基于 TPU 平台的 OCR 模型性能优化

识唔识得

摘要

OCR（Optical Character Recognition）技术在许多领域有广泛的应用，如文档数字化、自动数据录入、图书和档案的电子化、票据识别等等。使用 OCR 技术能够能极大减轻人力成本，提升效率。TPU（Tensor Processing Unit），一种专为加速深度学习任务设计的硬件加速器，为 OCR 模型的优化提供了强大的计算支持。本次赛题需要利用具有 TPU 的 Milk-V Duo 超紧凑嵌入式开发平台，实现低精度量化 OCR 模型，落地端侧场景。

应本赛题的要求，我们将开源解决方案PaddleOCR迁移到 Milk-V Duo 开发板上部署运行，并做了性能调优和基准测试。我们的主要工作总结如下：1、调研现有的轻量化开源 OCR 方案，并通过基准测试选择了 PaddleOCR 用于后续部署。2、通过 TPU-mlir 等一系列工具链，将 PaddleOCR 架构和权重进行简化和量化，生成 TPU 上可用的 cvimodel 格式。3、基于 tpu-sdk-cv180x 的相关 API，编写了 cvimodel 的运行时 cvirunner。4、基准测试了多种 det 和 rec 子模型的不同组合，以平衡推理结果和推理时间的需求，确定最佳部署方案和超参数。

解决方案

根据本次赛题的要求，需要开展的工作主要包括以下几方面：首先，我们需要确定跑在 Milk-V Duo 上的 OCR 模型，所以我们前提工作是调研开源的 OCR 模型，通过相应的测试确定合适的模型。然后，我们需要将选取模型进行量化和转换，保证模型能部署在 Milk-V Duo，最后，我们需要重新编写一个运行时，使转换的模型能在 Milk-V Duo 上运行。本项目的解决方案如下图所示。

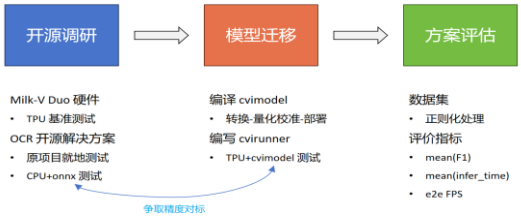

图 1 解决方案

1 相关调研

1.1 硬件调研

本次平台考虑的是一个基于 CV1800B 芯片的 Milk-V Duo 超紧凑嵌入式开发平台，其具有 1GHz 的 RISC-V C906 处理器同时集成了 CVITEK TPU，可用于 AI 应用开发。下面给出了 Milk-V Duo 板上 TPU 功能相关的测试基准，展示了轻量化基础计算机视觉神经网络模型在此设备上的性能表现数据，为后续 ocr 模型选型和性能评估提供了参考。

模型	GFLOPS	运行时间 (s)	FPS
resnet18	1.81	0.30	3.33
densenet121	2.83	0.85	1.18
mobilenetv2	0.30	0.20	5.00
shufflenetv2	0.14	0.16	6.25
squeezenet1.1	0.35	0.16	6.25
googlenet12	1.5	0.24	4.16
yolov5n	4.5	0.55	1.80

图 2 Milk-V Duo 测试基准

1.2 模型调研

在预研阶段，我们分别调研了四种开源 OCR 模型 PaddleOCR、RapidOCR、CnOCR 和 chineseocr_lite，它们各自具备独特的优点，能够满足不同场景的需求。

- PaddleOCR（ppocr）：基于百度 PaddlePaddle 开发，具有高性能和易用性，有开箱即用的优质预训练权重，覆盖了中英文及其他多国语言，以及纵向文本场景。

- RapidOCR： 本质为 PaddleOCR 的 onnx 推理发行版，实测不如原 PaddleOCR 项目。
- CnOCR ： 专注于中文字符识别，提供友好的 Python 接口和模块化设计，非常适合处理复杂中文文本的场景。
- chineseocr_lite ： 是为中文设计的轻量级解决方案，简单易用，适合嵌入式设备，但不支持竖排文本。

我们对上述四个项目做了基准测试，发现其中 ppocr 显著优于其他。进一步地，ppocr 有多个发行版本，均转换到 cpu+onnx 运行环境之后，我们用 A 榜数据集进行了较为全面的基准测试，测试结果如下：

