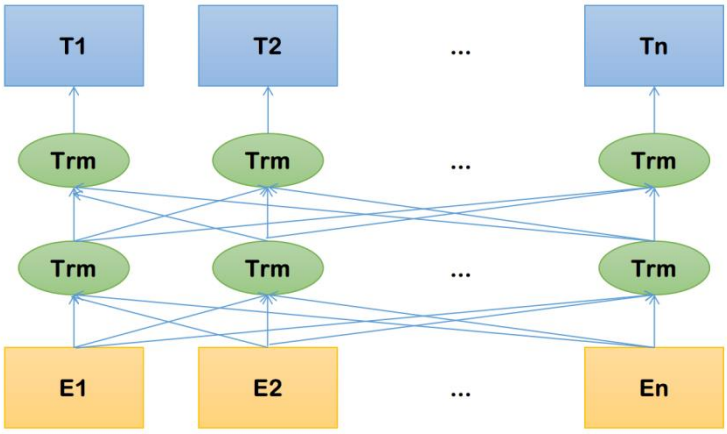
c) Transformer：其结构图如下所示，编码部分输入（Inputs）对应训练的数据集，解码部分输入(Outputs shifted right）为相应的标签。



Multi-Head Attention 结构图如下图所示：



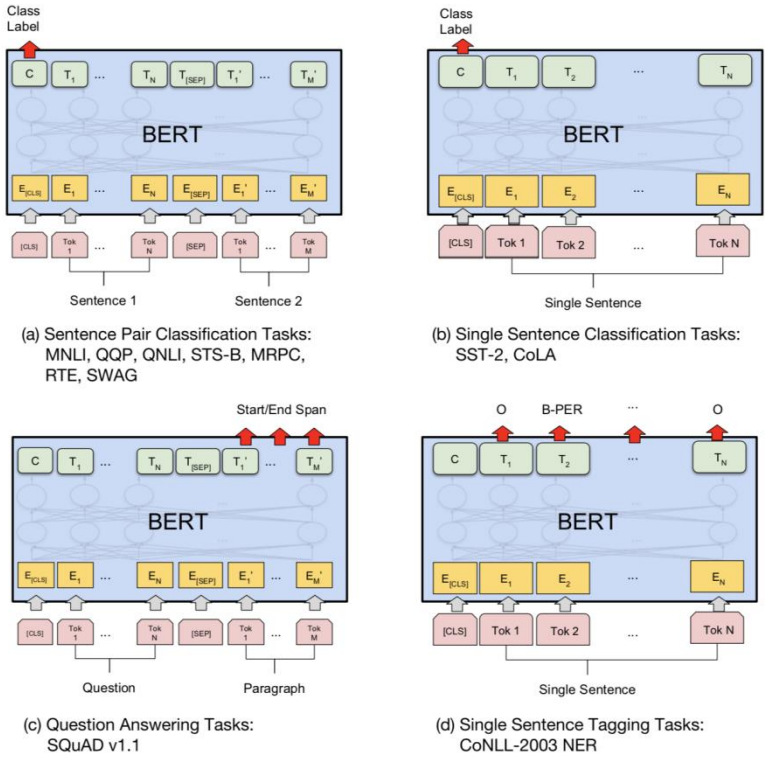
d) BERT：BERT 的全称为 Bidirectional Encoder Representation from Transformers。其采用了 Transformer 的 Encoder 部分，并且进行双向的 Block 连接，与 Transformer 相比，不仅可获取到语句之前的信息，还可获取到未来的信息。



