Hyperparameter Optimization for Deep Learning-based Automatic Melanoma Diagnosis System

Takashi NAGAOKA*, #

Abstract Deep learning is widely used in the development of automatic diagnosis systems for melanoma. However, there are some parameters called hyperparameters which should be set arbitrarily. Optimum setting of hyperparameters is challenging. The dermoscopic images on the database are trained on GoogLeNet. The hyperparameters verified in this study were random seed, solver type, base learning rate, epoch, and batch size. By using a genetic algorithm, these hyperparameters were optimized to obtain higher validation accuracy than other methods such as brute force or Bayesian optimization. The highest validation accuracy was 89.75%. The best hyperparameter settings were: 2 for random seed, RMSProp for solver type, 0.0001 for base learning rate, 30 for epoch, 32 for batch size, and 368 seconds for training time. Using the genetic algorithm, we successfully set the hyperparameters for efficient deep learning. Using the system developed in this study, we plan to search for a broader range of hyperparameters and identify multiple groups including lesions other than melanoma.

Keywords: deep learning, hyperparameters, melanoma, GoogLeNet, genetic algorithm.

Adv Biomed Eng. 9: pp. 225–232, 2020.

1. Introduction
Melanoma is a type of malignant skin tumor with poor prognosis, and the number of patients affected is increasing, mainly in Europe and the United States. Early detection is critical because melanoma can be almost completely cured if detected early [1–3]. However, melanoma continues to be diagnosed depending on the subjective opinions of doctors. Therefore, a highly accurate and quantitative diagnostic index is required.

Deep learning is widely used in the development of automatic diagnosis systems for melanoma. Esteva et al. [4] classified nearly 130,000 clinical images into three categories using deep learning and achieved an accuracy of 72.1%. Fujisawa et al. [5] also reported an accuracy of 92.4% after identifying 6,000 dermoscopic images with deep learning. Therefore, the automated diagnosis of melanoma by deep learning is very promising.

However, deep learning refers to certain parameters called hyperparameters, which are set arbitrarily. Hyperparameter settings have rich variations, and the use of inappropriate parameters decreases diagnostic performance. Identifying the optimum setting of hyperparameters is challenging.

Several approaches have been devised for optimization of the various parameters of deep learning. Lorenzo et al. [6] used Particle Swarm Optimization, and Young et al. [7] applied a genetic algorithm (GA) to optimize deep learning network structures. These reports addressed the optimization of the deep learning structure, not the hyperparameters. However, there is evidence that these optimization methods can be used to tune the various parameters of deep learning. In machine learning in general, grid search (which is equivalent to brute force), random search, Bayesian optimization, and GA are often used for hyperparameter optimization [8–10]. The random search is a method to make the search more efficient by adding randomness to the brute force. Bayesian optimization is a method of determining the next search parameters based on the results of the previous search. The computational cost of grid search is high, while random search, Bayesian optimization, and GA are uncertain due to randomness.
In this paper, we report the results of a search for the optimal hyperparameters for an automatic melanoma diagnosis system using deep learning, using a GA that does not easily fall into local stability. This study is expected to improve melanoma differentiation performance through optimization of hyperparameters for image discrimination by deep learning.

2. Methods

2.1 Dataset

In this study, the dermoscopic images in the HAM10000 database [11] provided by the International Skin Imaging Collaboration (ISIC, https://www.isic-archive.com/) were used. The database contains dermoscopic images of 6,705 nevi, 1,113 melanomas, and 2,197 other pigmented lesions. In this study, 1,000 nevus and 1,000 melanoma dermoscopic images were randomly extracted from the database. The image size of the database is 600 × 450 (width × height). In this study, the images were reduced to a size of 224 × 168, and random noise was added to the top and bottom 28 pixels to match the input image size for deep learning (224 × 224).