发行板本	det	rec	cls	官方参考 Hmean	A榜 推理时间 (ms)	A榜 F1-Score
v4	DBNet (4.51MB)	SVTR (10.3MB)	mbnetV3 (559KB)	62.24%	76.05	0.60724
v3	DBNet (2.31MB)	SVTR (10.1MB)		57.99% 62.90%	58.68	0.57585
v2	DBNet (2.22MB)	CRNN (7.99MB)		57.60%	43.02	0.52051
mobile	DBNet (2.22MB)	CRNN (4.22MB)			41.61	0.34883

图 3 ppocr 基准测试

从图 3 的结果可以看出，v4 版本的 f1-score 是最好，但是推理时间较长且模型较大，可能会导致板上运行时爆内存（实际上后面图 5 也可以看到 v4 也确实导致内存溢出）。v3 和 v2 两者的分数和推理时间相近，是可以着重考虑的，mobile 版本的 f1-score 下降太快，但是和 v2 的推理时间相差不大。因此综合来看，考虑到 Milk-V Duo 的资源情况，我们优先考虑 v2 系列，但也会保持对其他系列的兼容支持。

2 模型转换和运行时编写

2.1 模型转换

为了将 ppocr 模型转换为优化的 cvimodel 格式，需要经过多个步骤，即 paddle→onnx→mlir→cvimodel。首先，需要将 ppocr 原始模型转换为 onnx，在这里我们做了一点优化，即移除了模型末尾的 Softmax 层，因为部署时推理结果只需要 argmax 即可。在得到简化的 onnx 模型后，用 TPU-mlir 工具箱将其转换为 mlir 中间格式。随后基于校准数据集生成量化校准表后，即可进一步编译为最后的目标 cvimodel 格式。在此过程中，须确保正确设置模型输入张量的均值(mean)、尺度(scale)以及形状信息。在量化校准时，需要提前对 rec 数据集进行纯白色右端填充，以更加符合推理时的填充策略，否则样本会被过度横向拉伸，导致校准分布偏差。图像的预处理被融合入模型操作中，模型的输出也保持量化状态传出，即在 TPU 数据传输时尽量使用 uint8/int8 而非 f32，以大幅节省数据传输量。图 4 给出了转换、校准和编译所用相关的工具命令，及重要参数设定：

```
model_transform.py \
  --model_name $MODEL_NAME \
  --model_def $MODEL_DEF \
  --input_shapes $INPUT_SHAPE \
  --mean $MEAN \
  --scale $SCALE \
  --keep_aspect_ratio \
  --test_input $TEST_INPUT \
  --test_result $TEST_RESULT \
  --debug \
  --mlir $MLIR_MODEL_FILE
```

```
run_calibration.py $MLIR_MODEL_FILE \
  --dataset $CALI_DATASET \
  --input_num 300 \
  -o $CALI_TABLE_FILE
```

```
model_deploy.py \
  --chip $CHIP \
  --mlir $MLIR_MODEL_FILE \
  --quantize $QTYPE \
  --quant_input \
  $QUANT_OUTPUT \
  $QUANTIZE_TABLE \
  --calibration_table $CALI_TABLE_FILE \
  --test_input $TEST_INPUT \
  --test_reference $TEST_RESULT \
  --tolerance 0.85,0.45 \
  $COMPARE_ALL \
  --fuse_preprocess \
  --customization_format BGR_PACKED \
  --ignore_f16_overflow \
  --op_divide \
  --debug \
  --model $CVI_MODEL_FILE
```

图 4 模型转换相关参数设定

子模型	编译预设	GFLOPS (mlir)	ION 内存需求 (MB)	板上推理时间 (ms)
det	v4_det_int8	4.904	22.26	OOM
	v3_det_int8	3.918	14.10	270.805
	v2_det_int8	3.906	13.25	247.386
	mb_det_int8	3.906	13.25	223.39
	v2_det_int8_480	-	6.90	128.963
	v2_det_int8_320	-	3.72	46.317
rec	v3_rec_bf16	0. 942	10.07	95.523
	v2_rec_bf16	1.130	12.42	65.314
	mb_rec_bf16	0.286	7.93	33.612
cls	mb_cls_int8	0.119	0.77	-

