Assignment-1
Anannya Mathur 2019TT10953

The sequential code analysis:
Dataset used: Gene
Number of epochs: 10

Training time: 0.86 s
Test time: 0 s
MSE value for the test data: 0.054765

As observed, fann_run utilises 58.14% of the total time; thus, it will be a primary focus for further optimisation.
In total, there are three main functions that have to be optimised, which implement the feed-forward neural network plus enable back-propagation - fann_run, fann_update_slopes_batch, and fann_backpropagate_MSE.

Optimisation in Cuda:

1. Optimising fann_run (fann_run runs input through the neural network, returning an array of outputs):
    - Cudamemcpy, cudamalloc, cudafree consumed a lot of time. Each call took almost 3s. Neglecting the time consumed by memory copy calls, the use of the Cuda kernels reduced the training time from 0.86 s to 0.79s without compromising the accuracy of the neural network(as required).
      MSE value for the test data: 0.011871

```
==11852== Profiling application: ./fann.exe gene.train gene.test
==11852== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   33.17%  690.85ms    196908  3.5080us  1.0230us  20.032us  [CUDA memcpy DtoH]
                   28.45%  592.53ms    196908  3.0090us     607ns  17.568us  [CUDA memcpy HtoD]
                   20.22%  421.26ms     49227  8.5570us  2.4000us  26.560us  assign_fann_run(double*, fann_neuron*, int)
                   18.16%  378.36ms     49227  7.6860us  2.3030us  25.216us  assign2_fann_run(fann_neuron*, double*, int)
      API calls:   65.49%  38.9284s    393816  98.849us  10.900us  26.461ms  cudaMemcpy
                   19.75%  11.7428s    196908  59.635us  2.9000us  26.628ms  cudaFree
                   10.79%  6.41427s    196908  32.574us  3.6000us  144.30ms  cudaMalloc
                    3.92%  2.32953s     98454  23.661us  14.900us  2.5696ms  cudaLaunchKernel
```

2. Optimising fann_backpropagate_MSE(propagates the error backwards from the output layer)
    - Neglecting the time consumed by the memory calls, the training time decreased from the initial 0.86s to 0.75038s.
    - MSE value for the test data= 0.036946

```
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   31.52%  577.24ms    165148  3.4950us  1.0550us  20.705us  [CUDA memcpy DtoH]
                   27.50%  503.62ms    165148  3.0490us     607ns  20.769us  [CUDA memcpy HtoD]
                   24.19%  442.97ms     47640  9.2980us  2.4960us  26.240us  backpropagate_gpu(int, double*, double, double*)
                    8.84%  161.80ms     17467  9.2630us  3.0720us  26.272us  assign_fann_run(double*, fann_neuron*, int)
                    7.95%  145.62ms     17467  8.3360us  2.3670us  25.312us  assign2_fann_run(fann_neuron*, double*, int)
      API calls:   60.60%  37.7555s    393816  95.870us     200ns  48.778ms  cudaMemcpy
                   22.24%  13.8538s    196908  70.356us     300ns  28.892ms  cudaFree
                   12.71%  7.91637s    196908  40.203us     200ns  160.43ms  cudaMalloc
                    4.39%  2.73424s     98454  27.771us     300ns  2.7428ms  cudaLaunchKernel
                    0.06%  38.257ms         1  38.257ms  38.257ms  38.257ms  cuDevicePrimaryCtxRelease
                    0.00%  53.900us         1  53.900us  53.900us  53.900us  cuModuleUnload
                    0.00%  17.800us       101     176ns     100ns  1.0000us  cuDeviceGetAttribute
```

3. Optimising fann_update_slopes_batch(Updates slopes for batch training):

- Neglecting the time taken by memory copy calls, the training time decreased from 0.86s to 0.606s.

```
            Type  Time(%)      Time     Calls       Avg       Min       Max  Nam
e
 GPU activities:   41.71%  639.12ms    355748  1.7960us     992ns  12.896us  [CU
DA memcpy DtoH]
                   32.37%  495.90ms    355748  1.3930us     960ns  11.649us  [CU
DA memcpy HtoD]
                   13.75%  210.62ms     95280  2.2100us  1.8880us  11.712us  slo
pes_gpu(int, double*, double, fann_neuron*)
                    6.98%  106.90ms     47640  2.2430us  2.0480us  12.576us  bac
kpropagate_gpu(int, double*, double, double*)
                    2.65%  40.539ms     17467  2.3200us  2.1440us  8.5760us  ass
ign2_fann_run(fann_neuron*, double*, int)
                    2.55%  39.086ms     17467  2.2370us  2.0480us  10.496us  ass
ign_fann_run(double*, fann_neuron*, int)
```

