Automatic Hyper-parameter Tuning with Genetic Algorithms for Multi-Class Brain Tumor Detection

Eashan Singh
Computer Science, BS
eashan.singh1@knights.ucf.edu

1. PROBLEM DEFINITION

This research aims to develop a deep learning model for multi-class brain tumor detection with the use of genetic algorithms (GAs) [13] for optimized hyper-parameter tuning and model architecture [11, 12, 17, 18, 19]. Genetic Algorithms have been found to be effective at several optimization problems, taking inspiration from evolutionary processes and a "survival of the fittest" approach. Data augmentation techniques will also be used for improving the generalizability of our model and increasing the size of our dataset. [14, 15] Motivations for this project are three-fold:

1. Deep learning models can become highly complex and fine-tuning hyper-parameters can become a very time intensive task. Research has shown genetic algorithms(GA) offer an efficient way to find optimality of parameters [16] and for exploration of the hyper-parameter search space. [21]
2. This project aims to improve the models generalizability by exploiting genetic algorithm's (GAs) resistance to overfitting [22]. Combined with data augmentation techniques to increase the diversity and size of the dataset, the goal is to improve model performance on unseen data.
3. Leveraging pre-trained models via transfer learning can potentially help improve model accuracy and prove an alternative for other resource-intensive deep learning techniques, making it more viable in a clinical environment. [23, 24]

2. APPROACH

2.1 Dataset. The problem we aim to solve is a multi-class brain tumor identification problem [1, 2, 3, 4, 5]. The dataset which will be used for this study can be found here: https://github.com/sartajbhuvaji/brain-tumor-classification-dataset

The data is of brain MRI images and each MRI image falls under one of the 4 classes:

1) Glioma Tumor
2) Meningioma Tumor
3) Pituitary Tumor
4) No Tumor

2.2 Approach. Here is a proposed systematic approach to performing this research:

1. Pre-processing a brain tumor MRI dataset by employing normalization, de-noising, and re-sampling techniques and then segmenting regions with existence of tumors.
2. Employ a CNN architecture to develop a *baseline* model for various forms of brain tumor detection. Train the model by hand and record model performance observing key metrics like *Train/Val Loss, Accuracy, F1 score, Precision, and Recall*.
3. Implement a genetic algorithm to evolve a population of CNN hyper-parameters
4. Observe the model performance with the optimized hyper-parameters and analyze the differences in the recorded metrics.
5. Apply data augmentations to see if we can improve the results
6. Make conclusions on the ability of the genetic algorithm to improve model generalizability and performance. Discuss the added computational overhead and whether it warrants its use.

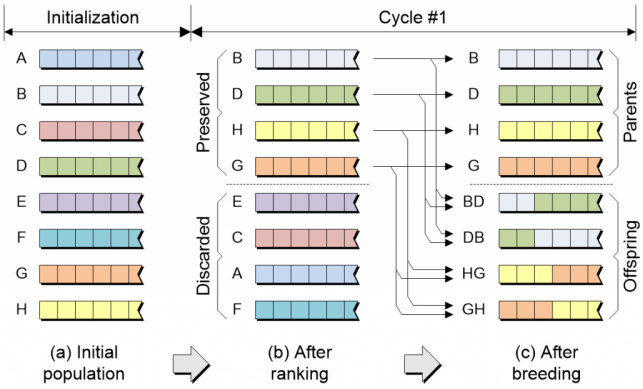
In this study, I will aim to optimize the following CNN hyper-parameters:

- Learning Rate
- Batch Size
- Number of filters

The selection of these hyper-parameters were made because these parameters are fairly universal to most Deep Neural Networks and thus findings from this work can hopefully translate meaningfully to other model architectures aside from the one used in this paper. Considerations were also made to allow the genetic algorithm to vary the architecture of the model itself by changing the number of layers, but increasingly large models would significantly increase the runtime of this process. Additionally, the improvements of other hyper-parameters can also be better observed on a model architecture kept constant. Varying the number of layers may affect the convergence time of the other 3 parameters as well because the optimal learning rate for example may not be mutually exclusive from the number of nodes or model parameters. Studies of optimized model architecture are worth exploring in future studies to help avoid confounding results.