图 5 cvimodel 的基本信息

图 5 展示了最终编译所得 cvimodel 的规格信息。可以看到，在 det 模型中，v4_det_int8 直接内存溢出，无法推理，遂舍弃。v3_det_int8、v2_det_int8 以及 mb_det_int8 的内存占用相近，但是 v2_det_int8 的推理时间相对 v3_det_int8 更短，mb_det_int8 的推理时间更短，但是后面可以看到 mb_det_int8 的推理效果较差，所以对于 det 我们着重考虑 v2。在 rec 模型中，mb_rec 的推理时间最短，作为重点考虑对象。注意到，实验中我们发现 rec 模型几乎无法进行 int8 量化，即使手工设定混合量化表，性能收益也不大，因此 rec 模型全量保持了 bf16 的量化数据类型。

2.2 运行时开发

在板上运行推理过程，即需要一个程序来加载并使用已经编译好的 cvimodel 文件，这就是模型的运行时(cvirunner)。我们基于 cviruntime (milkv-duo/tpu-sdk-cv180x) 来实现 MilkV-Duo 上推理 ppocr 的 cvimodel 的运行时，其代码逻辑框架如图所示。

```
Model det = load_model(), rec = load_model();

for (string &file : file_list) {
  Mat img = imread(file);

  Mat img = det.preprocess(img);
  int8 *segmap = det.infer(img);
  vector<Box> box_list = det.postprocess(segmap);

  for (Box &box : box_list) {
    Mat img_crop = warp_crop_perspective(img, box);

    Mat img_crop = rec.preprocess(img_crop);
    float32 *logits = rec.infer(img_crop);
    int[] token_ids = rec.postprocess(logits);
  }
}

unload_model(det); unload_model(rec);
```

图 6 运行时代码逻辑

需要注意的细节有：

- 在 det 前处理部分，我们需要将输入图像调整到目标尺寸，在 Resize 之后，为保持图像比例，在图像边缘添加 0 值填充。在 det 后处理部分，首先对模型输出应用二值化，然后使用轮廓检测算法提取二值化后图像中的目标轮廓，最后对检测到的文本轮廓进行 unclip。
- 在 rec 前处理部分，调整输入图像以适应识别模型的输入要求，同时保持长宽比不变。在图像边缘填充值 255 的像素，以补足到目标尺寸。对模型输出进行最大值索引查找，通常用于获取预测文本的索引表示。

在编写运行时中，对于分割图到检测框这一步转换，团队给出了一种 unclip 算法的近似实现。考虑一个面积 wh 的矩形框，要让其面积增加 k 倍，那么需要将其边长增加

$$r = \frac{1}{4}\left[\sqrt{(w+h)(w+h)+4(k-1)wh} - (w+h)\right]$$

这样，我们只需要将端点向外延伸$\sqrt{2}r$即可，如下图所示

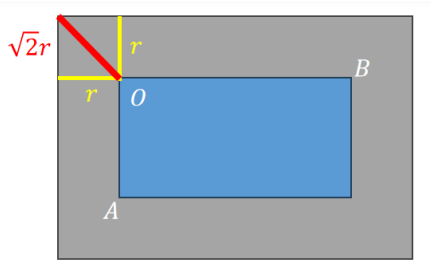

图 7 unclip 算法的近似实现

这样做的意义是不再需要加载额外的库来实现 unclip ，只需要进行一个简化的计算即可，从而减少端到端运行时间。

3 方案评估

首先是数据集，主办方提供的数据集有：A 榜 ICDAR2019-LVST，2350 个样本；B1 榜 MSRA-TD500，500 个样本；B2 榜：3992 个样本（对于尺寸超过 640 的图预先降采样处理，否则磁盘空间不够无法上传）。其次是评估指标：mean(F1)以文本框为单位，box_iou>0.5 且 text_sim>0.5 记为一个 TP；mean(infer_time)平均一次 det + 一次 rec 的纯 TPU 推理时间；real_fps 不计数据读取时间的端到端 FPS。在本项目中，我们也提供了多种模型组合进行比较和参考 (我们还支持更多模型组合，读者可以自行测试)。对 A 榜数据集，板上测试结果如下：

数据集	模型配置	F1	infer_time	e2e FPS	estimate/submit score
A榜	v2_det + v2_rec	0.44099	333.937	0.88	79.25498
	v3_det + mb_rec	0.42010	277.942	1.22	83.17896
	v2_det + mb_rec	0.42781	256.211	1.42	85.33433
	v2_det + mb_rec (480)	0.33901	155.279	1.885	90.36170
	v2_det + mb_rec (320)	0.20613	75.951	2.954	91.78934
	mb_det + mb_rec	0.32475	256.930	1.47	81.15095

