Each record corresponds to a 20-minute period of operation of the device that contains two inputs and one output:

- Input 1: The normalized number of requests received by the device during the 20 min period.
- Input 2: The processor average load during that period.
- Output: Normal Operation (0) or Crash (1).

Since the task of optimizing the hyperparameters from the neural network developed on the first project using an evolutionary algorithm would take some time to run and to get right, this was first developed, instead of the fuzzy system. For this, the *deap* library was used.

This time, to improve the results from the last project, new features were added:

- 'High_Load' – binary value, 1 if the load value exceeds 0.53
- 'High_features" – binary value, 1 if the load value exceeds 0.53 and the requests value exceeds 0.2
- 'Requests_mm' – Moving average with size 5 for the requests values
- 'Load_mm' – Moving average with size 5 for the load values

Each individual has a list based binary chromosome. This chromosome encodes the number of layers of the NN, the sizes of the NN's layers, and the used features. The chromosomes are generated randomly in two parts.

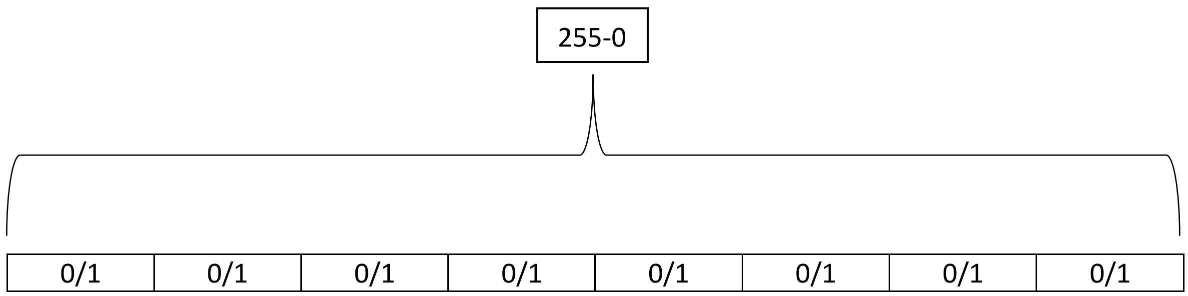
First, the features to be used, with the following structure:

Requests	Requests1	Requests2	Load	High requests	High_load	High Features	Requests _mm	Load_mm
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Each bit represents one feature, and if the bit is one, that feature is set to be used on the NN.

Next, the number of layers and the layer sizes are generated. The number of layers is generated randomly with numbers ranging from 1 and 3 (the maximum number of layers can be set by a variable.

The layers sizes are represented in binary. The maximum size can also be set from a variable, but this number must by a power of 2. For the project, the maximum size for each layer is 256 (2**8).

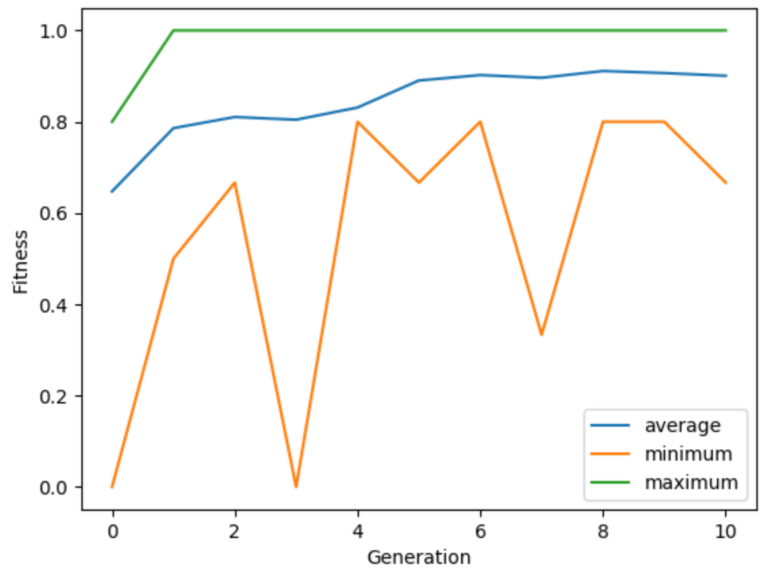


During the chromosome generation, the final list must be checked to ensure that it is valid. For the chromosome to be valid, the at least one feature must be selected, and there can be no layers with size zero.

The fitness of an individual is represented by the f-score (of the testing set) of an NN that uses the chromosome’s parameters. When an individual is evaluated, the previously mentioned parameters must also be checked beforehand. This second check is needed because an individual can become invalid due to mutations or crossovers. If an individual is invalid, the its fitness is set to be zero.

The development of the EA was faced with some degree of difficulty due to some strange behaviors of the *deap* library. For example, the generated individuals, although generated randomly, were always equal. This was solved by using the repeat function from the toolbox, even it generated function was repeated only once. Although this solved the issue of only having one type of individual, now each individual had an individual layer of list. Since this made the individuals incompatible with the mating and crossover functions and removing the unwanted list layer did not solve the issue (because of how *deap* deals with fitness), the mating and crossover functions needed to be altered to work with this new, although unwanted structure. These functions are included alongside the project and have the suffix “fix” on the function’s name.

Finally, after several days of bug tracking and fixing, the algorithm was working, and results were extracted:



From the graph it is possible to conclude that the evolutionary algorithm is working. The maximum fitness remains at the maximum value for almost all the run, indicating that the best individuals are affecting the next generations and being used for mating. The average trend has an upward trend for most of the run and the minimum trend is quite variable, mostly caused by poor performing individuals caused by mutations.

