CS5222 Advanced Architectures

Project 2 Part 3 Report

FPGA Lab - Optimization

Nguyen Dang Phuc Nhat (A0184583U)

Table of Contents

1 Classifier Optimization

The Ridge used in the default implementation is limited in terms of accuracy after quantization. With much effort into optimizing parameters, the misclassification error is still 18%.

Therefore, I followed my motto “When in doubt, pick SGD”.

I chose to use a `SGDClassifier`. The `loss` function chosen is **`‘log’`**, `penalty` type is **`‘l1’`**, and `learning_rate` is **`‘optimal’`** (which means the algorithm will figure the best learning_rate for us). With this, classification error is around 11-12%. There would be some fluctuation between training runs.

The best thing about this classifier is that we can apply the proper quantization formula:
`scale = (2^n - 1) / (2 * max(abs(values)))`

Accuracy barely changes after `int8` quantization.

2 Input Image Resizing

With the classifier optimized, it is time to optimize throughput.

The easiest way is to resize the image. The default image dimension is 16x16. Thus, it is natural to scale it down to 8x8 (we love powers of 2). It turned out nicely as misclassification rate is now around 11-15%. The fluctuation is stronger, but we can always retrain. After all, the classifier is a straightforward linear model. Training time is very little.

With the image resized, the input buffer `in_buf` is reduced by 4 times in size. The number of weights in `weight_buf` is also reduced by 4 times. Therefore, we have more room to partition these arrays. The number of partitions is increased to 64. With this, HLS latency is reduced from 443495 to 246518 cycles.

3 Hardware Statistics

As mentioned above, the latency after HLS is 246518 cycles.

Latency		Interval		Pipeline
min	max	min	max	Type
246518	246518	246519	246519	none

All loops that can be pipelined are pipelined. All pipelines have II of 1.

	Latency		Iteration	Initiation Interval		Trip	
Loop Name	min	max	Latency	achieved	target	Count	Pipelined
|- LOAD_OFF_1	5	5	2	1	1	5	yes
|- LOAD_W_1	110	110	11	-	-	10	no
| + LOAD_W_2	8	8	2	1	1	8	yes
|- LT	246399	246399	3850	-	-	64	no
| + LOAD_I_1	1407	1407	11	-	-	128	no
| ++ LOAD_I_2	8	8	2	1	1	8	yes
| + L1_L2	1286	1286	8	1	1	1280	yes
| + STORE_O_1	1152	1152	9	-	-	128	no
| ++ STORE_O_2	6	6	3	1	1	5	yes

Compared to the final version in part 2, the biggest saving comes from loading. With 4 times less inputs and 4 times less in weights count, loading is extremely fast.

Nonetheless, the greatest benefit is in the hardware utilization.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	33	-	-
Expression	-	-	0	1563
FIFO	-	-	-	-
Instance	0	0	2144	2272
Memory	70	-	1024	128
Multiplexer	-	-	-	2534
Register	-	-	2512	64
Total	70	33	5680	6561
Available	280	220	106400	53200
Utilization (%)	25	15	5	12

Not even 25% of the hardware resource is needed. Now this is time to test out the design on the actual FPGA board.

4 FPGA Inference

On the FPGA board, the validation error is only 12.99%. Though speedup compared to CPU is 21x, it is only because there are much fewer computations (even the CPU is fast).

The concrete measurement of runtime is 0.0035s on average, or 3.5ms.

In [5]:
```
# Evaluate validation accuracy
cpu_errors = 0
fpga_errors = 0
for idx in range(BATCH):
    fpga_label = np.argmax(c[idx])
    cpu_label = np.argmax(c_ref[idx])
    actual_label = l[idx] # np.argmax(l[idx])
    if (fpga_label!=actual_label):
        fpga_errors += 1.
    if (cpu_label!=actual_label):
        cpu_errors += 1.

# Report results
print("FPGA accuracy: {0:.2f}% validation error".format(fpga_errors/BATCH*100))
print("CPU accuracy:  {0:.2f}% validation error".format(cpu_errors/BATCH*100))
print("FPGA time: {}".format(fpga_time))
if (cpu_time < fpga_time):
    print("FPGA has a {0:.2f}x slowdown".format(fpga_time/cpu_time))
else:
    print("FPGA has a {0:.2f}x speedup".format(cpu_time/fpga_time))
```

```
FPGA accuracy: 12.99% validation error
CPU accuracy:  12.99% validation error
FPGA time: 0.003409587000078318
FPGA has a 21.84x speedup
```

Both accuracy and runtime (85% and 4.5ms) goals have been met.