图中的黄色部分为输入的嵌入(Embedding)表示，其由三种不同的 Embedding 求和而成，分别为 Token Embeddings，Segment Embeddings 以及 Position Embeddings。Token Embeddings 是模型输入中每个元素的词向量表示；Segment Embeddings 用来区分输入的前后两个语句；Position Embeddings 含义与 Transformer 相同，但 Transformer 中使用不同频率的三角函数直接计算得出，而 BERT 是通过训练学习得到的。上图中每一个黄色框均代表模型输入的一个元素对应的 Embedding 表示。紧接着将 Embedding 输入到 Transformer 的编码(Encoder)部分，由于 Encoder 没有对 Multi-Head Attention 进行 Mask 操作，这也是 BERT 能获取到双向信息的原因。经过多层串联的 Transformer 后即得到最终的输出，即上图中的蓝色框。

在训练时，BERT 包含两个预训练子任务。第一个子任务为 Masked Language Model，通过随机遮挡每一个句子中 15%的词，用其上下文来做预测，在计算损失函数时只计算被遮挡掉的输入元素。第二个子任务为 Next Sentence Prediction，可以理解简单的分类任务，用以判断两个输入语句是否为前后关系。

使用预训练模型进行微调时，BERT 针对不同的预测任务给予了不同的方法。如下图所示，由此可以看出 BERT 不仅适合做简单的文本分类、序列标注等常见任务，对于问答系统、信息检索、聊天机器人等任务都有着重大的突破。在本实验中，我们完成的是问答系统的任务，对应下图中的(c)部分。



e) BatchMatMulV2 算子：BatchMatMulV2 算子是 BERT 模型中重要的一个操作，其主要进行两个张量相乘的操作，其中这两个输入张量的 shape 可能不一致，因此还涉及到 BroadCast 操作，其还有两个参数可对两个输入张量进行共轭转制操作。

BatchMatMulV2 的输入输出如下：Arguments：

x：2 维或更高维度，输入 shape 为[...,r_x,c_x]

y：2 维或更高维度，输入 shape 为[...,r_y,c_y]

Attrs：adj_x：当为 True 时，对输入 x 进行共轭转制，默认为 False。adj_y：当为 True 时，对输入 y 进行共轭转制，默认为 False。

Return：Output：返回 3 维或更高维度的张量，输出 shape 为[...,r_o,c_o]

其和 BatchMatMul 算子的区别在于 BatchMatMulV2 支持 batch 层面的 broadcasting 操作。

4.实验内容：

a) BERT DLP 运行：将 BERT 网络移植在 DLP 中，包含模型量化，config 设置等操作。

b) BANGC BatchMatMulV2 算子实现：本次实验的重点。

c) 框架集成：将用 BCL 实现的 BatchMatMulV2 算子集成在 CNPlugin 中，通过 CNPlugin 接口集成在 TensorFlow 框架中。

d) 完整框架推理测试：执行 TensorFlow 的在线推理，比较添加 BANGC 算子前后性能差异。

e) 运行 BERT 大算子网络，比较大算子实现与上述实现的性能与精度。

5.实验步骤：

a) 登录云平台：ssh xxx@120.236.247.203 -p xxx

b) 初始化环境：cd /opt/AICSE-demo-student/env; source env.sh

c) cd /opt/AICSE-demo-student/demo/bert/bangc/PluginBatchMatMulV2Op

d) BatchMatMulV2 BANGC 算子实现与测试，分别补全 plugin_batch_matmul_v2_kernel.h 和 plugin_batch_matmul_v2_kernel.mlu 文件。

