1. **现状**

TensileConv 经过一年（中间有数月被暂停）的开发，现完成0.1.0版本

代码公开在<https://github.com/feifei14119/conv>

相关设计在申请专利中

1. **特性**

支持gfx803和gfx900硬件

支持bias的融合操作

支持relu, prelu的融合操作

1. **使用方法**

TensileConv提供两种使用方法

1. 静态库

使用静态库，用户可以将TensileConv集成到自己的程序中，在程序运行时随时寻找最优kernel

用户需要链接out目录下的libTensileConv.a库文件，包含Include目录下的TensileConv.h头文件，然后实例化DirConv1x1Fwd类并调用AutoTune方法即可，参考代码如下：

cid:image005.png@01D4B823.6E343B70

TensileConv提供了一个调用Lib库的示例工程，在LibTest目录下。

1. 独立工具

使用TensileConv编译出的可执行文件TensileConv.out，用户可以在程序部署阶段，预先搜寻最优kernel

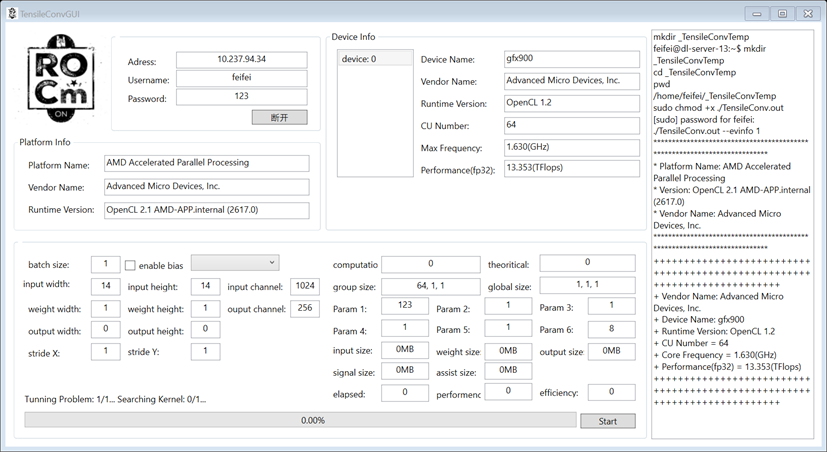
TensileConv.out通过命令行方式，可以指定输入NCHWK尺寸、Bias/Relu操作等，具体可通过TensileConv.out –help查看帮助