2.3 Genetic Algorithm Theory. This paper makes use of Genetic Algorithms(GAs) as a way to evolve a population of hyper-parameter encodings. GAs are a class of optimization algorithms which take inspiration from natural selection and survival of the fittest strategy.

- An initial population is made based on an encoding of candidate solutions and an elitism structure is used to select parent candidates for reproduction.
- Parents undergo crossover and mutation and child offspring become the parents of the next generation
- The algorithm runs until an optimal solution or stopping criteria is met.



[33] Figure 1. Genetic Algorithm high-level overview. In this diagram, we observe the seeding of an initial population of candidate solutions. The solutions are ranked and a selection strategy or "elitism" strategy is employed to encourage the best solutions to reproduce. The parents then undergo crossover and mutations(genetic operators) to create new offspring which become the parents of the next generation.

A candidate solution for this problem is made with the following encoding:

Encoding E<learning rate, batch size, # of filters>

The encoding is made with the (Distributed Evolutionary Algorithms in Python) DEAP library. For a given encoding E, the learning rate, batch size, and the number of filters are concatenated together to form a sample solution. The learning rate is allowed to take on values of (0.0001, 0.001, 0.005, and 0.01). These learning rates were selected due to common model tuning practice. The batch size can take on values of (16, 32, and 64), and filters can take on values of (8, 16, and 32). Filter sizes were kept small for a smaller number of model weights.

- Initialize a population with **N** encodings **E**
- For **R** runs:
  - For **G** generations:
    - Evaluate the fitness of all candidate solutions(evaluating hyperparameters on the classification task)
    - Employ a **tournament selection** strategy to pick parents for reproduction
    - Perform **crossover**
    - Perform **mutation**
    - Offspring of generation **g** become population of generation **g+1**
- Run until stopping criterion is met or **R** runs.

Figure 2. Standard run of the GA(pseudocode)

2.4 Model Architecture. For this problem, I made a custom CNN model to accommodate for the number of functional evaluations made in this problem. I wanted to keep the number of parameters for the model under 3,000,000 to allow the genetic algorithm to run in manageable durations for training. This is in the ballpark of MobileNet-V2 in terms of number of parameters. However the architecture I've made is slightly more flexible for training thus why I use this one. Below is a table of the model layer by layer with the layer type, output shape, and number of parameters.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
conv2d_1 (Conv2D)	(None, 146, 146, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 73, 73, 64)	0
dropout (Dropout)	(None, 73, 73, 64)	0
conv2d_2 (Conv2D)	(None, 71, 71, 64)	36928
conv2d_3 (Conv2D)	(None, 69, 69, 64)	36928
dropout_1 (Dropout)	(None, 69, 69, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 34, 34, 64)	0
dropout_2 (Dropout)	(None, 34, 34, 64)	0

conv2d_4 (Conv2D)	(None, 32, 32, 128)	73856
conv2d_5 (Conv2D)	(None, 30, 30, 128)	147584
conv2d_6 (Conv2D)	(None, 28, 28, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 128)	0
dropout_3 (Dropout)	(None, 14, 14, 128)	0
conv2d_7 (Conv2D)	(None, 12, 12, 256)	295168
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 256)	0
dropout_4 (Dropout)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 256)	2359552
dropout_5 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1028

Total params: 2,942,020

Trainable params: 2,942,020

Non-trainable params: 0

Table 1. Model Architecture

3. EXPERIMENTS

3.1 Experimentation. In an attempt to optimize the parameter of our CNN, experimentation was done to find the ideal hyper-parameters of our genetic algorithm (GA). Testing was done on the following GA hyper-parameters:

1. *Population size (10,15,20,25)*
2. *Mutation type* (uniform random, gaussian, polynomial)
3. *Mutation rates* (0, 0.001, 0.01, 0.1)
4. *Crossover type* (single-point, two-point, blend, simplex, linear, simulated binary)
5. *Crossover rates* (0.01, 0.05, 0.1, 0.5, 0.9)
6. *Selection strategy* (tournament, rank, proportional)

This is the list of GA hyper-parameters that were the most effective at the tumor classification task and used for the rest of the paper.

