Improving genetic algorithms applied to hyper-parameter optimization using weights merges

A.L. GONZALEZ V. VARRUN
algon | varrun @kth.se

January 15, 2021

Abstract

Creation of a Deep Learning model requires knowledge, time, and computational power to figure out the best configuration of a neural network. This process is called hyper-parameter optimization and can be automated with search algorithms, such as genetic algorithms, which are inspired by Darwin's Theory of Evolution. These are based on the concept that a pool of parents with the best genes produce better offspring. The objective of this study is to check if the weight merge technique of producing the best genes can be used to improve the performance of genetic algorithms applied in hyper-parameter optimization. We can hence define a genetic algorithm and a tool that can be used in future studies to speed up execution time of fitness function while maintaining accuracy of the results.

Keywords: Genetic Algorithm; Hyper-parameter optimization; Deep Learning; Neural Network; Evolutionary Algorithm

Contents

List of Acronyms and Abbreviations

GA Genetic Algorithm
HPO Hyper Parameter Optimization

1 Introduction

Deep Learning is an Artificial Intelligence technique that has become very popular in the past years because of its successes in different fields. To build a deep learning model It is necessary to define its architecture and initial configuration in a process called hyperparameter selection [1]. The internal values, called weights, are then modified in order to achieve the desired result. This process is called training, where the neural network is taught to learn the desired results, which in turn results in elevated costs and computational power. The choice of the optimal hyperparameters is crucial for the elaboration of an accurate and efficient model. Incorrect pre-configuration of a deep learning system can lead to models being unable to fit the training data or performing poorly with the test data.

A new type of algorithm was developed in the last decade, called genetic algorithm [2]. This is an evolutionary algorithm that was inspired by natural selection in nature to find an optimal solution. It is based on the concept that a pool of parents with the best genes produce better offspring.

Genetic algorithms consist of generations called population, and the total size of populations are defined as number of solutions. Each solution is called individual, and each individual solution in turn has a chromosome. Each chromosome has a set of genes, which are represented as strings of 0s and 1s. Each individual in a set of genes has a fitness function. This fitness function defines the competitiveness of an individual. A higher fitness value is desired in order to ensure the best individuals in a mating pool. The individuals in the mating pool are called parents, where two parents are selected from the mating pool to generate two offspring (children). There are two different techniques to create offspring: crossover and mutation. The entire process is reiterated until an optimal individual is obtained, which would be the best solution to the problem. The process is best described in the following visualization [3].

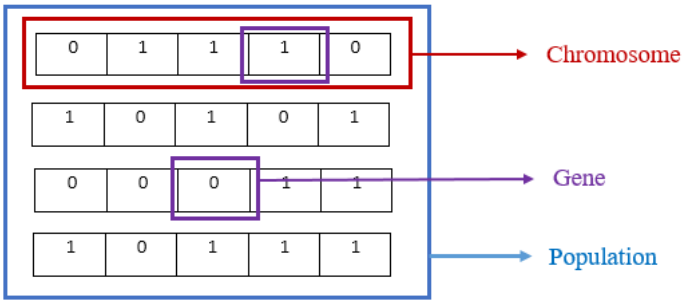

Figure 1: Composition of individuals in a genetic algorithm

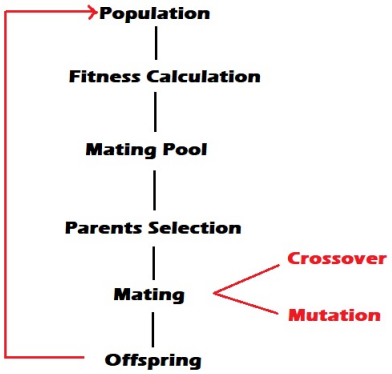

Figure 2: Genetic Algorithm flowchart

This research reviews some successful applications of genetic algorithms in hyperparameter optimization, highlights the concept of weight recombination, and evaluates whether it can benefit the performance of the algorithm. The automation of the search for the best parameters to configure a neural network can

democratize artificial intelligence, allowing more people and companies to use these tools without requiring advanced knowledge in Deep Learning.