e) CNPlugin 算子集成，补全 cnplugin.h 并编译新的 Cambricon-CNPlugin。

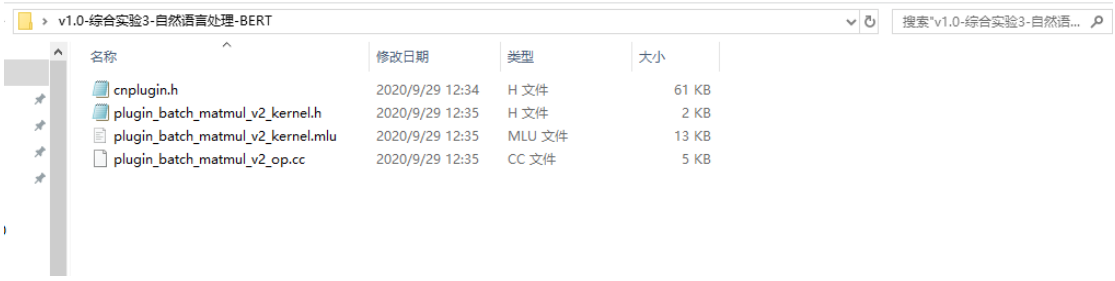
f) TensorFlow 算子集成，将下述文件夹中的文件依次添加到 TensorFlow 源码中（由于课程时间关系，该部分代码直接给出）并重新编译 TensorFlow：/opt/AICSE-demo-student/demo/bert/tf-implementation/tf-add-batchmatmulv2; /opt/AICSE-demo-student/env/ tensorflow-v1.10

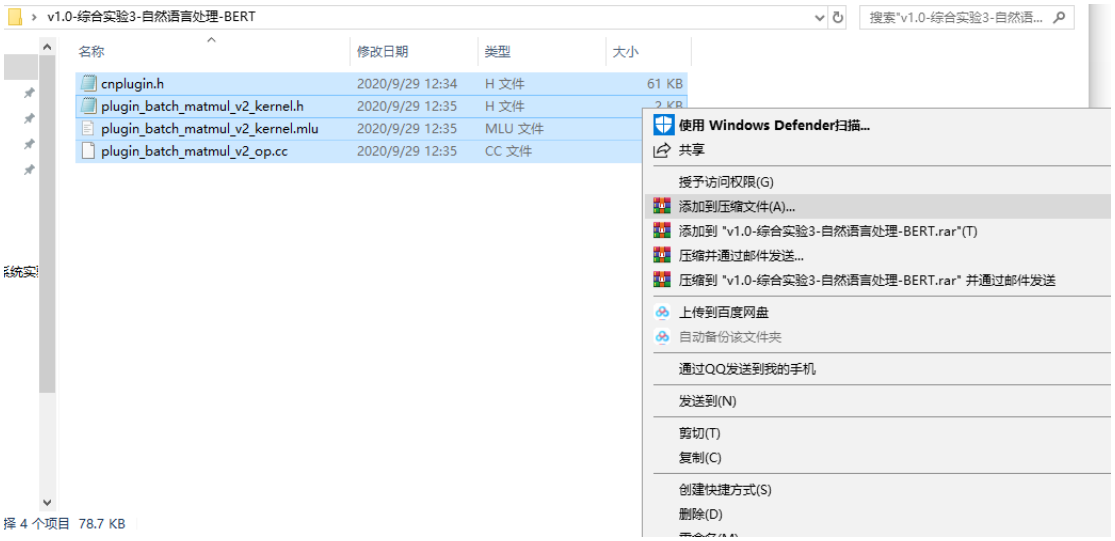
g) 框架推理测试，在提供的 pb 模型中添加合并算子。进入 /opt/AICSE-demo-student/demo/bert/bert-master-batchmatmulv2 文件夹下，执行：run_aicse.sh。

h) 大 算 子 推 理 测 试 ， 进 入 /opt/AICSE-demo-student/demo/bert/bert_fast，执行：run_aicse.sh。