Population Size	20
Mutation Type	Gaussian
Mutation Rate	0.1
Crossover Type	Uniform
Crossover Rate	0.5
Selection Strategy	Tournament selection, size 3
Termination Criteria	10 generations

Table 2. Genetic Algorithm Hyper-parameters

Testing was done on population sizes of 10, 15, 20, and 25. Results are generally better for larger population sizes, as was the case here for population size 25. However the performance difference between population size 20 and 25 was very minimal therefore to reduce runtime complexity, a population size of 20 was used. Three mutation strategies, uniform, gaussian, and polynomial were tried and results were best for mutations following a gaussian distribution at a rate of 0.1.

Crossover was tested for single-point, two-point, blend, simplex, linear, simulated binary and results were best for single-point crossovers at a rate of 0.5. The selection strategy I employed was a tournament selection of size 3, which did comparably well to other selection strategies.

The evaluation of a baseline model was done first to see what the best results I could achieve were without any help from optimization algorithms for **30 epochs**. This consisted of me trying to optimize the model by hand, typical of what most researchers do when fine-tuning model parameters. The baseline in this paper is defined as my best attempt to fit the model to the data with reasonable fine-tuning hyper-parameters. Because people will fine-tune models differently, it is hard to establish a “ground truth” baseline performance because the fine-tuning process is quite subjective. Researchers place different amounts of emphasis on the fine tuning process and therefore one of the goals is to see how much we can improve the performance of our model with the assumption that I think I have done the fine-tuning to the best of my ability.

After evaluating performance on the baseline model, the GA will then simulate the evolution of hyper-parameters and record the best performance over 3 runs and 10

generations. The average of the 3 best performances in each run will be taken and used as comparison to the baseline.

3.5 Evaluation Metrics To measure model performance, the following evaluation metrics were used:

- Accuracy
- Recall
- F1 score
- ROC curve

4. RESULTS

4.1 Baseline Performance. After attempting to fine-tune the model architecture described above, I was able to achieve the best results with the following hyper-parameters:

- **Learning Rate**: 0.0001
- **Batch Size**: 64
- **# of Filters**: 32

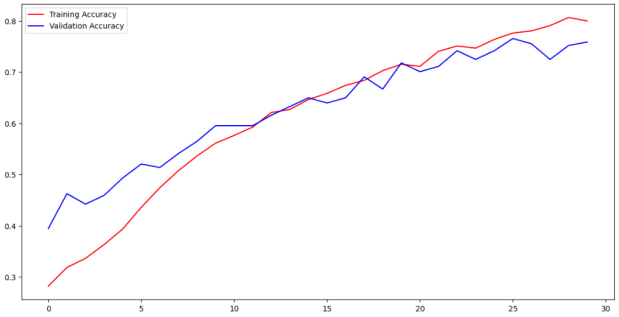


Figure 3. Baseline Training accuracy(red) and validation accuracy(blue) over 30 epochs. Best achieved training accuracy was **80.6%** percent and best validation accuracy was **76.45%.**

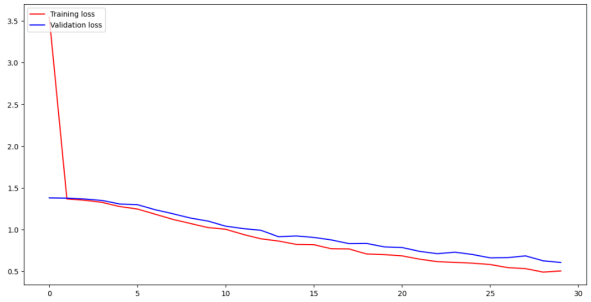


Figure 4. Training loss(red) and validation loss(blue) over 30 epochs. Best achieved training loss was 0.48 and best validation loss was 0.60.

