

Appendix 1 - System Manual

This appendix will describe how to use the systems of the project. Most of the attention is directed toward the main system. How the two other systems, i.e. the merging subpopulations system and the convolutional system, differs from the main system, will be explained toward the end of this appendix.

1 Running the Main System

The program is written in Python 3, and requires the following non-standard libraries to run:

- TensorFlow
- Matplotlib
- idx2numpy

The program consists of four program files:

- main.py
- genetic_algorithm.py
- neural_net.py
- ensemble.py

The program can be started with the command *python3 main.py*.

Two command line arguments can be specified with this command:

- **-c "configuration file"** - specifies the file path to a configuration file where all system parameters are set (see section 2 of this appendix)
- **-r "restore file"** - specifies the file path to a restore file which allows a previous non-completed run to be continued (see section 3 of this appendix)

If no configuration file is specified, default values are used instead (see section 2). If no restore file is specified, the program will start running from the first generation.

Additionally, the system requires the dataset which the system will use for training to be available. The relative file path to the datasets can be specified in the configuration file.

2 Configuration File of the Main System

The configuration file should be a text file. Its filepath is specified by the command line argument described in section 1. Every parameter in the file must be on a separate line. The syntax of each line should be as follows:

parameter name: value

If a line has an invalid parameter name, that line will simply be ignored. If it has an invalid value, the program might crash, depending on what parameter and value it is.

2.1 Normal Parameters

Table 1 lists all parameters that might be specified in the configuration file, except those used to restrict the search space (these are described in the next subsection).

2.2 Restrictions of the Search Space

Table 2 shows the parameters which can be set to restrict the search space. If a line is not specified, no such restriction will be set. That means the default value of every restriction is to not use any restrictions at all.

3 Output of the Main System

This section will briefly explain the output produced by the system.

The following is printed to the standard output stream while the system is running:

- After every generation, information about the best nets are printed. The different values printed for each individual represents the following, in that order: was the net created in this generation, id, id of first parent, id of second parent, age, mutation, topology, learning rate, learning rate decay, activation function, optimizer, cost function, dropout probability, batch size, steps trained, accuracy on validation set, diversity on validation set, scaled accuracy on validation set, scaled diversity on validation set, fitness value.
- At the end of the system's run, final results are printed. This includes validation and testing accuracy for the best individual, as well as testing accuracy for the ensemble. Also, if detailed results are set to be printed by that parameter in the configuration file, the average diversity, average accuracy on the validation set, average accuracy on the testing set, highest accuracy on the validation set, highest accuracy on the testing set, and rate of new individuals for every generation will be printed, as well as the distribution of what generation the individuals in the final population were created in.

Additionally, the system produces the following output:

Parameter Name	Description	Valid Values	Default Value
Data set	What dataset to use	mnist, emnist, fashion_mnist, chess or yeast	mnist
Population size	Size of the population	Positive integers	10
Children per generation	Number of new individuals that will be created in every generation	Positive integers, or "same" (same as the population size)	same
Total time	Total training time for each neural network in seconds	Positive integers	150
Generations	Number of generations	Positive integers	10
Data case fraction	Fraction of the dataset to be used in total	Numbers in the range [0, 1]	1
Age factor	To what degree younger individuals should be prioritized	Positive numbers	2
Crossover rate	The fraction of children which will be created by two parents	Numbers in the range [0, 1]	0.5
Random immigrant rate	The fraction of children which will be created as random immigrants without parents	Numbers in the range [0, 1]	0.1
Ratio of generations with alpha=0	The fraction of generations in the end where only accuracy will be considered	Numbers in the range [0, 1]	0.25
Validation interval	Steps between each early stopping test will be executed during training	Positive integers	100
Patience	Steps before training will be terminated if no improvements have been found	Positive integers	10000
PFC sample rate	Fraction of the validation set to be used to calculate diversity	Numbers in the range [0, 1]	1
PFC individuals	Number of other individuals each individual will be compared against when calculating diversity	Integers in the range $[1, PopulationSize - 1]$, or "all" (all individuals will be used)	all
Early stopping fraction	Fraction of the used dataset to be used to evaluate the neural networks during training to determine early stopping	Numbers in the range [0, 1]	0.10
Validation fraction	Fraction of the used dataset to be used to evaluate the fitness of each individual	Numbers in the range [0, 1]	0.20
Test fraction	Fraction of the used dataset to be used as an independent test set for final testing	Numbers in the range [0, 1]	0.15
Max cpus	Maximum number of neural networks to be trained in parallel	Positive integers	20
Print length	Maximum number of individuals to be printed after each generation	Positive integers	20
Mnist path	File path to where the MNIST dataset is stored	File path strings	mnist/
Emnist path	File path to where the EMNIST dataset is stored	File path strings	emnist_balanced/
Fashion mnist path	File path to where the Fashion-MNIST dataset is stored	File path strings	fashion_mnist/
Chess path	File path to where the chess dataset is stored	File path strings	chess/krkopt.data
Yeast path	File path to where the yeast dataset is stored	File path strings	yeast/yeast.txt
Results path	File path to where the results will be stored	File path strings	results.txt
Stored path syntax	Base name of the restore files created after every generation (the full name will be the base followed by the generation number and ".txt")	Strings	stored
Nets directory	File path to the folder where the neural nets will be stored	File path strings	nets/
Print detailed results	Whether detailed results will be printed when the system is done running	Booleans	true

