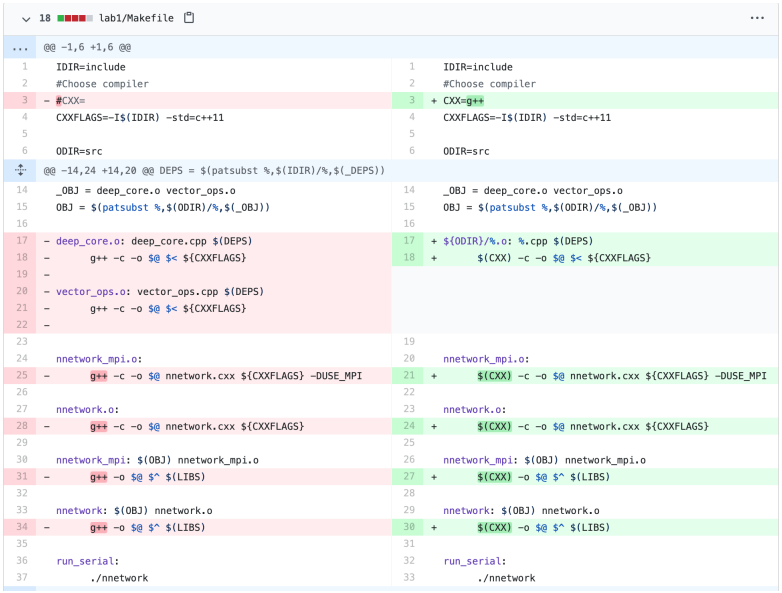
Report for Lab 1

Task 1

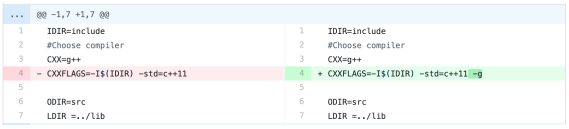
The attached screenshot shows the changes made to introduce a variable for the `C++` compiler used.



```
18 lab1/Makefile
@@ -1,6 +1,6 @@
 IDIR=include
 #Choose compiler
-#CXX=
+CXX=g++
 CXXFLAGS=-I$(IDIR) -std=c++11

 ODIR=src
@@ -14,24 +14,20 @@ DEPS = $(patsubst %,$(IDIR)/%,$(_DEPS))
 _OBJ = deep_core.o vector_ops.o
 OBJ = $(patsubst %,$(ODIR)/%,$(_OBJ))

-deep_core.o: deep_core.cpp $(DEPS)
-	g++ -c -o $@ $< ${CXXFLAGS}
-
-vector_ops.o: vector_ops.cpp $(DEPS)
-	g++ -c -o $@ $< ${CXXFLAGS}
-
+${ODIR}/%.o: %.cpp $(DEPS)
+	$(CXX) -c -o $@ $< ${CXXFLAGS}

 nnetwork_mpi.o:
-	g++ -c -o $@ nnetwork.cxx ${CXXFLAGS} -DUSE_MPI
+	$(CXX) -c -o $@ nnetwork.cxx ${CXXFLAGS} -DUSE_MPI

 nnetwork.o:
-	g++ -c -o $@ nnetwork.cxx ${CXXFLAGS}
+	$(CXX) -c -o $@ nnetwork.cxx ${CXXFLAGS}

 nnetwork_mpi: $(OBJ) nnetwork_mpi.o
-	g++ -o $@ $^ $(LIBS)
+	$(CXX) -o $@ $^ $(LIBS)

 nnetwork: $(OBJ) nnetwork.o
-	g++ -o $@ $^ $(LIBS)
+	$(CXX) -o $@ $^ $(LIBS)

 run_serial:
 	./nnetwork
```

Task 2

(a) To make a debug build, we added the following flag.



```
@@ -1,7 +1,7 @@
 IDIR=include
 #Choose compiler
 CXX=g++
-CXXFLAGS=-I$(IDIR) -std=c++11
+CXXFLAGS=-I$(IDIR) -std=c++11 -g

 ODIR=src
 LDIR =../lib
```

(b) We changed the following lines to fix the *segmentation* fault.

```
@@ -115,7 +115,7 @@ int main(int argc, char * argv[]) {
         print ( yhat, 10, 10 );
         cout << "Ground truth:" << "\n";
         print ( b_y, 10, 10 );
-        vector<float> loss_m +//= yhat - b_y;
+        vector<float> loss_m = yhat - b_y;
         float loss = 0.0;
         for (unsigned k = 0; k < BATCH_SIZE*10;
     ++k){
```

(c) After variable *size* is initialized, the value it contains is 32768. The variable z is a *const* reference vector, containing floats. `z[3] = 1.96690202`.

```
Breakpoint 1, relu (z=...) at src/deep_core.cpp:77
77          vector <float> output;
(gdb) l
72          return output;
73      }
74
75      vector <float> relu(const vector <float>& z){
76          int size = z.size();
77          vector <float> output;
78          for( int i = 0; i < size; ++i ) {
79              if (z[i] < 0){
80                  output.push_back(0.0);
81              }
(gdb) p size
$1 = 32768
(gdb) p z[3]
$2 = (const __gnu_cxx::__alloc_traits<std::allocator<float>, float>::value_type &) @0x5555555f20dc: 1.96690202
```

Task 3

(a) First, we have created a rule in the *Makefile* to enable performance profiling. In particular, the rule is given by

```
run_perf:
    perf record ./nnetwork     .
```

(b) After running the added rule, we have inspected the generated performance file `perf.data` by running the command `perf report`. Overall, there are three time-consuming operations.

(1) The most time is spent in the `dot` function in the file `vector_ops.cpp`. It requires about 67.62% of running time and computes the product of two matrices. Optimizing it could highly improve the total running time of the training.

(2) The second and third most time-consuming operations are array accesses on vector data-structures with 18.41% and 9.36% respectively. These operations might be optimized by reducing the number of cache misses while accessing vector elements.

(3) Apart from those three operations, there are not any others with significant share on the running time.

(c) To monitor the number of LLC cache misses, we have run the command `perf stat -e LLC-load-misses ./nnetwork`. The program returned

```
 Performance counter stats for './nnetwork':

         2,043,328      LLC-load-misses:u

     215.866059446 seconds time elapsed

     215.823836000 seconds user
       0.035999000 seconds sys
```