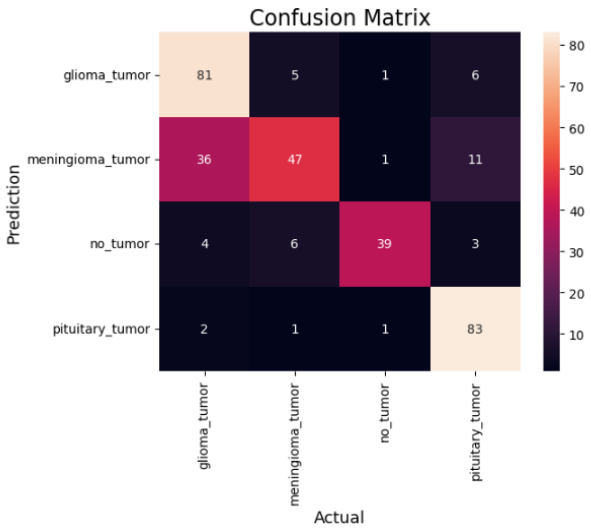


Figure 5. Baseline Confusion matrix. There are 4 labels for this task: glioma, meningioma, pituitary, and no tumor.

As we can observe from Figure 3. The validation accuracy does a good job following the training accuracy over the course of 30 epochs. The accuracy has yet to converge to a value yet but we can see the rate of change beginning to dip. Similarly, the training and validation loss in Figure 4. show that loss is still steadily decreasing over the 30 epochs and validation loss correlates well with training loss. The confusion matrix appears to do a good job with most classifications with the highest number of misclassifications occurring for glioma tumors being mistaken as meningioma. The number of "no tumor" classifications are predominantly accurate with meningioma tumors showing the most number of false positives.

4.2 Optimized GA-Model Performance. The genetic algorithm identified the following set of hyper-parameters as the best performing:

- **Learning Rate**: 0.001
- **Batch Size**: 32 or 64
- **#of Filters**: 32

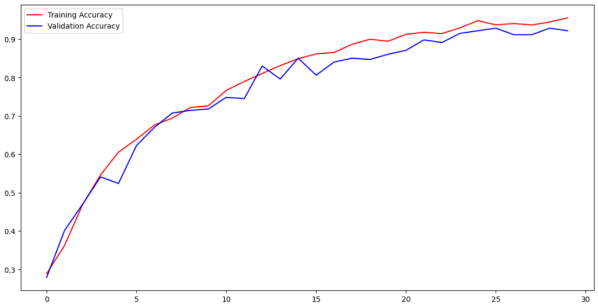


Figure 6. GA-Optimized Training accuracy(red) and validation accuracy(blue) over 30 epochs. Best achieved training accuracy was **95.54%** percent and best validation accuracy was **92.86%.**

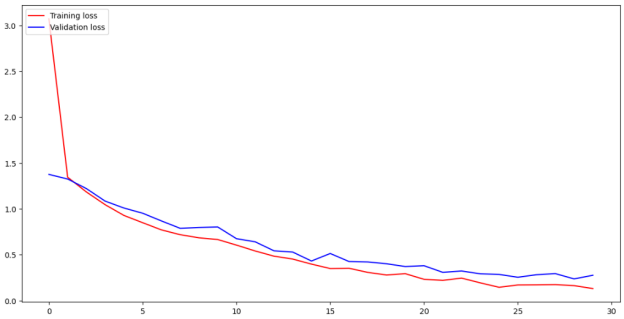


Figure 7. GA- Optimized Training loss(red) and validation loss(blue) over 30 epochs. Best achieved training loss was **0.13** and best validation loss was **0.23**.