Table 1: Standard parameters

Parameter name	Valid values	Example
Restrict topology type	How the topology will be restricted. Valid values are: exact (topology will not be evolved), min_max (topology is restricted by a minimum and maximum number of layers, and neurons per layer), fixed_layers_min_max (number of layers are hardcoded, and for every layer a minimum and maximum number of neurons are set, which might differ from layer to layer).	exact
Restrict topology exact	List of positive integers, representing the topology if the type is "exact".	[50, 20, 12]
Restrict topology min layers	Positive integer representing the minimum number of hidden layers if the type is "min_max".	2
Restrict topology max layers	Positive integer representing the maximum number of hidden layers if the type is "min_max".	3
Restrict topology min neurons	Positive integer representing the minimum number of neurons in any layer if the type is "min_max".	5
Restrict topology max neurons	Positive integer representing the maximum number of neurons in any layer if the type is "min_max".	20
Restrict topology exact layers min neurons	List of positive integers, representing the minimum number of neurons per layer if the type is "fixed_layers_min_max". The length of the list represents the number of layers.	[20, 10, 10]
Restrict topology exact layers max neurons	List of positive integers, representing the maximum number of neurons per layer if the type is "fixed_layers_min_max". The length of the list represents the number of layers.	[300, 75, 40]

Restrict activation function	List of activation functions to be part of the search space. Valid activation functions are: sigmoid, tanh, softsign, relu, leaky-relu, swish, selu, softplus.	["relu", "sigmoid"]
Restrict cost function	List of cost functions to be part of the search space. Valid cost functions are: mse, cross_entropy, huber.	["mse"]
Restrict optimizer	List of optimizers to be part of the search space. Valid optimizers are: gd, adam, adagrad, rmsprop, ftrl.	["adam", "adagrad"]
Restrict learning rate	Tuple of two values that represents the lower and upper bound for the learning rate.	[0.01, 0.3]
Restrict learning rate factor	Tuple of two values that represents the lower and upper bound for the learning rate decay.	[1e-08, 1e-05]
Restrict dropout prob	Tuple of two values that represents the lower and upper bound for the dropout probability.	[0, 0]
Restrict batch size	Tuple of two values that represents the lower and upper bound for the batch size.	[32, 128]

Table 2: Restricting parameters

- The trained neural nets are saved in the folder specified in the configuration file, and are removed when that individual is removed from the population. However, the nets in the final population are kept when the system is done running.
- A restore file is created after each generation, named as specified in the configuration file. This file contains all variable values required to restart the system run from this generation, in case the run will be interrupted at a later point. The restore file from the previous generation is automatically removed when the next one has been created.
- A results file is created after the final generation. It contains the same results as those printed to the standard output stream after the final generation, as explained above. Additionally, it contains information about all the individuals in the final population.

4 Merging Subpopulations System

The merging subpopulations system works in approximately the same way as the main system, with the following exceptions:

- The system contains an additional program file, named "super_population.py", which defines a class holding all the normal populations which the system creates.
- The configuration file has an additional parameter, with name "Start populations", determining the number of populations in the first generations of the algorithm's run. There are some requirements to the value of this parameter, which has been explained in the main report.

5 Convolutional System

The convolutional system also works in almost the same way as the main system, with the following exceptions:

- The system contains an additional program file, named "convnet.py", which is used to represent the topology of an individual.
- It is not possible to restrict the topology through the configuration file, as the topology is quite complex for this type of net. Such restrictions could of course have been implemented, but this has not been the focus of the thesis as this is only a side system, and because no tests with such restrictions have been conducted on this system.