参考命令如下：

./TensileConv -s 14 -c 1024 -k 512 -n 1 –bias 1 –relu 0

              3. GUI

                       作为娱乐项目, TensileConv提供了一个Windows的远程图形界面，在out目录下的TensileConvGui.exe文件

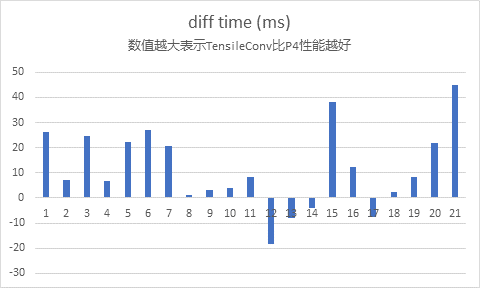


                     注意：该GUI仅作娱乐，不保证功能完备且不保证后继维护。

1. **性能**

**4.1 TensileConv在Fiji上的性能与P4性能对比 (batch size = 1)**：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | W | H | C | K | P4 | Tensile | Diff | Efficiency |
| **1** | 7 | 7 | 512 | 2048 | 83.6218 | 57.2 | 26.4218 | 20.90% |
| **2** | 7 | 7 | 1024 | 512 | 37.025 | 29.7 | 7.325 | 20.10% |
| **3** | 7 | 7 | 1024 | 1024 | 86.1411 | 61.4 | 24.7411 | 19.50% |
| **4** | 7 | 7 | 2048 | 512 | 69.4365 | 62.6 | 6.8365 | 19.10% |
| **5** | 7 | 7 | 2048 | 1024 | 137.124 | 114.8 | 22.324 | 20.80% |
| **6** | 7 | 7 | 2048 | 2048 | 245.688 | 218.5 | 27.188 | 21.90% |
| **7** | 14 | 14 | 256 | 1024 | 59.4628 | 38.8 | 20.6628 | 30.80% |
| **8** | 14 | 14 | 512 | 256 | 23.4109 | 22.4 | 1.0109 | 26.70% |
| **9** | 14 | 14 | 512 | 512 | 41.5939 | 38.6 | 2.9939 | 31.00% |
| **10** | 14 | 14 | 512 | 1024 | 78.3258 | 74.2 | 4.1258 | 32.20% |
| **11** | 14 | 14 | 1024 | 256 | 47.6812 | 39.3 | 8.3812 | 30.40% |
| **12** | 14 | 14 | 1024 | 512 | 61.8133 | 80 | -18.1867 | 29.90% |
| **13** | 14 | 14 | 1024 | 1024 | 141.149 | 149.3 | -8.151 | 32% |
| **14** | 14 | 14 | 1024 | 2048 | 263.045 | 267.1 | -4.055 | 35.80% |
| **15** | 28 | 28 | 128 | 512 | 68.8487 | 30.5 | 38.3487 | 39.20% |
| **16** | 28 | 28 | 512 | 128 | 45.2395 | 32.9 | 12.3395 | 36.30% |
| **17** | 28 | 28 | 512 | 256 | 56.222 | 63.9 | -7.678 | 37.40% |
| **18** | 28 | 28 | 512 | 512 | 131.123 | 128.9 | 2.223 | 37.10% |
| **19** | 28 | 28 | 512 | 1024 | 249.129 | 240.9 | 8.229 | 39.70% |
| **20** | 56 | 56 | 256 | 128 | 76.4928 | 54.5 | 21.9928 | 43.80% |
| **21** | 56 | 56 | 256 | 256 | 145.869 | 101.1 | 44.769 | 47.30% |
| **22** | 56 | 56 | 256 | 512 | 282.918 | 191.7 | 91.218 | 49.90% |

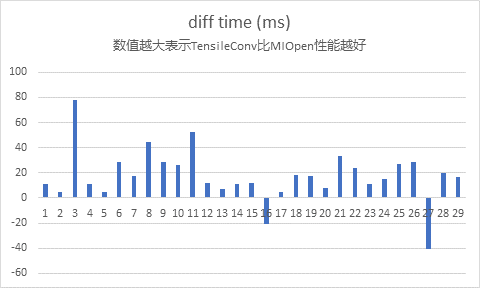


注：P4数据来源于台湾团队在阿纳金项目中实测数据，参见如下：

<http://myteamsna.amd.com/sites/DCGPUGC/_layouts/15/xlviewer.aspx?id=/sites/DCGPUGC/Baidu/Anakin/Project%20material/Anakin%20Kernels%20Perf_NV_AMD_update_mi8.xlsx&Source=http%3A%2F%2Fmyteamsna%2Eamd%2Ecom%2Fsites%2FDCGPUGC%2FBaidu%2FForms%2FAllItems%2Easpx%3FRootFolder%3D%252Fsites%252FDCGPUGC%252FBaidu%252FAnakin%252FProject%2520material%26FolderCTID%3D0x01200000CC020AFC77E9449AA01D366EEA853F%26View%3D%257BE407D3D7%252D81D1%252D49E1%252D9E01%252D7CF5BEF30544%257D>

