# Hyperparameter tuning using Quantum Genetic Algorithms

1st Athanasios Lentzas
*School of Informatics*
*Aristotle University of Thessaloniki*
Thessaloniki, Greece
alentzas@csd.auth.gr

2nd Christoforos Nalmpantis
*School of Informatics*
*Aristotle University of Thessaloniki*
Thessaloniki, Greece
christofn@csd.auth.gr

3rd Dimitris Vrakas
*School of Informatics*
*Aristotle University of Thessaloniki*
Thessaloniki, Greece
dvrakas@csd.auth.gr

***Abstract*—Correctly tuning the hyperparameters of a machine learning model can improve classification results. Typically hyperparameter tuning is made by humans and experience is needed to fine tune them. Algorithmic approaches have been extensively studied in the literature and can find better results. In our work we employ a quantum genetic algorithm to address the hyperparameter optimization problem. The algorithm is based on qudits instead of qubits, allowing more available states. Experiments were performed on two datasets MNIST and CIFAR10 and results were compared against classic genetic algorithms.**

***Index Terms*—hyperparameter tuning, quantum genetic algorithms, evolutionary programming, machine learning, optimization.**

## I. Introduction

In machine learning, correctly tuning the several parameters of a model could affect its performance. This process is usually carried out by hand, by observing the results and progressively refining the parameters. Hand tuning requires experience, constant attendance and can consume a fair amount of time. Algorithmic approaches have been proposed in the literature [9] allowing the automation of the procedure, reducing time required and yielding better results.

Evolutionary programming has been extensively used in optimization problems [16]. Hyperparameter tuning is an optimization problem, where the best value for each parameter is selected in order to increase the accuracy of the model [13]. Genetic algorithms have been used to tune parameters on various machine learning models, from deep neural networks [15], [20], to support vector machines [2], [3].

A variation of genetic algorithm, based on quantum physics have been proposed in the literature [11]. Published papers have shown that quantum genetic algorithms have faster convergence while still maintain a global search ability.

A plethora of quantum-inspired evolutionary algorithms have been published [21]. In this paper, Quantum Genetic algorithm was employed to address the hyperparameter tuning problem. A variation of the classical implementation was used, based on qudits. Our approach was evaluated on the MNIST [8] and CIFAR10 [18] datasets. Results obtained were compared against results from classic genetic algorithm.

Section 2 covers quantum genetic algorithm. Both qubit and qudit based algorithms are presented and discussed. Section 3 describes the setup of the experiments conducted. Section 4 shows the result of the experiments and the efficiency of quantum genetic algorithm on hyperparameter optimization. The paper concludes with future work on Section 5.

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH–CREATE–INNOVATE (project code:95699 - Energy Controlling Voice Enabled Intelligent Smart Home Ecosystem)

## II. Quantum Genetic Algorithms

Genetic algorithm (GA) is an evolutionary type of algorithm extensively used in optimization problems [4]. The main idea behind GA is to mimic evolutionary process as seen in nature. An initial population of chromosomes (made from genes) is created and evaluated. The best individuals are then selected as parents and new chromosomes are created. During the breed step, a mutation can occur, changing the value of a gene.

Inspired by Quantum Computing, and based on quantum mechanics, Quantum Genetic Algorithm (QGA) was proposed [6], [11]. The main characteristic of QGA is the faster convergence to the local best, while performing global search. Instead of bits, the smallest unit of information is a qubit [5]. A qubit can be on the '0' or '1' state, just like a normal bit, or on the superposition state. When on superposition, the qubit is on both states simultaneously, but collapses to one of them when observed [11]. The state of a qubit can be described by (1), where $\alpha$ and $\beta$ are complex numbers and represent the probability amplitudes of each state.

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

When observing the state of a qubit, $|\alpha|^2$ and $|\beta|^2$ are the probabilities that the qubit is on the '0' or the '1' state respectively. It is guaranteed that the normalization state (2) is satisfied.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2}$$

A chromosome in QGA is a set of N qubits. The chromosome's state vector represents all the information of the quantum system. As qubits are always on the superposition state until observed, a chromosome that has *m* qubit can

represent $2^m$ states. In order to represent the same number of states on classic GA, $2^m$ chromosomes are needed.