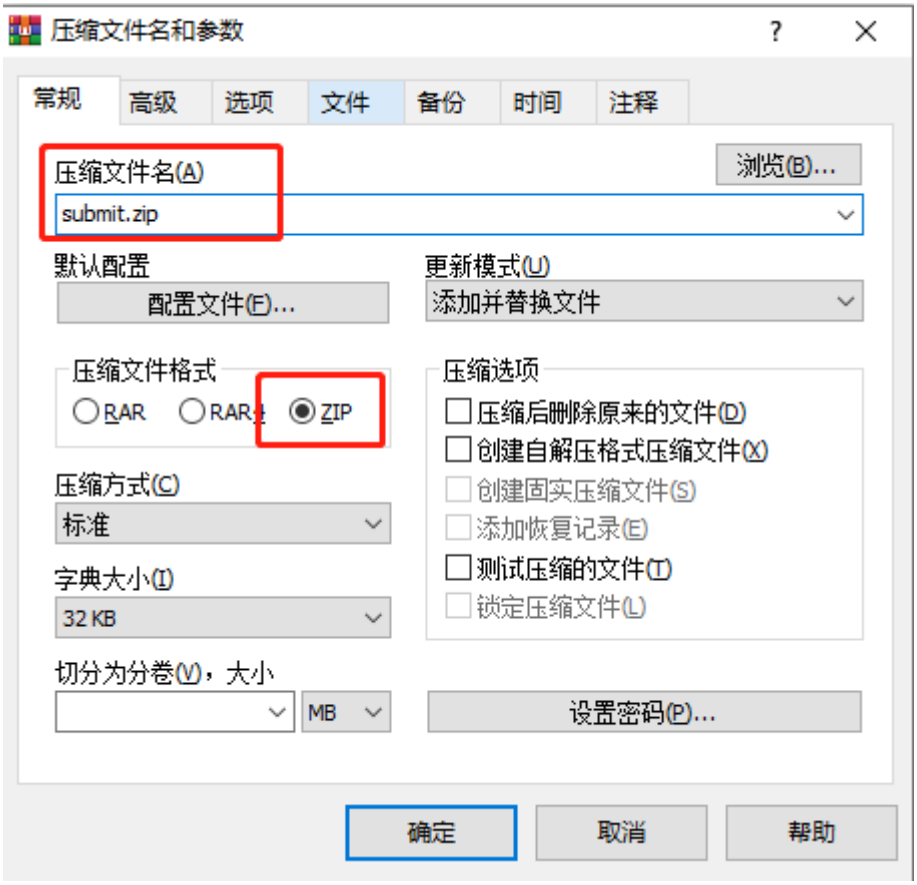
6. 上传自动评测平台

a) 将程序按以下文件列表整理，打包成压缩文件，请选中所有文件直接压缩，不要压缩文件夹。请注意文件名和图中保持一致。

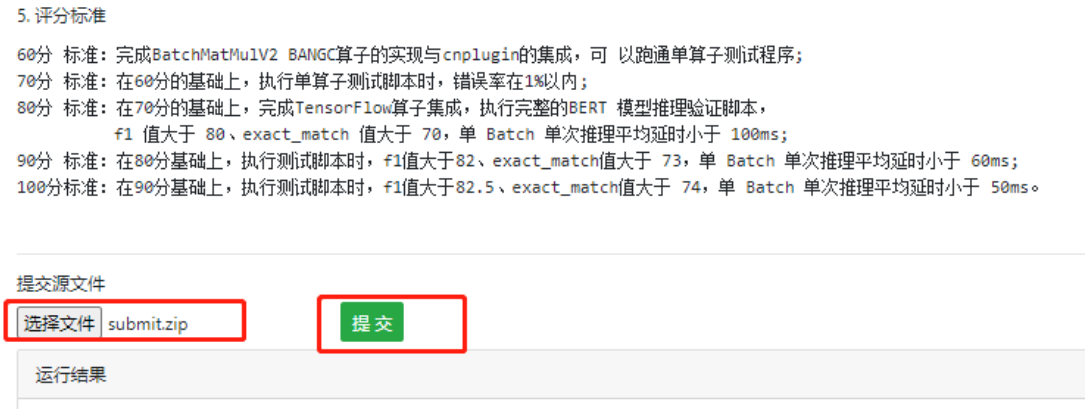




b) 压缩文件格式为 zip 格式。文件名命名为 submit.zip



c） 将 submit.zip 上传到评测平台。选择文件并点击提交。



d） 上传之后请等待系统评测。

e） 评测结束之后会自动显示评测结果。