On the Hardness of Parameter Optimization of Convolution Neural Networks Using Genetic Algorithm and Machine Learning

Hyeon-Chang Lee
Dept of Computer Science
Kwangwoon University
Seoul, Republic of Korea
qzecwxad@naver.com

Dong-Pil Yu
Dept of Computer Science
Kwangwoon University
Seoul, Republic of Korea
yoodongphil@naver.com

Yong-Hyuk Kim
Dept of Computer Science
Kwangwoon University
Seoul, Republic of Korea
yhdfly@kw.ac.kr

ABSTRACT

We introduce a method for optimizing parameters in convolution neural network (CNN) using a genetic algorithm (GA). In the experiment, 11 CNN parameters were chosen and considered as one chromosome. We generated 150 datasets were created by arbitrarily changing the parameters. Among approximately 30 types of models with the highest cross validation, the dataset trained with a random forest model was used as the fitness function in our GA, and the optimized parameter was obtained. To improve the GA, we attempted to filter data and amplify training steps. The randomly revised parameters showed insignificant results, but the final 10 parameter sets showed 67.4% accuracy, which was 13.7% higher than that of the dataset obtained randomly. Among these, it showed a parameter that improved by 1.7% compared to that of the existing dataset.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**; *Machine learning*;

KEYWORDS

Generic Algorithm, machine learing, hyper parameter tuning

ACM Reference Format:
Hyeon-Chang Lee, Dong-Pil Yu, and Yong-Hyuk Kim. 2018. On the Hardness of Parameter Optimization of Convolution Neural Networks Using Genetic Algorithm and Machine Learning. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3208772

1 INTRODUCTION

Parameter optimization is a typical challenge faced in the field of deep learning, which does not have clear guidelines. Despite research focusing on the effects of parameter tuning using algorithms [1, 2], it has been still more common to manually or arbitrarily change parameters [3]. However, these approaches turn out to be time consuming tasks that require experiential knowledge [4]. With the development of improved computation performance, we no longer have to use manual or random search and we can use a heuristic approach. In line with these changes, attempts to tune parameters using genetic algorithm (GA) were used to discover optimized model parameters. By using GA, we attempted parameter optimization of a convolution neural network (CNN) model which classifies audio files. An ideal approach will be to optimize the accuracy with a fitness function after testing and training the model. However, this approach incurs huge costs. To tackle this, a dataset was created with random values, and a machine learning model was built to approximate accuracy. We replace this model with a fitness function and find the optimal parameters using the GA.

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 Initial Model

Computations were performed on a PC with an Intel Core i7-6850K @ 3.60 GHz CPU, 4 NVIDIA GeForce GTX 1080 Ti GPUs, and 64 GB RAM. In the experiment, the CNN model given in TensorFlow Speech Recognition Challenge[1] was used. The given model was used to classify audio files and contained preprocess and CNN portions that included parameters. The preprocess portion contained nine parameters, including the ones where the audios were shifted or audios were mixed with noises. Nine parameters were added for each CNN filter, such as width, height, count, max pool, and drop out. The convolution number was arbitrarily fixed to 3. Each parameter had a default value, resulting in 75.32% accuracy. The initial GA approach used all parameters as chromosomes, and the training results with the given model parameters were used as a fitness function. However, determining the fitness functions took 8 hours on average, which means that applying the GA would take approximately 100 years. To solve this problem, the fitness or the number of fitness functions were reduced. The fitness was reduced by half every 125 generations by changing the GA, resulting in similar performance. However, we used a GA to decrease the computation time by half compared to the one given above with a fitness value of 46,875. In addition, the existing model was trained 18,000 times, while the GA training was limited to 5,000 times, and the dataset was reduced. In spite of these attempts, the computational challenges were difficult to overcome. These GA approaches were