Instead of the crossover operation used in GA, in QGA a quantum gate is used. The simplest form of a quantum gate, that performs the phase rotation can be seen in (3).

$$R(\theta) = \begin{matrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{matrix} \tag{3}$$

Applying the quantum gate on the initial state vector $|q\rangle$ (1) a new state vector $|q'\rangle$ is created (4). Quantum inspired genetic algorithms have also been extended by implementing GA operators, such as quantum disaster and mutation [19], as well as exploiting parallelization [7] reducing computing time.

$$R(\theta)|q'\rangle = \begin{matrix} \cos(\theta_0 + \theta) \\ \sin(\theta_0 + \theta) \end{matrix} \tag{4}$$

### A. Qudit Quantum Genetic Algorithm

Instead of the classical QGA using qubits, an algorithm using qudits [12] to encode a chromosome can be used. A qudit, is a quantum unit of information of *n* states, able to be in any superposition of those. The simplest form of many valued logic is the ternary quantum logic using qutrits. Qutrit based QGA was evaluated and compared against QGA using qubits [17]. Results shown that QGA using qutrits had a faster convergence compared to conventional QGA.

A chromosome based on qudits can be represented as a matrix of *N* qudits, each with μ states (5). An individual is determined by N qudits, with (6) representing the state of the *i* qudit. After initialization, the qudit can be in any of the available states with equal probability. The initial state representation of the qudit can be seen in (7).

$$\begin{matrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{N-1} & \alpha_N \\ \beta_1 & \beta_2 & \cdots & \beta_{N-1} & \beta_N \\ \gamma_1 & \gamma_2 & \cdots & \gamma_{N-1} & \gamma_N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_1 & \mu_2 & \cdots & \mu_{N-1} & \mu_N \end{matrix} \tag{5}$$

$$\{\alpha_i, \beta_i, ..., \mu_i\} \tag{6}$$

$$|q\rangle = \frac{1}{\sqrt{\mu}}|0\rangle + \frac{1}{\sqrt{\mu}}|1\rangle + \cdots \frac{1}{\sqrt{\mu}}|\mu\rangle \tag{7}$$

After the initialization of the population, each chromosome is observed. Observation forces each qudit to collapse to one of the available states. Evolution of the generation is based on the modification of the probability amplitudes. The best chromosome of the generation is chosen and each qudit's state is examined. For every individual on the population, each qudit is modified by increasing the probability amplitude of the state matching the best chromosome, while decreasing the rest. As a result after evolution, the states that produced the best solution are more likely to appear. By decreasing the probability amplitudes of non optimal states, it is guaranteed that (2) is not violated.

A parameter *r* is used to denote the decrease rate of probability amplitudes of non-optimal values. Parameter *r* can be adaptive or static. According to [17] static rate could reduce the convergence speed of the algorithm, thus an adaptive rate was employed. The aforementioned process simulates a quantum rotation, with the angle of the rotation determined by the rate the probability amplitudes change. As the value of *r* increases, the rotation angle decreases and vice versa.

### B. Quantum Disaster

Evolution process, as described above, may cause the QGA to fall into a local optimal solution. In order to escape from local optimal solutions quantum disaster was employed [10]. During quantum disaster a number of chromosomes, except for the best, are re-initialized. This is achieved by changing the probability amplitudes of each qudit, resulting in the initial state shown on (7). The result of the aforementioned procedure is the extension of the search area for optimal value.

Quantum disaster was evaluated in order to justify its usage. Experiments were performed on CIFAR10 dataset. QGA with and without quantum disaster were implemented. Both algorithms run for 10 generations and the experiment was repeated 10 times. As seen on Fig. 1 the average best solution per generation was better when quantum disaster was enabled. Additionally, the best solution found while using quantum disaster, was approximately 3% better. The main difference was that the best solution was found on every run when quantum disaster was on. Without quantum disaster, the best solution was only found once, as the algorithm was trapped on local optimal solutions.

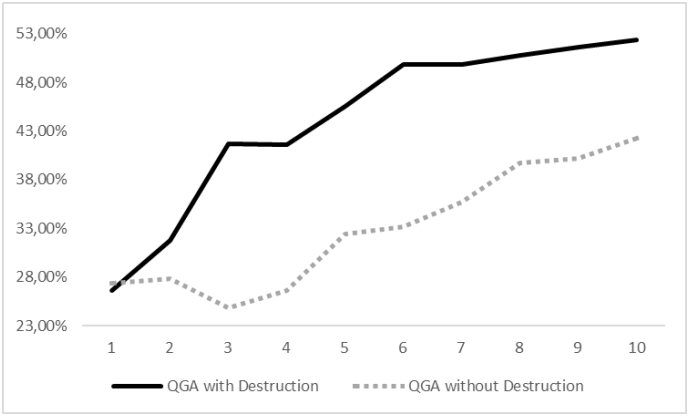


Figure 1. Average best solution per generation with and without Quantum Destruction

As already mentioned, QGA can converge to the local solution while performing global search. Quantum disaster operation promotes that ability. Comparing the average fitness of GA and QGA (see Fig 2), the global search functionality can be seen. The average fitness curve for classic GA is smoother compared to QGA. As individuals from the QGA population search different areas for optimal solution, the average fitness fluctuates.