2 Background

Despite the popularised use of back propagation-based methods for training neural networks, J. N.D. Gupta et al. [4] tested a new approach using genetic algorithms for training neural networks. In this case, the individuals of the population are deep learning models that are evaluated during the fitness function, and their internal parameters (weights) are tuned in the crossover and mutation stages. J.N.D. Gupta et al. concluded that this approach to find the best architecture for a neural network, that is, to perform a hyper-parameter optimization has major benefits. Further research by D. J. Montana [5] works on this concept and incorporates the idea of combining weights of different models, keeping in each case the lowest values.

The first implementation of this algorithm applied successfully to the optimization of hyperparameters is in Young's et al. work[6] where it is used to define the best configuration of the model but leaving aside the layers and neurons of the network, a fact that stands out as an interesting topic for future research.

Other related work[7] suggests that one can continue to build on works by J.N.D. Gupta's et al. and D.J. Montana's et al. that use the genetic algorithm for neural network training and also combine it with the hyper parameter optimization process. In other words, to modify internal weights and hyperparameters in the crossover and mutation stages. Although our current research considers this an interesting approach, we believe that training the network only with the genetic algorithm would unnecessarily multiply the number of solutions to be evaluated by the genetic algorithm, when backpropagation algorithms train these models relatively effectively.

2.1 Problem discussion

The performance of these genetic algorithms, in terms of execution time, is the main concern when using them together with artificial neural networks, especially in the optimization of hyper-parameters. Authors such as Y. Sun et al.[8], consider that it is necessary to dedicate efforts in the improvement of the fitness functions to accelerate the global time of the algorithm.

One of the best results is found in Esteban's et al. work, applying this combination of algorithms to large-scale image classifiers. In their research[9], they work with the concept of *weight inheritance* from parents to children at the crossover stage. This inheritance occurs only when the shape of the parent's neural network layers match and no combination of these weights is carried out (in case the parents have learned different things from the data set). The author of this study poses a question of possible future study of weight inheritance and using a combination of weights to solve fitness function issues.

In this research, we propose to include weight merge technique in the crossover stage. This merging of weights of the best individuals (models) is done to speed up the fitness function and to improve the performance of the whole algorithm.

2.2 Research questions and hypothesis

The purpose of this research is to find a tool based on genetic algorithms that will improve the process of identifying the optimal hyperparameters for a neural network, reusing in each step the previous training, given in a given data set.

Based on the ideas of Esteban et al. about weight inheritance and future work on the combination of weights (merge) and the problems to be solved with respect to the fitness functions posed by Y. Sunet al., we pose the following research question:

Can weight merge speed up the execution of genetic algorithms applied to hyperparameter optimization?

Understanding *weight merge* as a combination of the weights of the progenitors in the crossover stage of a genetic algorithm, regardless of the shape of its layers, our hypothesis is:

The use of weight merging can improve the performance of the fitness function, reducing its execution time and accelerating the process of finding the optimal architecture.

3 Research methodology

In this section, we present the methodology followed during this research, chosen based on the project requirements and the hypothesis. Standard datasets have been used to perform the same experiment, derive metrics and to evaluate the results.

3.1 Choice of research method

The project makes use of quantitative research focused on the validation of a hypothesis through experiments and the statistical analysis of the data. These experiments test the proposed technique, and the analysis provides us a better understanding if the variables of the problem (duration of the fitness function and result of the algorithm) verify or deny the hypothesis.

The reason why the study was decided to be carried out as an experimental research with a deductive approach, was to generalize the behaviour of this technique from the experiments, data collected and using statistical analysis as the primary method (metrics such as skewness and mean) [10].

3.2 The experiment and metrics

The experiment consists of running two genetic algorithms for the training of neural networks with several datasets. The first instance is the implementation of a basic algorithm based on previous work in the field. The aim of this project is not to test the most advanced solutions or evaluate them with the largest and most complex datasets, but to test the impact of a specific technique on the overall performance of the algorithm. Therefore, the focus is on a basic implementation that does not require massive computing power. The second instance is based on the first, but includes the weight merging technique implemented (see Algorithm 1 for the skeleton of both algorithms).

In both cases, the algorithm generates, during several rounds, a list of deep learning models. These models are trained and tested during the fitness function stage, and the result of these tests decides whether to discard the model or to use it in the mutation and crossover stages. In the crossover stage, the remaining individuals (the best candidates) are grouped by pairs. Each pair is iterated against a gene list. The following procedure of comparing pairs of layers is followed: if the layers of both parents have the same shape, the average of their weights are calculated, obtaining a new layer with the same dimensions. In other cases, the shape of the larger or smaller layers are decreased or increased respectively, in order to merge the two.