2.2 Deep learning

In this study, GoogLeNet Inception v1 [12] trained with the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset [13] was used as the deep learning network structure. Harangi et al. [14] also trained GoogLeNet with the HAM10000 database, and reported accuracies of 77.2% and 82.5% for nevi and melanoma, respectively. Chaturvedi et al. [15] trained MobileNet [16] with the HAM10000 database and reported accuracies of 83.15%, 69.23%, and 69.23% for seven categories of skin lesions, nevi, and melanoma, respectively. Even when using datasets other than HAM10000, a melanoma diagnosis using dermoscopic images by deep learning has a reported accuracy of approximately 80% [17–19].

The development environment of this study was as follows: NVCaffe ver. 0.15.14, Python ver 2.7.6, Ubuntu operating system 18.04.3 LTS, Intel i9-7980XE (2.6 GHz) central processing unit, 64 GB memory, and NVIDIA Quadro P5000 (16 GB memory) graphics processing unit (GPU).

In this study, the same training and validation datasets were used to verify the change in diagnostic performance due to hyperparameters. The dermoscopic images of each class were randomly assigned to the training and verification datasets at a ratio of 8:2. Early stopping [20] was not adopted, and the network was trained until the final epoch.

The hyperparameters verified in this study were random seed, solver type, base learning rate, epoch, and batch size. The random seed denotes the initial value of the pseudo random number, while the solver type refers to the type of gradient method used for deep learning. Compared to the standard steepest descent method, several methods have been proposed to prevent divergence and false convergence by adding randomness and retaining the past gradient [21]. The base learning rate denotes the initial value of the network parameter learning rate, while the epoch indicates the number of times each data included in the training dataset is used for training. The batch size denotes the number of data read in one training. Note that the learning rate is multiplied by 0.1 for every 33% of the epoch. Table 1 shows the range of values for each hyperparameter. The total number of hyperparameter combinations in this study was 1,200. Each parameter could have more values, but was limited to the range shown in Table 1 for comparison with brute force. It took approximately four days to train the network for all combinations with our study environment.

2.3 Genetic algorithm

Figure 1 shows the flow of the GA used in this study when the population is 5. The five hyperparameters described in the previous section of each individual in this study were genes. The network was trained and evaluated according to the hyperparameters of each individual, and an evaluation value (described later) was calculated. After the population was sorted in the order of the evaluation value, it was processed in the order of selection, crossover, and mutation.

In the selection section, the individual with the best evaluation value was retained and propagated into two individuals. The total population was maintained by culling the worst evaluated individual.

Except for the best evaluated individual, the individual with the closest evaluation value was crossed by a single point. The crossover position was set at random.

For every parameter in every individual, there was a certain probability that the parameter would be mutated. Finally, all the parameters in all the individuals were mu-

| Hyperparameter         | min | max | step | subtotal |
|------------------------|-----|-----|------|----------|
| random seed            | 0   | 4   | +1   | 5        |
| solver type*           | 0   | 4   | +1   | 5        |
| base learning rate     | 0.1 | 0.0001 | x0.1 | 4        |
| epoch                  | 10  | 30  | +10  | 3        |
| batch size             | 8   | 64  | x2   | 4        |

* 0: Adam, 1: Nesterov, 2: SGD, 3: AdaGrad, 4: RMSProp
tated with a preset probability. The population obtained after these processes became the next generation population. The same processing was repeated up to the preset number of generations.

The evaluation value was obtained using Equation (1),

$$\text{Evaluation value} = (1 - \text{acc}_t) \times 1,000,000 + (1 - \text{acc}_v) \times 1,000 + t/10,000.0$$

(1)

where $\text{acc}_t$ is the training accuracy, $\text{acc}_v$ is the validation accuracy, and $t$ is the time required for training.