The best configuration (best fitness) was as follows:

- Layer sizes:
  - (100, 23, 153)
- Input features
  - ['Requests1', 'Load', 'High_requests', 'Requests_mm']

This configuration yields the following results:

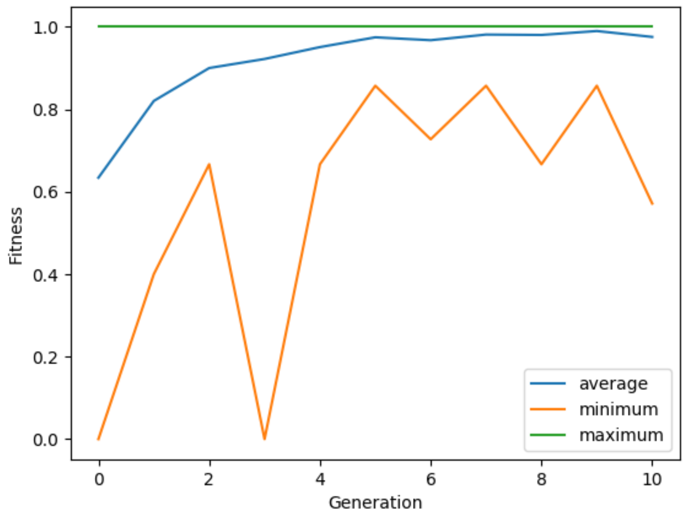
- Training set:
  - F-measure: 1.0
  - Confusion Matrix:

    [490 , 0]
    [0 , 8]

- Testing set:
  - F-measure: 1.0
  - Confusion Matrix:

    [495 , 0]
    [0 , 4]

A new run was made with oversampling enabled to see if it would give the same results:



The best configuration (best fitness) was as follows:

- Layer sizes:
  - (69, 55, 43)
- Input features
  - ['Requests', 'Requests1', 'Load', 'High_requests', 'High_features', 'Requests_mm']

This configuration yields the following results:

- Training set:
  - F-measure: 1.0
  - Confusion Matrix:

    [490 , 0]
    [0 , 490]

- Testing set:
  - F-measure: 1.0
  - Confusion Matrix:

    [495 , 0]
    [0 , 4]

With the all the time spent working on the EA task, the developed fuzzy system, although working, the parameters are not 100% optimized.

The best configuration (best fitness) was as follows:

- Inputs:
  - ['Load', 'Requests_High', 'Requests_mm"]
- Input ranges
  - Load
    - High: [0.55, 1, 1]
    - Medium: [0.45, 0.5, 0.65]
    - Low: [0, 0, 0.55]
  - Requests_High
    - High: [1, 1, 1]
    - Low: [0, 0, 0]

- Requests_mm
  - High: [1.8, 2, 3.55]
  - Medium: [1.25, 1.5, 1.85]
  - Low: [0, 0, 1.3]

This configuration yields the following results:

- Dataset:
  - F-measure: 0.82
  - Confusion Matrix:

    [984 , 1]
    [3 , 9]

Now, with the developed systems, a new dataset is used as a standalone test set

LTFS:

- Training set (old dataset):
  - F-measure: 0.57
  - Confusion Matrix:

    [965 , 2]
    [593 , 394]

- Testing set (new dataset):
  - F-measure: 0
  - Confusion Matrix:

    [173 , 0]
    [6 , 0]

MLP NN (1^{st} project with old expert options, no oversampling):

- Training set (old dataset):
  - F-measure: 0.95
  - Confusion Matrix:

    [984 , 1]
    [0 , 12]

- Testing set (new dataset):
  - F-measure: 0.75
  - Confusion Matrix:

    [189 , 1]
    [3 , 6]

EA (with additional expert features, no oversampling):

- Training set (old dataset):
  - F-measure: 0.92
  - Confusion Matrix:

- Testing set (new dataset):
  - F-measure: 0.9
  - Confusion Matrix:

[983 , 2]
[0 , 12]

[188 , 2]
[0 , 9]

EA (with additional expert features, with oversampling):

- Training set (old dataset):
  - F-measure: 0.99
  - Confusion Matrix:

    [983 , 2]
    [0 , 985]

- Testing set (new dataset):
  - F-measure: 0.7
  - Confusion Matrix:

    [182 , 8]
    [0 , 9]

Fuzzy system:

- Dataset:
  - F-measure: 0.46
  - Confusion Matrix:

    [189 , 1]
    [6 , 3]

Conclusions

On this project, new approaches were explored with the goal to improve the results from the first project. Firstly, an evolutionary algorithm was used to optimize the 1st project NN hyperparameters. The optimized values given by the EA yielded very good results, on the old dataset and the new dataset. A slight decrease was found testing with the new dataset with the results from the oversampled EA system. This increase did not only come from the optimized parameters, but also from the new features that were added. The fuzzy system, although not optimizes would also be a good option for this project, I would say that it would be possible to achieve f-score values close to one, on both datasets. A great portion of the time spent on the project was debugging fixing and running the EA algorithm. Each development cycle was very long, at times the system seemed to be working only to crash after a couple of hours. Each run with 10 generations and 50 population took several hours on an instance of the SIGMA server. But in the end this approach seemed to be the best.