For each algorithm execution (all the rounds), the total duration of the algorithm, the number of rounds required, the final accuracy obtained, and the fitness duration for each model for each round are collected. The latter is the most important metric because it allows us to compare the impact of the weight-merge method on the performance of the fitness function.

3.3 Data

The study uses three commonly known and used datasets to train the algorithm: MNIST, Boston housing prices and wine quality. The datasets chosen were well-known, and not computationally intensive. The main idea of the project is to improve the efficiency in terms of speed while keeping the accuracy same, hence the complexity is for a later study.

Algorithm 1 Genetic algorithm

```
p ← GENERATEPOPULATION()                                  ▷ Create the individuals
while accepted accuracy not reached do
    p ← GENERATION(P)
end while
procedure GENERATION(p)
    for each indv in p do
        FITNESSFUNCTION(indv)
    end for
    SELECTION(p)                                          ▷ Sort by accuracy and remove the N worst
    CROSSOVER(p)
    MUTATION(p)
    return p
end procedure
procedure FITNESSFUNCTION(p)
    TRAIN(INDV.MODEL)
    indv.accuracy ← TEST(indv.model)
end procedure
procedure MUTATION(p)
    for each indv in p do
        for each gene in chromosome do
            x ← RANDOM()
            if x ≤ probability then
                gene ⇐ GENESTORE[random()]                ▷ Pick other gene from the store
            end if
        end for
    end for
end procedure
```

MNIST [11] is a large database of handwritten digits by the National Institute of Standards and Technology. The database was chosen as the standard training set for this project due to its widespread use and prevalence in training various image processing systems. The MNIST dataset consists of 60,000 training and 10,000 testing images.

The Boston housing prices is a dataset by the US Census in the Boston area and has been used extensively as a benchmark to test and train algorithms. It consists of 506 unique rows and 14 attributes such as crime per capita in the locality, average number of rooms, and age of the building. It is commonly used in regression algorithms to predict housing prices given a value.

The wine quality dataset contains physiochemical and sensory attributes of the variants of Portuguese 'Vinho Verde' wine. It is generally used for both classification and regression tasks. There are 12 different attributes such as fixed acidity, volatile acidity, density, pH, and sulphates. It is a data science classic that is easy to use and train.

3.4 Analysis and presentation

At the end of the experiments, the metrics are grouped by algorithm (with or without weight-merge) and evaluated. The chief motive is to ensure that the final precision/error of the algorithm that implements weight merge technique is equivalent or better to one without this implementation. We do not expect to have better results in the second implementation, but it does not make sense to analyze the performance of the fitness function if the final implementation works worse in the case of weight merge. The accuracy of the model will be collected in classification cases and the Mean Absolute Error in regression problems.

Figure 3: MNIST sample dataset

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10

Figure 4: Boston prices sample dataset

Two visualisations describe the results of this study: a histogram with the duration of the fitness function, separated by the type of algorithm (with or without weight fusion), and a bar graph with the final duration of each algorithm in each experiment.

With the first figure, at the moment since this methodology is defined, the resulting time distribution of the fitness function was expected. In case the hypothesis is correct, in the weight merge algorithms, we expect to have more individuals in the left part of the histogram (more models with low fitness function times) than in the case of the algorithms without weight merge. This phenomenon can be quantified by measuring the skewness, the higher this value is, the more to the right is the distribution of the figure's mass. [12]

4 Result and analysis

All the experiments were executed in a Google Colab instance. For each of the dataset, the algorithm was executed twice, one with the *USE_MERGE* flag set to *true* (merge), and another with it set to *false* (basic). The table 1 shows a summary of the results of the experiments.

