1. finish the three kernel codes in "cuda_functions.cu" WITHOUT shared memory.

Source code: 1.cu

```
comp3231s10@comp3231s10:~/A2$ make all
/usr/local/cuda-9.2/bin/nvcc -I/include -I. -lineinfo -arch=sm_35 --ptxas-
options=-v --use_fast_math -o cuda_functions.o -c cuda_functions.cu
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z14layer1_sigmoidPfPh' for 'sm_35'
ptxas info    : Function properties for _Z14layer1_sigmoidPfPh
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 6 registers, 336 bytes cmem[0]
ptxas info    : Compiling entry function '_Z19layer1_feature_mapsPfPhS_' for
'sm_35'
ptxas info    : Function properties for _Z19layer1_feature_mapsPfPhS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 16 registers, 344 bytes cmem[0]
ptxas info    : Compiling entry function '_Z16layer1_init_biasPfS_' for 'sm_35'
ptxas info    : Function properties for _Z16layer1_init_biasPfS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 7 registers, 336 bytes cmem[0]
/usr/local/cuda-9.2/bin/nvcc -o cnn conv_layer4.o utils.o cuda_functions.o -L/
lib64 -lcudart

comp3231s10@comp3231s10:~/A2$ make run
./cnn
CPU Elapsed time is 6.690272 s
number of detections = 8
detection nr 0 = 30 km/h, box pos= x 1072, y 304, confidence = 71
detection nr 1 = 30 km/h, box pos= x 1068, y 308, confidence = 96
detection nr 2 = 30 km/h, box pos= x 1072, y 308, confidence = 98
detection nr 3 = 50 km/h, box pos= x 312, y 420, confidence = 97
detection nr 4 = 50 km/h, box pos= x 316, y 420, confidence = 74
detection nr 5 = 50 km/h, box pos= x 312, y 424, confidence = 98
detection nr 6 = 50 km/h, box pos= x 316, y 424, confidence = 88
detection nr 7 = 50 km/h, box pos= x 704, y 432, confidence = 57

 ######## GPU running... ########
cudaMalloc success.
Time to execute layer1_feature_maps:  3.4 ms
GPU Elapsed time is  6.1 ms
Checking GPU results of layer 1 ...
GPU layer 1 passed.
```

Time to execute layer1_feature_maps (in ms)										Fastest
3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4

GPU Elapsed time (in ms)										Fastest
6.2	6.2	6.1	6.2	6.1	6.2	6.2	6.3	6.3	6.3	6.1

2. use shared memory to cache the input (i.e. the image).

Source code: 2.cu

```
comp3231s10@comp3231s10:~/A2$ make all
/usr/local/cuda-9.2/bin/nvcc -I/include -I. -lineinfo -arch=sm_35 --ptxas-
options=-v --use_fast_math -o cuda_functions.o -c cuda_functions.cu
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z14layer1_sigmoidPfPh' for 'sm_35'
ptxas info    : Function properties for _Z14layer1_sigmoidPfPh
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 6 registers, 336 bytes cmem[0]
ptxas info    : Compiling entry function '_Z19layer1_feature_mapsPfPhS_' for
'sm_35'
ptxas info    : Function properties for _Z19layer1_feature_mapsPfPhS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 22 registers, 1348 bytes smem, 344 bytes cmem[0]
ptxas info    : Compiling entry function '_Z16layer1_init_biasPfS_' for 'sm_35'
ptxas info    : Function properties for _Z16layer1_init_biasPfS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 7 registers, 336 bytes cmem[0]
/usr/local/cuda-9.2/bin/nvcc -o cnn conv_layer4.o utils.o cuda_functions.o -L/
lib64 -lcudart

comp3231s10@comp3231s10:~/A2$ make run
./cnn
CPU Elapsed time is 6.680574 s
number of detections = 8
detection nr 0 = 30 km/h, box pos= x 1072, y 304, confidence = 71
detection nr 1 = 30 km/h, box pos= x 1068, y 308, confidence = 96
detection nr 2 = 30 km/h, box pos= x 1072, y 308, confidence = 98
detection nr 3 = 50 km/h, box pos= x 312, y 420, confidence = 97
detection nr 4 = 50 km/h, box pos= x 316, y 420, confidence = 74
detection nr 5 = 50 km/h, box pos= x 312, y 424, confidence = 98
detection nr 6 = 50 km/h, box pos= x 316, y 424, confidence = 88
detection nr 7 = 50 km/h, box pos= x 704, y 432, confidence = 57

 ######## GPU running... ########
cudaMalloc success.
Time to execute layer1_feature_maps:  0.8 ms
GPU Elapsed time is  3.6 ms
Checking GPU results of layer 1 ...
GPU layer 1 passed.
```

Time to execute layer1_feature_maps (in ms)										Fastest
0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8

GPU Elapsed time (in ms)										Fastest
3.7	3.6	3.7	3.7	3.7	3.6	3.6	3.7	3.7	3.7	3.6

Report

I used the block size specified in the CUDA file and didn’t use hardcoded number.

It is obvious that the execution time of program 2 (3.6 ms) is less than program 1 (6.1 ms). Except kernel “layer1_feature_maps()”, all other parts in two programs are the same and use around 2.7 ms to execute. As layer1_init_bias() and layer1_sigmoid() use little time (~ 0.2 ms) to execute, we can neglect them. What we want is to examine whether there are real speedup if we use shared memory to cache the input, so we should focus on the execution time of kernel “layer1_feature_maps()”. Program 2 used significantly less time than program 1. If we cache the input by shared memory, the speedup is 3.4 / 0.8 = 4.25. It reveals that shared memory is very powerful and we should make use of them when necessary.

“layer1_init_bias" block size: 16∗16∗2
”layer1_feature_maps" block size: 8∗8∗8
“layer1_sigmoid" block size: 14∗14∗2

No surprise, I use 3D block and 3D grid to finish the task, since there are multiple feature maps. The block sizes are neither too big nor too small, since it may help balance the load. And for some GPUs, maximum number of resident threads per SM is lower than 2048. For instance, if we set the number of threads per block to be 1024, we may not be able put two blocks in the same SM. This may affect the performance. I tried to reuse some calculation results and store them in variables, such as index, row, column and depth. Also, I balanced the load of each thread, so my codes have reasonable performance. In the code comments, there is detailed explanation of the logic of the program.