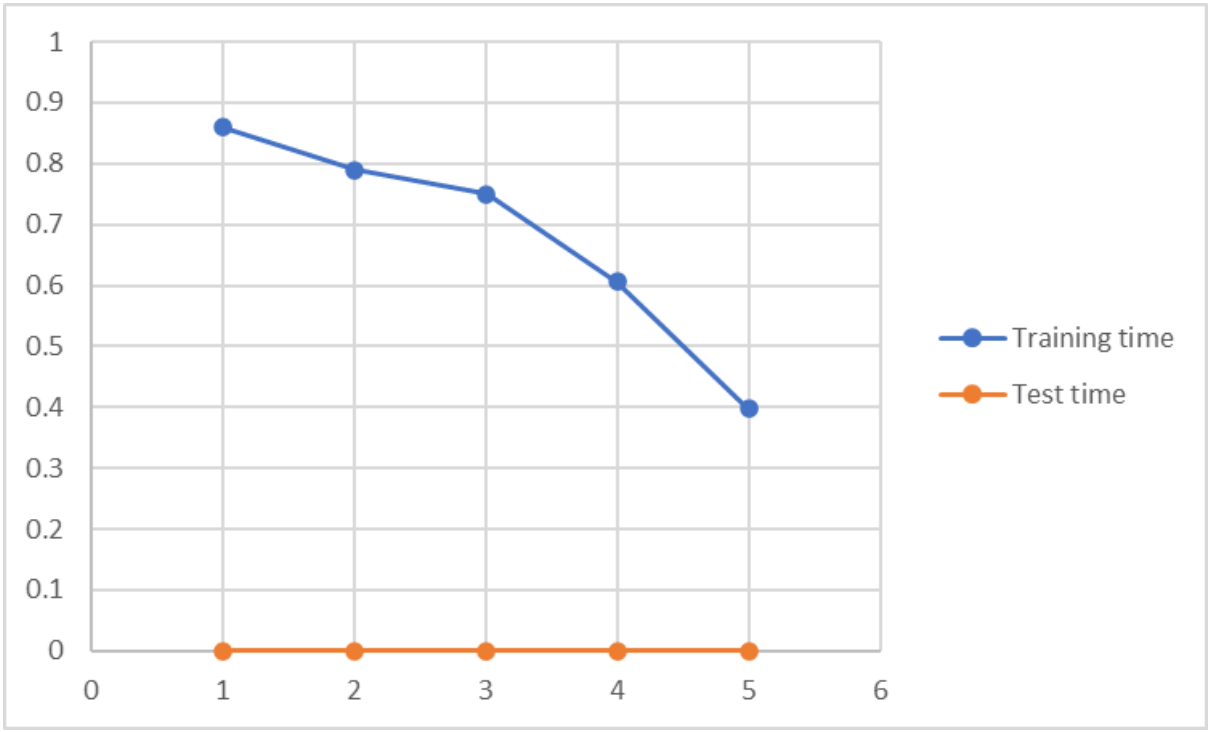
MSE value for the test set: 0.04481

4. Optimising fann_update_weights_irpropm (The training algorithm):

- The function initially took 0.2s, but the optimisation reduced the time taken to 32.993us; therefore, the training time currently stands at 0.397s.

```
MSE value for the test data (0.017704)
Cleaning up.
==10744== Profiling application: ./fann gene.train gene.test
==10744== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Nam
e
 GPU activities:   41.71%  639.12ms    355748  1.7960us     992ns  12.896us  [CU
DA memcpy DtoH]
                   32.37%  495.90ms    355748  1.3930us     960ns  11.649us  [CU
DA memcpy HtoD]
                   13.75%  210.62ms     95280  2.2100us  1.8880us  11.712us  slo
pes_gpu(int, double*, double, fann_neuron*)
                    6.98%  106.90ms     47640  2.2430us  2.0480us  12.576us  bac
kpropagate_gpu(int, double*, double, double*)
                    2.65%  40.539ms     17467  2.3200us  2.1440us  8.5760us  ass
ign2_fann_run(fann_neuron*, double*, int)
                    2.55%  39.086ms     17467  2.2370us  2.0480us  10.496us  ass
ign_fann_run(double*, fann_neuron*, int)
                    0.00%  32.993us        10  3.2990us  2.8480us  4.2880us  gpu
_update_wts(int, int, double*, double*, double*, float, float, float, float, dou
ble*)
      API calls:   42.62%  23.8864s    451028  52.959us     121ns  139.80ms  cud
aMalloc
                   36.46%  20.4335s    451028  45.304us     282ns  19.858ms  cud
aFree
                   17.12%  9.59276s    902056  10.634us     118ns  28.913ms  cud
aMemcpy
                    3.81%  2.13292s    225504  9.4580us     340ns  1.7406ms  cud
aLaunchKernel
                    0.00%  847.09us         2  423.55us  419.88us  427.21us  cuD
eviceTotalMem
                    0.00%  411.42us       192  2.1420us     110ns  91.155us  cuD
eviceGetAttribute
                    0.00%  49.611us         2  24.805us  19.686us  29.925us  cuD
eviceGetName
                    0.00%  19.337us         2  9.6680us  1.8210us  17.516us  cuD
eviceGetPCIBusId
                    0.00%  4.5350us         4  1.1330us     161ns  3.4130us  cuD
eviceGet
                    0.00%  1.6420us         3     547ns     127ns     824ns  cuD
eviceGetCount
```

MSE value: 0.017704



The speedup achieved by the code= 0.86/0.397=2.166.

Amdahl's law:
Since around 40% of the code is serial, the maximum speedup that can be achieved= 100/40=2.5.

On the mushroom dataset,
Time taken by the serial code to train the data: 0.86s.
Time taken by the parallel code(neglecting the Cuda memory calls): 0.607s
The speedup achieved by the code= 0.86/0.607=1.4168.
The test time taken by both the codes=0s.