**4.2 TensileConv在Vega64上的性能与MIOpen性能对比：**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | W | H | C | K | N | MIOpen | Tensile | Diff | Efficiency |
| **1** | 7 | 7 | 512 | 2048 | 1 | 48.347 | 37.3 | 11.047 | 20.70% |
| **2** | 7 | 7 | 512 | 2048 | 4 | 83.525 | 79 | 4.525 | 39% |
| **3** | 7 | 7 | 512 | 2048 | 32 | 535.293 | 457.7 | 77.593 | 53.80% |
| **4** | 7 | 7 | 1024 | 512 | 1 | 34.684 | 23.2 | 11.484 | 16.60% |
| **5** | 7 | 7 | 1024 | 512 | 4 | 49.829 | 44.8 | 5.029 | 34.40% |
| **6** | 7 | 7 | 1024 | 512 | 32 | 290.18 | 261.6 | 28.58 | 47.10% |
| **7** | 7 | 7 | 2048 | 512 | 1 | 62.256 | 44.9 | 17.356 | 17.10% |
| **8** | 7 | 7 | 2048 | 512 | 8 | 186.276 | 141.4 | 44.876 | 43.60% |
| **9** | 7 | 7 | 2048 | 512 | 16 | 293.917 | 264.9 | 29.017 | 46.50% |
| **10** | 14 | 14 | 256 | 1024 | 8 | 138.193 | 111.6 | 26.593 | 55.20% |
| **11** | 14 | 14 | 256 | 1024 | 16 | 261.466 | 208.7 | 52.766 | 59% |
| **12** | 14 | 14 | 512 | 512 | 4 | 76.726 | 65.1 | 11.626 | 47.30% |
| **13** | 14 | 14 | 512 | 512 | 8 | 126.325 | 119 | 7.325 | 51.80% |
| **14** | 14 | 14 | 1024 | 256 | 1 | 35.227 | 24.2 | 11.027 | 31.80% |
| **15** | 14 | 14 | 1024 | 256 | 4 | 77.681 | 65.9 | 11.781 | 46.70% |
| **16** | 14 | 14 | 1024 | 256 | 16 | 226.666 | 247.4 | -20.734 | 49% |
| **17** | 28 | 28 | 128 | 512 | 1 | 23.904 | 19.1 | 4.804 | 40.30% |
| **18** | 28 | 28 | 128 | 512 | 4 | 68.837 | 50.4 | 18.437 | 61.30% |
| **19** | 28 | 28 | 128 | 512 | 8 | 123.274 | 105.9 | 17.374 | 58.10% |
| **20** | 28 | 28 | 128 | 1024 | 1 | 40.493 | 32.3 | 8.193 | 47.60% |
| **21** | 28 | 28 | 128 | 1024 | 4 | 126.687 | 93.3 | 33.387 | 66% |
| **22** | 28 | 28 | 128 | 1024 | 8 | 228.53 | 204.8 | 23.73 | 60.10% |
| **23** | 28 | 28 | 512 | 256 | 1 | 42.575 | 31.4 | 11.175 | 49.00% |
| **24** | 28 | 28 | 512 | 256 | 4 | 125.747 | 110.7 | 15.047 | 55.60% |
| **25** | 28 | 28 | 512 | 256 | 8 | 238.302 | 211 | 27.302 | 58.30% |
| **26** | 28 | 28 | 512 | 256 | 16 | 463.727 | 435.3 | 28.427 | 56.60% |
| **27** | 28 | 28 | 512 | 256 | 32 | 857.292 | 897.6 | -40.308 | 54.90% |
| **28** | 28 | 28 | 512 | 1024 | 1 | 145.248 | 125.5 | 19.748 | 49.10% |
| **29** | 28 | 28 | 512 | 1024 | 4 | 445.262 | 428.9 | 16.362 | 57.40% |