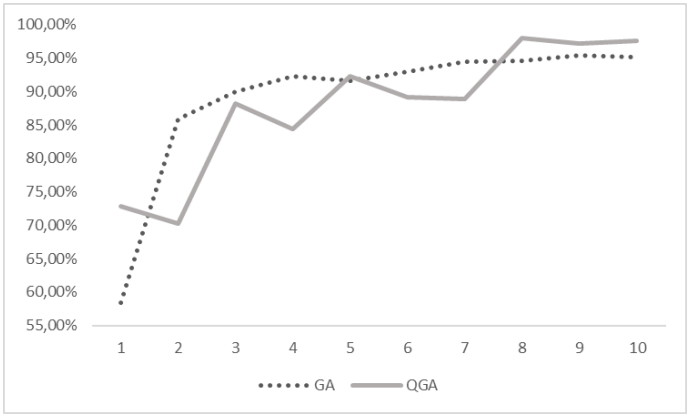


Figure 2. Average fitness of GA & QGA population (10 evolution periods)

## III. Our approach

In our work hyperparameter tuning was performed using QGA based on qudits. Instead of a predefined number of states, commonly used in related work, our approach works with any number of states. Each parameter may have different number of available states. As a result the search space can be expanded resulting in better results. The number of states is limited by hardware and constrained by available time - the bigger the search space, the longer time required.

Our approach was evaluated against GA. QGA based on qutrits [17] and binary QGA [11] could not be compared with the method proposed on this paper. As the aforementioned methods require three and two states respectively, the search space would be limited resulting in non comparable results. N-state QGA were implemented with a wide range of possible values for each parameter. As a result, the value chosen for each parameters was closer to optimal.

### A. Experiments

Hyperparameter tuning using QGA experiments were performed on a PC equipped with Intel i9 CPU, 64 GB Ram, Nvidia Titan XP GPU running Ubuntu. The relevant algorithms were implemented on Python programming language, using Keras with Tensorflow backend [1].

Two datasets were used for the experiments, the MNIST and CIFAR10. The hyperparameters tuned were: the number of neurons, number of layers, activation function and optimizer. Available values for each parameter can be seen on Table-I. One of the assumptions made on the experiments was that each layer had the same number of neurons.

Both GA and QGA were implemented and evaluated on the same datasets. Classic GA were implemented with 0.2 chance for mutation, 0.4 of the population retained after each generation and 0.1 probability of a rejected network to remain in the population. During evolution, two different random chromosomes from the retained population were chosen as parents and two children were bred. The process was repeated until the number of children bred were the same as the chromosomes rejected. Each gene of the chromosome represented the value of one of the parameters.

Implementation of the QGA was based on the qudit approach already discussed. The chromosome consisted of four qudits. The available states of each qudit were the values presented on Table-I. The probability of quantum disaster was set to 0.1. The value of $r$ was adaptive, changing if the best solution remained the same in successive generations. Initial probability amplitudes were set to $\frac{1}{\sqrt{n}}$ for each state, where $n$ was the number of available states of the qudit.

Hyperparameter tuning was performed on a Multilayer Perceptron (MLP). The accuracy of the trained MLP was used as fitness function. Since the optimal values of the parameters were unknown beforehand, both algorithms run for 10 generations and the overall best network was returned as optimal solution. Population was set to 20 chromosomes for both GA and QGA. For each algorithm, experiments were repeated 20 times. The top 10 chromosomes and the average accuracy of the generation were logged.

## IV. Results

Results obtained from the experiments showed that QGA performed better compared to GA (Table II). On MNIST dataset the performance of both algorithms was similar. The overall best solution returned by the QGA was slightly better but found one generation after the overall best returned by GA. On CIFAR10 dataset, QGA outperformed the GA. The best solution had an accuracy score of 56.69% compared to 53.18%. In addition to that, it was found on the 4th generation compared to the 7th.

The two algorithms returned different solutions for both datasets (see Table III). As GA are prone to local optimum entrapment, the best solution depended on the initial population. If the best individual on the first generation was close to a local optimal solution, the implemented genetic algorithm could not always escape and converge to the global best.

Examining the average of the best solution on each generation QGA outperformed GA on each trial. As seen on Fig. 3 and Fig. 4, for MNIST and CIFAR10 datasets respectively, solutions provided by QGA was always better compared to GA. Additionally, on CIFAR10 dataset (Fig. 4) it is clear that, most of the times, the best solution was found earlier compared to GA. Finding the best solution (or one that is close to the best) as soon as possible is crucial when the search space is big and the optimal values are unknown, since it could potentially reduce the algorithm's generations. Execution time for both algorithms was similar and QGA did not add a significant computational overhead.