In this study, the smaller the evaluation value obtained using equation (1), the better was the evaluation of the individual. First, $(1/\text{uni} - 1/\text{uni} \times \text{acc}_t)$ was rounded to four decimal places and multiplied by one million. The closer the training accuracy is to 1, the smaller is the evaluation value. Since the largest factor was multiplied, the highest priority was given to the convergence of training by the final epoch. This operation excluded hyperparameter combinations that did not converge in training. Next, $(1/\text{uni} - 1/\text{uni} \times \text{acc}_v)$ was also rounded to four decimal places and multiplied by one thousand, and added to the evaluation value. An individual with a higher validation accuracy had a lower evaluation value. Therefore, based on this evaluation value, the individual whose training converged by the final epoch and exhibited the highest generalization capability was selected. Finally, the training time $t$ was rounded to the fifth decimal place and divided by 10,000, and added to the evaluation value. It was expected that the parameter with the shortest training time among all the parameters with the same general performance would be selected. By minimizing this value, the hyperparameter combination with the highest generalization capability and the shortest training time was selected while the training converged to the final epoch.

The hyperparameters required for the GA were the number of individuals, mutation rate, and number of generations. Matlab R2019a (The MathWorks, Inc., Massachusetts, USA) was used for statistical analyses of the evaluation values and parameters. The significance level was set at 5%.

For comparison with our proposed approach, Bayesian optimization was considered in addition to the brute force. Up to 1,000 generations of hyperparameter combinations which had the minimum evaluation value shown in Equation 1 were searched. An open source program written in python was used for Bayesian optimization [22].

3. Results

3.1 Deep learning

The network was trained using all combinations of the hyperparameters. In this study, if the training accuracy exceeded 0.95, the training was considered to have converged. The results indicated that the training converged at 539 out of 1,200 combinations of the hyperparameters. Figure 2 shows the probability frequency distribution of the validation accuracy for all combinations of
the hyperparameters for which the training converged. The highest validation accuracy was 89.8%. The best hyperparameter settings were identified as follows: 2 for random seed, RMSProp for solver type, 0.0001 for base learning rate, 30 for epoch, 32 for batch size, and 368 seconds for training time. On the other hand, the worst validation accuracy was 74.0%. The evaluation value obtained from these parameters was 102.5368. The worst hyperparameters settings were as follows: 0 for random seed, SGD for solver type, 0.1 for base learning rate, 0 for epoch, 8 for batch size, and 169 seconds for training time. The average, median, and mode of the validation accuracy were 85.0, 85.3, and 85.8%, respectively. The validation accuracies of approximately 98% of the hyperparameter combinations fell in the 80 to 90% range.

Validation accuracy was analyzed using one-way analysis of variance (one-way ANOVA). The results are shown in Fig. 3. Significant differences were found for several combinations of solver type, base learning rate, and batch size. Training time was also analyzed using one-way ANOVA. The results are shown in Fig. 4. Significant differences were also found for several combinations of base learning rate, epoch, and batch size. The numerical values of the analyses shown in Figures 3 and 4 are summarized in Table 2.

3.2 Genetic algorithm

Figure 5 shows the analysis results of the GA by one-way ANOVA. The populations ranged from 3 to 13 in steps of 2, while the mutation rates ranged from 5 to 30% in steps of 5%. For each combination, the GA was repeated for up to 1,000 generations to evaluate the number of generations with the above-mentioned best hyperparameters. For a mutation rate of 5%, the best parameters could not be found up to the last generation. If the mutation rate exceeded 10%, the best parameters could be found up to the last generation. The best parameters could be found at the earliest possible time (approximately 65 generations) with a mutation rate of 20%. The results of the one-way ANOVA indicated that it was significantly slower to find the best parameters when the mutation rate was only 10%. When the mutation rate was fixed at 20%, the best parameters were found for all the

![Fig. 2](image)

**Fig. 2** Probability frequency distribution of validation accuracy (solid line, left axis) and cumulative frequency distribution (dotted line, right axis) for all combinations of hyperparameters for which training converged.

![Fig. 3](image)

**Fig. 3** Analysis results of validation accuracy by one-way ANOVA (*: p < 0.05).
Fig. 4  Analysis results of training time by one-way ANOVA (*: p < 0.05).