[1] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3208772

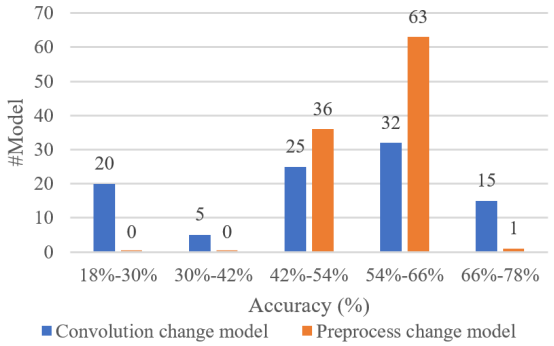

Figure 1: Accuracy-Model histogram according to parameter change

difficult to use in real experiments, so a similar approach close to these methods was used in the experiment. To figure out which parameters had higher impact, we used a method to make changes in some parameters. One hundred data points were randomly chosen and compared. In Figure 1, the parameters with CNN changes had higher impact, and GA was used by incorporating some of the CNN parameters. We used 11 parameters such as width and height. The fitness functions about an hour per iteration. This was revised to save more time as illustrated in Section 2.2.

2.2 GA Using Machine Learning

The GA used 70% probability of one-point crossover and 39% probability of mutation. The population size was 250 and was generated 1,000 times. The fitness function used an approximate model close to the actual model. For sampling, the parameters were given arbitrarily, and 150 data points were generated for a week. Each accuracy value was calculated. The average for randomly generated 150 data points was 53.68% and the standard deviation was 16.25%.
After training with machine learning, the root mean square error was calculated with 10-fold cross validation for each model. In Table 1, the best-performing model was the random forest (RF) model, so we used this model as a fitness function for approximation. The highest RF model value in the random dataset was 72.13%. The GA replaced with the RF model took around 10 minutes. The average value of RF model of the parameters found was 72.56%, the average accuracy was 67.39%, and the standard deviation was 12.12%. Compared to the model with 150 randomly selected data points, the model searched with GA showed 13.71% higher accuracy. Moreover, the highest accuracy found in a model was 79.91%, which is 1.7%

Table 1: Ten-fold cross validation of 150 data

Machine learing model	Root mean square error (%)
Random forest	10.23
Random commitee	11.11
Linear regression	13.12
Support vector regression (SVR)	13.16
Random tree	13.59

Table 2: Ten-fold cross validation trained with selected 100 data

Model	Root mean square error (%)
SVR	5.74
Random forest	5.79
Linear regression	5.89
Gaussian processes	6.07
Random commitee	6.29

higher than the existing model. However, the result of training the model with a default training step for 18,000 times showed 4.66% lower accuracy (75.25%). In conclusion, the model found with GA was found to be optimal within the initial setting. In addition, to reduce cross validation, the dataset was created by filtering data whose accuracy was lower than 50%. In Table 2, it is illustrated that the cross validation improved significantly, but the result of the GA shows reduced accuracy. Using biased data remarkably reduced the performance.

3 CONCLUSIONS

We attempted parameter optimization by using GA with a heuristic approach, which is different from the existing manual search and random search. The experiment was limited due to the limited available computational power. The parameters searched with GA showed higher average accuracy than that of the models from the random dataset. However, there was a low probability of finding a model with accuracy of 10%. To improve accuracy, we increased the training steps, which resulted in reduced accuracy. Increasing the number of training steps in a model used to improve the performance of the fitness function decreased the accuracy of the model. The GA in this experiment only found the optimal values for the given environment. In future research, we will try to minimize these limitations by using a computer with higher performance. Although in this experiment, we had obtained 150 data points in a week due to the schedule of the experiment, this can be also considered as a small sample. In the following research, we will use much more data. Furthermore, all parameters including training steps will be used as chromosomes. Based on these modifications, we expect to test an approach using GA and machine learning.

ACKNOWLEDGMENTS

This research was supported by a grant [KCG-01-2017-05] through the Disaster and Safety Management Institute funded by Korea Coast Guard of Korean government.

REFERENCES

[1] B. Kegl J. Bergstar, R. Bardenet and Y. Bengio. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*. 2546–2554.
[2] Y. Bengio J. Bergstar, R. Bardenet and B. Kegl. 2011. Implementations of algorithms for hyper-parameter optimization. In *NIPS Workshop on Bayesian Optimization*. 29–33.
[3] Y. Bengio J. Bergstra. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13 (2012), 281–305.
[4] R.P. Adams J. Snoek, H. Larochelle. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*. 2951–2959.