## V. Conclusion

Hyperparameter tuning is a research area with increased interest. In this paper we investigated the use of quantum genetic algorithm on the specific problem. QGA were implemented along with quantum disaster operation. QGA with and without quantum destruction were evaluated before implementing the aforementioned operation. Using quantum disaster allowed the

Table I
AVAILABLE VALUES FOR EACH PARAMETER.

| Parameter | Values | Number of States |
|---|---|---|
| Number of neurons | 8, 16, 32, 64, 128, 256, 300, 512, 768, 850, 1024, 2048 | 12 |
| Number of layers | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 | 15 |
| Activation function | relu, elu, tanh, sigmoid, softmax, selu, expontential, linear, softplus, softsign | 10 |
| Optimizer | rmsprop, adam, sgd, adagrad, adadelta, adamax, nadam | 7 |

Table II
RESULTS FOR MNIST & CIFAR10 DATASETS

| Algorithm | MNIST Best | MNIST Generation | CIFAR10 Best | CIFAR10 Generation |
|---|---|---|---|---|
| GA | 98.51% | **6th** | 53.18% | 8th |
| QGA | **98.68%** | 7th | **56.69%** | **4th** |

Table III
BEST PARAMETERS RETURNED

| Parameter | GA | | QGA | |
|---|---|---|---|---|
| | MNIST | CIFAR10 | MNIST | CIFAR10 |
| Number of Neurons | 1800 | 256 | 1200 | 512 |
| Number of Layers | 2 | 1 | 2 | 8 |
| Activation | relu | sigmoid | sigmoid | softplus |
| Optimizer | adagrad | sgd | adamax | adamax |

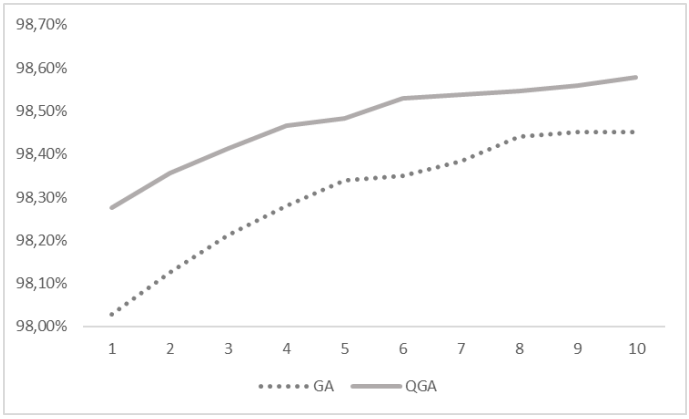


Figure 3. Average best solution per generation for MNIST dataset

algorithm to avoid local optimal solutions and converge to global best.

Results showed that QGA converged to the local solution while still performing global search, thus escaping local optimal. N-state QGA were employed with different number of states for each parameter tuned. The best solution found on both datasets (MNIST and CIFAR10) was better and was found earlier (generation wise) compared to classic GA.

Compared to classic QGA and QGA based on qutrits, N-state QGA are more suitable for hyperparameter tuning. As the number of possible states for each parameter varies, a state independent approach is recommended. In our work, the search space can be, in theory, expanded indefinitely. The only limitations are hardware and execution time.

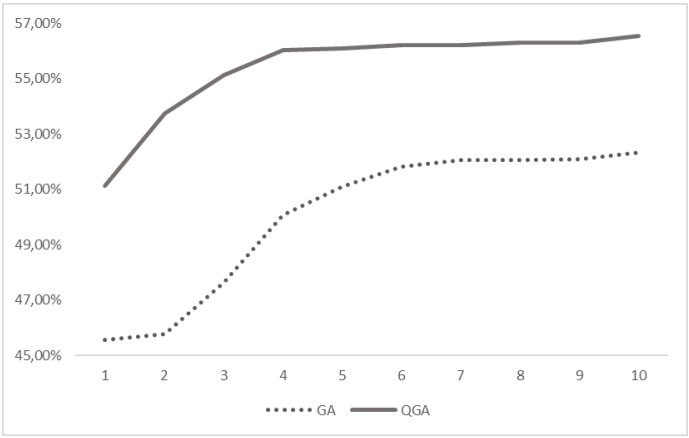


Figure 4. Average best solution per generation for CIFAR10 dataset

## VI. FUTURE WORK

In our work the evolutionary algorithms did not alter the topology of the MLP. There are methods proposed in the literature that employ GA not only to tune the various parameters of a neural network, but also alter it's architecture and topology [14]. QGA could potentially improve such techniques and further investigation is needed.