Table 2  Analysis results by one-way ANOVA.

| (a) random seed | (b) solver type | (c) base learning rate | (d) epoch | (e) batch size |
|-----------------|-----------------|------------------------|-----------|---------------|
| 0 | 1 | 2 | 3 | 4 | Adam | Nesterov | SGD | AdaGrad | RMSProp | 0.0001 | 0.001 | 0.01 | 0.1 | 10 | 20 | 30 | 8 | 16 | 32 | 64 |
| 3rd quartile | 0.860 | 0.865 | 0.863 | 0.868 | 0.863 | 0.867 | 0.858 | 0.863 | 0.858 | 0.872 | 0.868 | 0.861 | 0.855 | 0.794 | 0.863 | 0.865 | 0.863 | 0.853 | 0.863 | 0.868 | 0.863 |
| median | 0.853 | 0.853 | 0.855 | 0.855 | 0.853 | 0.856 | 0.850 | 0.853 | 0.848 | 0.864 | 0.855 | 0.853 | 0.841 | 0.774 | 0.853 | 0.855 | 0.853 | 0.843 | 0.855 | 0.855 | 0.855 |
| average | 0.847 | 0.848 | 0.853 | 0.853 | 0.849 | 0.851 | 0.851 | 0.848 | 0.843 | 0.856 | 0.857 | 0.852 | 0.832 | 0.774 | 0.849 | 0.852 | 0.848 | 0.836 | 0.853 | 0.857 | 0.855 |
| 1st quartile | 0.838 | 0.843 | 0.843 | 0.843 | 0.838 | 0.838 | 0.845 | 0.840 | 0.835 | 0.848 | 0.845 | 0.845 | 0.813 | 0.752 | 0.843 | 0.845 | 0.840 | 0.826 | 0.845 | 0.848 | 0.848 |

3rd quartile | 369 | 369 | 369 | 369 | 368 | 370 | 368 | 368 | 369 | 368 | 369 | 368 | 374 | 503 | 169 | 335 | 501 | 502 | 407 | 368 | 350 |
| median | 335 | 335 | 336 | 335 | 335 | 338 | 278 | 335 | 336 | 304 | 335 | 273 | 350 | 502 | 138 | 248 | 373 | 336 | 273 | 247 | 235 |
| average | 297 | 303 | 298 | 305 | 301 | 304 | 292 | 298 | 315 | 297 | 299 | 288 | 329 | 403 | 143 | 274 | 409 | 356 | 301 | 277 | 261 |
| 1st quartile | 186 | 234 | 194 | 235 | 235 | 235 | 234 | 235 | 234 | 171 | 171 | 176 | 246 | 294 | 124 | 237 | 360 | 170 | 272 | 246 | 234 |

Fig. 5  Analysis results of the genetic algorithm by one-way ANOVA (*: p < 0.05).
populations. Population 7 had the smallest total number of trainings (approximately 450 times and equivalent to 65 generations), defined as the product of the population and the number of generations for which the best parameter could be found. The results of one-way ANOVA showed significant differences between populations 3 and 7.

Bayesian optimizations up to 1,000 generations (1,000 trainings) were repeated 50 times, but the best combination by brute force was never selected in the method we adopted. The average and standard deviation of the evaluation values by Bayesian optimization were 135.4527 and 6.6097, respectively. Compared to the best evaluation value (102.5368), this implied that \( acc_v \) was approximately 2.2% lower in the Bayesian optimization compared to the GA and the brute force. Furthermore, the average processing time per generation in our system was 243.3 microseconds for the GA and 6.21 seconds for Bayesian optimization.

4. Discussion

4.1 Deep learning

The deep learning training was conducted with 2,000 dermoscopic images. The result of this training showed that differences of up to 15% in accuracy existed for all combinations of the hyperparameters even though the input dataset was the same. This result also confirms the hyperparameter dependence of the deep learning.

Each hyperparameter is discussed below. Only five random seeds were tested, but no differences in accuracy among the random seeds were observed. Most deep learning frameworks can perform deterministic operations. The results of this study confirm that the network can be evaluated by fixing an appropriate random number.