注：MIOpen为1月25日在github上抓取的MIOpen 1.7.0的最新代码

测试时锁定Vega64为DPM7并且强制风扇转速最大，实时监测显卡温度未超过35℃

使用MIOpenDriver测试时设置了MIOPEN\_FIND\_ENFORCE=4环境变量，并且使用了-s 1选项

**4.3 TensileConv在Anakin中与MIOpen以及原Anakin的kernel性能对比：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| param | nv-miopen | nv- dachuan | nv-tensile | **Best solution** |
| 768 512 1 1 1 1 1 1 0 0 1 | -0.315362 | -0.22034 | 0.287435 | tensile |
| 1024 1000 1 1 1 1 1 1 0 0 1 | -0.3105814 | -0.2674344 | 0.0135796 | tensile |
| 1024 1000 1 1 1 1 1 1 0 0 1 | -0.3089184 | -0.2596804 | 0.0479306 | tensile |
| 1024 1000 1 1 1 1 1 1 0 0 1 | -0.3081782 | -0.2690912 | 0.0137108 | tensile |
| 2048 2048 1 1 1 1 1 1 0 0 1 | -0.303889 | -0.388155 | -0.034415 | tensile |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.3016918 | -0.3194388 | 0.0113332 | tensile |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.3014422 | -0.3211642 | 0.0542538 | tensile |
| 1024 1000 1 1 1 1 1 1 0 0 1 | -0.2975638 | -0.4664488 | 0.0463562 | tensile |
| 2048 2048 1 1 1 1 1 1 0 0 1 | -0.288166 | -0.409714 | -0.034643 | tensile |
| 2048 2048 1 1 1 1 1 1 0 0 1 | -0.28645 | -0.395691 | -0.034111 | tensile |
| 2048 2048 1 1 1 1 1 1 0 0 1 | -0.285356 | -0.397734 | -0.034472 | tensile |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.2811776 | -0.3170886 | 0.0113684 | tensile |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.2724054 | -0.5664234 | 0.0737226 | tensile |
| 2048 2048 1 1 1 1 1 1 0 0 1 | -0.265331 | -0.482952 | -0.034494 | tensile |
| 2048 2048 1 1 1 1 1 1 0 0 1 | -0.236997 | -0.676727 | -0.011952 | tensile |
| 2048 1024 1 1 1 1 1 1 0 0 1 | -0.2107828 | -0.1799308 | -0.0035448 | tensile |
| 1024 1000 1 1 1 1 1 1 0 0 1 | -0.2047912 | 0.0365929 | 0.0146868 |  |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.2004264 | 0.0403951 | 0.0103476 | tensile |
| 2048 1024 1 1 1 1 1 1 0 0 1 | -0.19829 | -0.172868 | -0.003052 | tensile |
| 2048 1024 1 1 1 1 1 1 0 0 1 | -0.198242 | -0.18003 | -0.002764 | tensile |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.1887624 | 0.0436928 | 0.0110836 |  |
| 1280 1000 1 1 1 1 1 1 0 0 1 | -0.181972 | 0.0414336 | 0.011026 |  |
| 1024 108 1 1 1 1 1 1 0 0 1 | -0.1684018 | -0.1536408 | 0.0335722 | tensile |
| 512 1024 1 1 2 2 1 1 0 0 1 | -0.157462 | -0.0282666 | 0.020401 |  |
| 512 1024 1 1 2 2 1 1 0 0 1 | -0.1573244 | -0.6265344 | 0.0276426 | tensile |
| 1024 512 1 1 2 2 1 1 0 0 1 | -0.1537212 | -0.0313184 | 0.0417098 |  |
| 512 1024 1 1 2 2 1 1 0 0 1 | -0.14991 | -0.602559 | 0.032065 | tensile |
| 512 1024 1 1 2 2 1 1 0 0 1 | -0.145687 | -0.609699 | 0.044816 | tensile |
| 1024 512 1 1 2 2 1 1 0 0 1 | -0.1435292 | -0.0614962 | 0.0149268 | tensile |

注：数据来源自KK的2019.02.12测试结果

1. **后继工作**

后继如继续开发，会支持更多硬件（Vega20/MI100），更多精度 (fp16)，更多weight尺寸 (1x3 5x1等)，

并尝试更多优化方法

1. **致谢**

感谢杨博在技术上从头至尾的支持，包括但不限于kernel原型编写、workload划分、datapath设计、prefetch设计

感谢Gary Fu在神经网络以及卷积操作上的扫盲

感谢Jessie Peng 在硬件问题上的帮助

感谢黄亮在代码重构时给出的帮助和建议