In the case of the MNIST dataset, a classification problem, both algorithms show similar accuracies, hence the weight merge technique does not have a negative impact in the performance of the genetic algorithm. The fitness function's average time duration, which measures the average time it takes to train a neural network model, is 33% lower in the case of the algorithm with merge technique. The distribution

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Figure 5: Wine quality sample dataset

Dataset	Average time (basic)	Average time (merge)	Time skewness (basic)	Time skewness (merge)	Result (basic)	Result (merge)
MNIST	70.7	47.3	1.709575	4.503455	accuracy 0.9641	accuracy 0.9745
The Boston house-price	15.8	7.3	0.231586	1.732404	MAE 3.0085	MAE 3.0086
Wine Quality	9.26	6.65	1.471797	2.675971	MAE 0.4493	MAE 0.4804

Table 1: Summary of the results

of variable ”time” in the case of the improved algorithm is much more skewed than in the case of the basic algorithm. This skewness can be seen in Figure 6, as well as the total duration improvement of the whole algorithm, as shown in Figure 7.

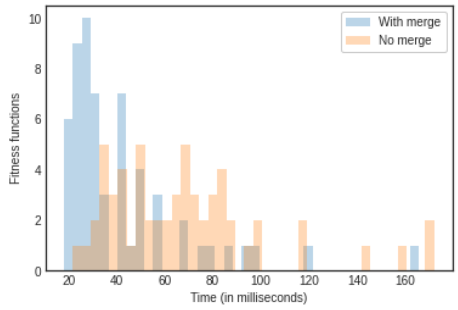

Figure 6: Fitness function time histogram Minst dataset

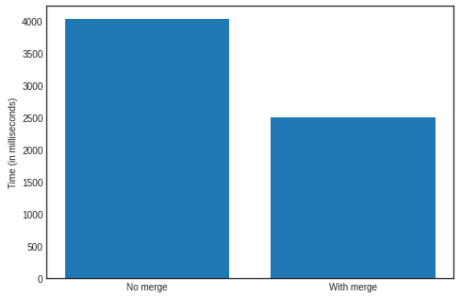

Figure 7: Total duration Minst dataset

In the case of the Boston house-price and the wine datasets, which are regression problems, both show no change in Mean Average Error. However, in the case of weight-merge technique, lower average times and bigger skewness in the distribution of the variable time than the basic algorithm are observed.

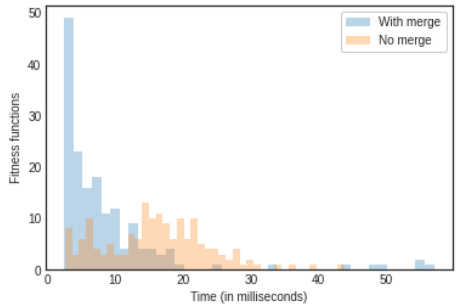

Figure 8: Fitness function time histogram Boston dataset

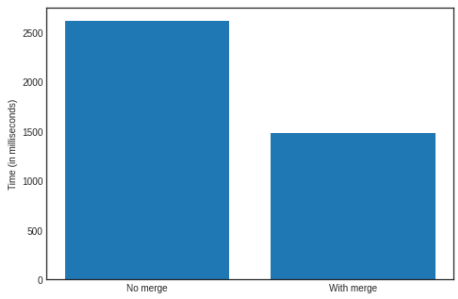

Figure 9: Total duration Boston dataset

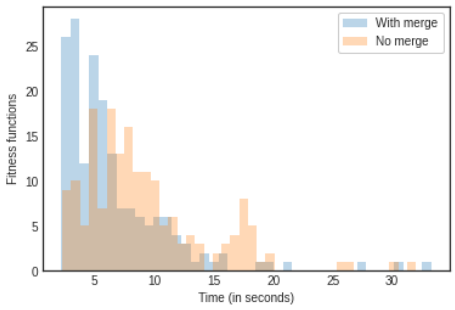

Figure 10: Fitness function time histogram Wine dataset

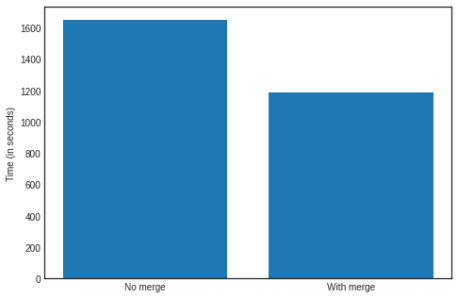

Figure 11: Total duration Wine dataset

5 Discussion

The purpose of this research was to find a way to combine the weights of the neural networks, in the crossing stage of a genetic algorithm in a way that reduces the duration of the fitness functions without altering the performance of the obtained models.