For the dataset used in this study, RMSProp was determined to be the best solver type. However, only AdaGrad showed a significantly lower validation accuracy than RMSProp, and no significant difference was observed among the other solver types. Moreover, no significant difference in training time among the solver types was found. It should be noted that compared to other solver types, RMSProp had slightly more hyperparameter combinations for which training did not converge. Therefore, solver types other than AdaGrad are considered to be suitable for the dataset of this study.

The base learning rate has a significant effect on accuracy. Excessive learning rate makes training difficult to converge. Even if the training converges, a tendency exists for low validation accuracy and long training time. On the other hand, a low learning rate requires a long epoch. Therefore, a learning rate of either 0.001 or 0.0001 is considered appropriate for the study dataset.

Insufficient epochs increase the possibility that training will not converge, and excessive epochs require extra training time. Therefore, it can be said that the epoch is one of the most influential hyperparameters for efficient training. For our study dataset, the number of epochs should be set to at least 20, and preferably 30.

It is generally considered that the larger the batch size, the more efficient is the training. Also, as the batch size increases, the number of trainings per epoch decreases, and thus, training time also decreases. The results of this study follow these conventions. Although the batch size depends on the input image size and the memory size of the GPU, it is preferable to set the batch size as large as possible.

4.2 Genetic algorithm

As mentioned above, the GA contains two types of hyperparameters: population and mutation rate. For our dataset, a population of 7 and a mutation rate of 20% are optimal.

Since the initial values of all genes are determined at random, and the parameters change randomly due to crossover or mutation, the path to the optimal parameters is also highly random. Figure 6 shows the frequency distribution of the generation that reaches the optimal parameters. In this study, the optimal parameters were found the earliest in the first generation and the latest in the 271st generation (1,897 trainings). The worst case required more trainings than the total number of combinations (1,200). The mean number of generations that reached the optimal parameters was 64.48 generations (451 trainings), the corresponding median was 40.5 generations (284 trainings), and the corresponding mode was 34 generations (238 trainings). Therefore, it is considered that the optimal hyperparameters can be efficiently found by the GA. Since the best parameters are not actually known, the GA needs to be repeated for more generations. According to the results of this study, the optimum parameters can be found by repeating the
GA for up to 50 generations (350 trainings) or at most 100 generations (700 trainings). Since the GA changes the parameters at random, the same combination of parameters may be selected. Figure 7 shows the number of new parameter combinations for each generation. Approximately 350 parameter combinations are selected in 100 generations (700 trainings). This effect becomes more pronounced as the generation advances. Therefore, this means that advanced generations finish training sooner than earlier generations. Specifically, as shown in Figure 7, the training of the 100th generation is completed when approximately half of the first generation has finished. Therefore, even if the GA is repeated for many generations, the optimal parameter can be found earlier than in the case of the brute force. The GA shows a shorter processing time than the brute force and higher diagnostic performance than the Bayesian optimization. Also, the genetic algorithm uses relatively simple methods such as sorting and searching, and thus processing is very fast. In the method we employed, optimization with GA was more than 25,000 times faster than the Bayesian optimization. This result also strongly supports the usefulness of the genetic algorithm.

5. Conclusion

We believe that AI approach is useful for various clinical situations; its usability and regulatory issue have been discussed in detail [23]. In this study, we developed a method for optimizing deep learning hyperparameters to develop an automatic diagnosis system for melanoma using dermoscopic images. Dermoscopic images of melanoma and benign nevi were extracted from the ISIC database to create a deep learning dataset. Using GoogLeNet for deep learning and a GA to search for the optimal hyperparameters, we succeeded in discriminating between melanoma and benign nevi with an accuracy of 89.75%. Using the system developed in this study, we plan to search for a broader range of hyperparameters and identify multiple groups including lesions other than melanoma.