图 8 A 榜数据板上测试

从图中可以看出：v2_det+v2_rec 是能够获得最高的 F1-score，但是推理时间较长；v2_det+mb_rec(320)能够获得最佳的推理时间，但是 F1 分数较低；综合来看，v2_det+mb_rec 和 v2_det+mb_rec（480）这两种组合是较好的，我们将主要关注这两个方案在 B 榜的数据上的表现，B 榜测试结果如图 8 所示：

B1榜	v2_det + mb_rec	-	313.969	0.42	-
	v2_det + mb_rec (480)	-	198.213	0.52	-
	v2_det + mb_rec (320)	-	115.033	0.58	-
B2榜	v2_det + mb_rec	0.44719740	255.359	1.96	99.38789602
	v2_det + mb_rec (480)	-	152.944	2.70	-

图 9 B 榜数据板上测试

以 F1 为优先考虑，实际我们 B 榜最终提交的模型配置即 v2_det+mb_rec；但若进一步考虑降低推理时间，可以尝试输入尺寸为 480 的版本，预计综合评分约会提高 5 分，并取得一个较好的 F1-score 和推理时间的平衡。(网站显示我们的最终分数 B 榜分数为 99.38789602，但我们不清楚这个数字是什么公式计算出来的，毕竟无法上传 time_infer 信息。)

在测试过程，我们也发现了目前的模型方案在不同的场景下呈现了不同的效果。下面给出一些典型的推理结果的样例，供读者参考。

1、密集印刷字体：从图中可以看到所有的文字、字母和数字几乎完全正确识别，说明v2_det+mb_rec的方案组合在较标准的字体下，能够正确检测。

图 10 密集印刷字体

2、密集印刷字体+纵向排列的文字：对于印刷字体，可以看到正确率仍然较高，同时对于纵向排列的也可以识别。但是对于下方的奇怪的字体时，识别则出现了错误。

图 11 密集印刷字体+纵向排列

从图 10 看到了对于非标准的印刷字体的识别，出现了错误，下面我们对考察一些非标准的字体的场景。

3、手写字体：对于手写字体，检测框是的基本正确的，但是识别部分效果较差，可能原因为 rec 模型的识别能力下降。

图 12 手写字体

4、艺术字体：和上面是同样的问题，但是这张图中，检测框是不完整的，同时识别也是较差的。可能是由于光源导致的 det 识别不完整，同时 rec 对非标准字体的识别能力较弱。

图 13 艺术字体

4 总结与讨论

本赛题，我们基于 ppocr 系列移植了一套能运行在 Milk-V Duo (CV1800B) 板上的模型，且支持任意组件混搭。针对 Milk-V Duo 的板上资源情况，优化了 cvimodel 编译参数，同时优化了数据校准方式。此外，在开发运行时过程中，我们还提出了一种 unclip 算法的近似实现，编写了一套高效的 cvimodel 运行时。在开发的过程中，我们也发现了迁移模型过程中，具有较大的精度损失，从 ppocr 的 cpu 版本迁移到 tpu 上的 cvimodel，F1-score 下降了 0.2，其中的原因可能是因为在模型转换过程中的前后处理简化和量化过后的低召回率导致 (实际上对于嵌入式设备，我们不得不主动调低召回率来换取更高的实时性)。同时对于非标准字体，识别效果出现了较大的下降。这些可以作为未来进一步优化的方向。

致谢

感谢主办方相关人工作人员对于选手参赛的支持，感谢那块小小的 Milk-V Duo，辛苦了。

参考

[1] Du, Yuning, et al. "Pp-ocrv2: Bag of tricks for ultra lightweight ocr system." arXiv preprint arXiv:2109.03144 (2021).
[2] Hu, Pengchao, et al. "TPU-MLIR: A Compiler For TPU Using MLIR." arXiv preprint arXiv:2210.15016 (2022).
[3] Open neural network exchange.https://github.com/onnx/onnx.
[4] https://github.com/RapidAI/RapidOCR.
[5] https://cnocr.readthedocs.io/zh-cn/stable/.
[6] https://github.com/DayBreak-u/chineseocr_lite.
[7] https://github.com/milkv-duo/tpu-sdk-cv180x.
[8] https://milkv.io/zh/docs/duo/overview.