Experiments were only performed on a MLP classifier. Future work should focus on classifiers with more complex layer architecture such as convolution neural networks. A proper representation of the problem has to be proposed as different types of layers are available (max pooling layer, convolution layer etc). In addition to that, common layer architecture must be taken into consideration, i.e. a convolution layer is usually followed by a max pooling layer.

Additionally, it is necessary to perform a detailed analysis on the parameter $r$, used to modify the probability amplitudes. Although work on that area has already been published [17], further investigation is required. The value of $r$ determines how fast the algorithm switches to local search. Thus, it is important to explore how to properly change it during the execution of the algorithm, in order to achieve better results.

## ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

## REFERENCES

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 265–283. USENIX Association, Savannah, GA (2016)

[2] Cohen, G., Hilario, M., Geissbuhler, A.: Model selection for support vector classifiers via genetic algorithms. an application to medical decision support. In: Barreiro, J.M., Martín-Sánchez, F., Maojo, V., Sanz, F. (eds.) Biological and Medical Data Analysis. pp. 200–211. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)

[3] Friedrichs, F., Igel, C.: Evolutionary tuning of multiple svm parameters. Neurocomputing **64**, 107 – 117 (2005), trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004

[4] Goldberg, D.E., Holland, J.H.: Genetic Algorithms in Search, Optimization, and Machine Learning David E. Goldberg The University of Alabama T. Machine Learning (1979)

[5] Kendon, V.: Quantum computing. In: Computational Complexity: Theory, Techniques, and Applications (2013)

[6] Kuk-Hyun Han, Jong-Hwan Kim: Genetic quantum algorithm and its application to combinatorial optimization problem. In: Proceedings of the 2000 Congress on Evolutionary Computation. (2002)

[7] Kuk-Hyun Han, Kui-Hong Park, Ci-Ho Lee, Jong-Hwan Kim: Parallel quantum-inspired genetic algorithm for combinatorial optimization problem (2002). https://doi.org/10.1109/cec.2001.934358

[8] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998). https://doi.org/10.1109/5.726791

[9] Luo, G.: A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Network Modeling Analysis in Health Informatics and Bioinformatics **5**(1), 18 (May 2016). https://doi.org/10.1007/s13721-016-0125-6

[10] Miao, H., Wang, H., Deng, Z.: Quantum genetic algorithm and its application in power system reactive power optimization. In: CIS 2009 - 2009 International Conference on Computational Intelligence and Security (2009). https://doi.org/10.1109/CIS.2009.133

[11] Narayanan, A., Moore, M.: Quantum-inspired genetic algorithms (2002)

[12] Nisbet-Jones, P.B.R., Dilley, J., Holleczek, A., Barter, O., Kuhn, A.: Photonic qubits, qutrits and ququads accurately prepared and delivered on demand. New Journal of Physics (2013). https://doi.org/10.1088/1367-2630/15/5/053007

[13] Sammut, C., Webb, G.I.: Hyperparameter Optimization. In: Encyclopedia of Machine Learning and Data Mining, pp. 625–625. Springer US, Boston, MA (2017)

[14] Stanley, K.O., Miikkulainen, R.: Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. pp. 569–577. GECCO'02, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002)

[15] Suganuma, M., Shirakawa, S., Nagao, T.: A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 497–504. GECCO '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071229

[16] Taylor, C.E.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Complex Adaptive Systems. John H. Holland. The Quarterly Review of Biology (2004)

[17] Tkachuk, V.: Quantum Genetic Algorithm Based on Qutrits and Its Application. Mathematical Problems in Engineering (2018). https://doi.org/10.1155/2018/8614073

[18] Tobergte, D.R., Curtis, S.: Learning Multiple Layers of Features from Tiny Images. Journal of Chemical Information and Modeling (2013). https://doi.org/10.1017/CBO9781107415324.004

[19] Wang, H., Liu, J., Zhi, J., Fu, C.: The Improvement of Quantum Genetic Algorithm and Its Application on Function Optimization. Mathematical Problems in Engineering (2013)

[20] Young, S.R., Rose, D.C., Karnowski, T.P., Lim, S.H., Patton, R.M.: Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. pp. 4:1–4:5. MLHPC '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2834892.2834896

[21] Zhang, G.: Quantum-inspired evolutionary algorithms: A survey and empirical study. Journal of Heuristics **17**(3), 303–351 (Jun 2011). https://doi.org/10.1007/s10732-010-9136-0