References

1. Thompson JF, Scolyer RA, Kefford RF: Cutaneous melanoma. Lancet. 365(9460), 687–701, 2005.
2. American Joint Committee Cancer Staging Manual. 6th edn. Springer-Verlag, New York, 2002.
3. Erdmann F, Lorret-Tieulent J, Schüz J, Zeeh H, Greinert R, Breitbart EW, Bray F: International trends in the incidence of malignant melanoma 1953-2008- are recent generations at higher or lower risk? Int J Cancer. 132(2), 385–400, 2013.
4. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S: Dermatologist-level classification of skin cancer with deep neural networks. Nature. 542(7639), 115–118, 2017.
5. Fujisawa Y, Otomo Y, Ogata Y, Nakamura Y, Fujita R, Ishitsuka Y, Watanabe R, Okiyama N, Ohara K, Fujimoto M: Deep learning-based, computer-aided classifier developed with a small dataset of clinical images surpasses board-certified dermatologists in skin tumour diagnosis. Br J Dermatol. 180(2), 373–381, 2018.
6. Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kavulok, Luciano Sanchez Ramos, José Ranilla Pastor: Particle swarm optimization for hyperparameter selection in deep neural networks. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO ’17). Association for Computing Machinery, New York, NY, USA, pp. 481–488, 2017.
7. Steven R. Young, Derek C. Rose, Thomas P. Karnowski, Seung-Hwan Lim, Robert M. Patton: Optimizing deep learning hyperparameters through an evolutionary algorithm. In Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments (MLHPC’15). Association for Computing Machinery, New York, NY, USA, Article 4, 1–5, 2015.
8. Bergstra JS, Bardenet R, Bengio Y, Kégl B: Algorithms for hyperparameter optimization. Adv Neural Inf Process Syst. 24, 2546–2554, 2011.
9. Liashchynskyi P, Liashchynskyi P: Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv: 1912.06059, 2019.
10. Snoek J, Larochelle H, Adams RP: Practical Bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst. 24, 2951–2959, 2012.
11. Tschandl P, Rosendahl C, Kittler H: The Ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci Data. 5, 180161, 2018.
12. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A: Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 1–9, 2015.
13. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A C, Fei-Fei L: ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis. 115, 211–252, 2015.
14. Harangi B, Baran A, Hajdu A: Classification of skin lesions using an ensemble of deep neural networks. 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). 2575–2578, 2018.
15. Chaturvedi SS, Gupta K, Prasad P: Skin lesion analyzer: An efficient seven-way multi-class skin cancer classification using Mo-
16. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv: 1704.04861, 2017.

17. Ramalakhan K, Shang Y: A mobile automated skin lesion classification system. International Conference on Tools with Artificial Intelligence (ICTAI). 138–141, 2011.

18. Ruiz D, Berenguer V, Soriano A, Sánchez B: A decision support system for the diagnosis of melanoma: A comparative approach. Expert Syst Appl. 38(12), 15217–15223, 2011.

19. Kawahara J, Hamarneh G: Multi-resolution-Tract CNN with Hybrid Pretrained and Skin-Lesion Trained Layers. MLMI@ MICCAI 2016, 2016.

20. Prechelt L: Early Stopping - But When? Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg, 53–67, 2012.

21. Ruder S: An overview of gradient descent optimization algorithms. arXiv preprint arXiv: 1609.04747, 2016.

22. Fernando Nogueira: Bayesian Optimization: Open source constrained global optimization tool for Python. https://github.com/fmfn/BayesianOptimization, 2014.

23. Chinzei K, Shimizu A, Mori K, Harada K, Takeda H, Hashizume M, Ishizuka M, Kato N, Kawamori R, Kyo S, Nagata K, Yamane T, Sakuma I, Ohe K, Mitsuishi M: Regulatory science on AI-based medical devices and systems. Adv Biomed Eng. 7, 118–123, 2018.

Takashi NAGAOKA
Takashi NAGAOKA received the Ph.D. degree in engineering from Waseda University, Japan, in 2005. He is lecturer at Faculty of Biology-Oriented Science and Technology, Kindai University, Japan, from 2015. His research fields are dermatology, hyperspectral imaging, and medical systems. He is member of the Japanese Society for Medical and Biological Engineering, the Society of Instrument and Control Engineers, and the Japanese Skin Cancer Society.