As showed in Section 4, specifically in Table 1, the accuracy of the final model is not altered when the weight fusion technique is applied to any of the tested data sets. This means that the training of models that have inherited a combination of weights from two previous networks do not get trapped in local minima, but neither does it improve the final performance of the hyperparameter optimization algorithm. This is to be expected, since the proposed improvement is not in the result of the algorithm, but in its duration.

Table 1 and Figures 6 and 10 also show the impact of the weight merge in the duration of the fitness function in the algorithm. The average time is considerably reduced. The skewness is also greater, with all the distribution values concentrated in the lower values in the case of algorithm with weight merge.

It can be stated that the combination of weights (or weight merge) of previous models during the crossover phase does not improve the final results of the genetic algorithm of hyperparameter optimization, but accelerates the training process of the models during the fitness function.

6 Conclusions and future work

This project demonstrates that the combination of model weights in a genetic algorithm of optimization of hyperparameters prevents wastage of training time in each generation. Regardless of the number of rows and columns of the weight matrices to be combined, the training information is preserved at each step of the algorithm, accelerating the training process of the fitness functions and the genetic algorithm.

The training of each model in a generation is independent, which enables this process to be carried out in parallel. The possibility of combining this technique with distributed neural network training systems where different nodes train the same neural network with different datasets can be studied [13]. In this case, each node could train a different network as part of the genetic algorithm with its corresponding part of the dataset, and randomly combine the weights of the best nodes. This allows distributed data systems to train and perform hyperparameter optimization in synchronization to each other.

This project has used 2-dimensional layers, such as fully dense layers, but 3 or n-dimensional layers such as convolutional or recurrent [14] were not studied. Hence, the combination of layers of different number of dimensions has been not tested and should be studied in future works.

References

[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for Hyper-Parameter Optimization," 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[2] V. Mallawaarachchi, "Introduction to genetic algorithms- including example code," July 2017. [Online]. Available: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3/

[3] A. Gad, "Genetic algorithm implementation in python," July 2018. [Online]. Available: https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6/

[4] J. N. D. Gupta and R. S. Sexton, "Comparing backpropagation with a genetic algorithm for neural network training," *Omega*, vol. 27, no. 6, pp. 679–684, December 1999. [Online]. Available: https://ideas.repec.org/a/eee/jomega/v27y1999i6p679-684.html

[5] D. Montana, "Neural network weight selection using genetic algorithms," 2002.

[6] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, ser. MLHPC '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2834892.2834896. ISBN 9781450340069. [Online]. Available: https://doi.org/10.1145/2834892.2834896

[7] L. Xie and A. Yuille, "Genetic cnn," 2017.

[8] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, p. 3840–3854, Sep 2020. doi: 10.1109/tcyb.2020.2983860. [Online]. Available: http://dx.doi.org/10.1109/TCYB.2020.2983860

[9] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," 2017.

[10] A. Håkansson, "Portal of research methods and methodologies for research projects and degree projects," in *Proceedings of the International Conference on Frontiers in Education : Computer Science and Computer Engineering FECS'13*. CSREA Press U.S.A, 2013. ISBN 1-60132-243-7 pp. 67–73, qC 20131210. [Online]. Available: http://www.world-academy-of-science.org/worldcomp13/ws

[11] Y. LeCun, C. Cortes, and C. J. Bruges, "The mnist database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[12] E. Gaztañaga, R. Croft, and G. Dalton, "Variance, skewness and kurtosis: results from the apm cluster redshift survey and model predictions," *Monthly Notices of the Royal Astronomical Society*, vol. 276, pp. 336–346, 1995.

[13] H. Mustafa and D. Dasgupta, "Distributed genetic algorithm to big data clustering," ser. 2016 IEEE Symposium Series on Computational Intelligence (SSCI). Athens, Greece: IEEE, 2015. doi: 10.1109/SSCI.2016.7849864. [Online]. Available: https://www.researchgate.net/publication/313807643_Distributed_genetic_algorithm_to_big_data_clustering

[14] D. Caetano, A. Cardoso, E. Naves, and E. Lamounier, "An experiment on the use of genetic algorithms for topology selection in deep learning." Hindawi, 2019. doi: 10.1155/2019/3217542. [Online]. Available: https://www.hindawi.com/journals/jece/2019/3217542/