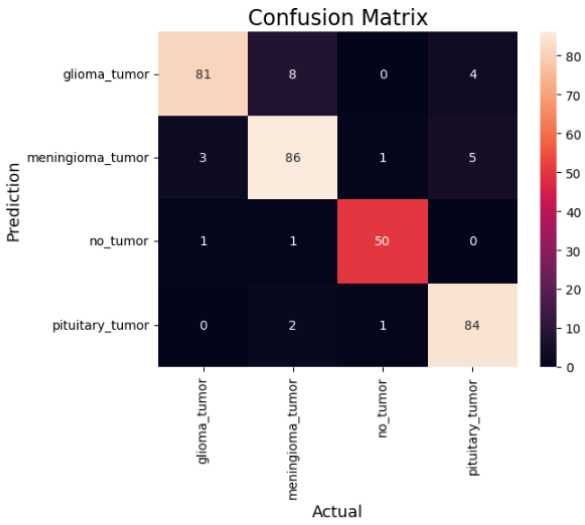


Figure 8. GA-Optimized Confusion matrix.

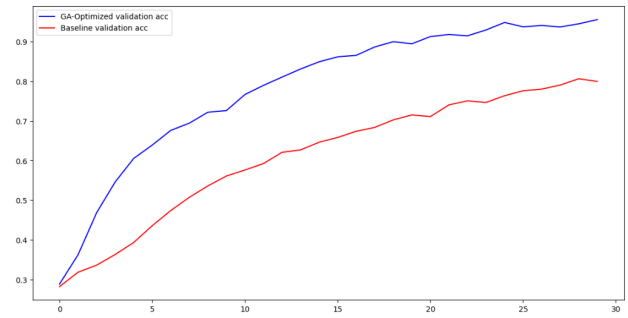


Figure 9. Comparison of Baseline vs GA-Optimized Validation Accuracy over 30 epochs. Best baseline val accuracy(red) is 76% and best GA-optimized accuracy(blue) is 93%

	Baseline	GA-Optimized
Best Accuracy	**0.7645**	**0.9286**
Precision	**0.7807**	**0.9218**
Recall	**0.7645**	**0.9204**
F1 Score	**0.755**	**0.9203**

Table 3. Key Performance Metrics Comparison.

As we observe from Figure 6. The GA-optimized model training and validation accuracy are closely aligned and have begun to converge around the 20 epoch mark. The best validation accuracy achieved for newly found hyper-parameters was almost 93%, which is considerably higher than the fine-tuning results I was able to obtain for the baseline results. Figure 7 and 9 tell a similar story as we observe the training loss and see a comparison of the two sets of hyper-parameters. The improvements in accuracy made via the baseline approach are a lot more linear than the GA-improved hyper-parameters. The improved set of hyper-parameters actually had a 10 times larger learning rate which may explain the behavior we see in Figure 9. The small learning rate in the baseline probably meant the accuracy was not able to converge fast enough in 30 epochs, which is why we see the training loss/accuracy for the baseline has yet to converge. The large search space for hyper-parameter combinations can make it difficult to identify where the model suffers and in this case, the learning rate seems to be the culprit. The model performed best on 32 filters, and 0.001 learning rate. The batch size was not as significant to model performance. Both 64 and 32 performed similarly in terms of model accuracy. Larger

batch size generally means more computation time, therefore the 32 batch size is preferred. These optimizations are impressive because we were able to improve validation accuracy from the baseline by nearly 17% in the same number of epochs. The baseline would've probably approached close to the same results as the GA-optimized hyper-parameters had it run for more epochs.

The confusion matrix in Figure 8. also shows an improvement in misclassifications. We saw earlier in the baseline performance that glioma tumors were being mistaken as meningioma but that value has gone down from 36 to 3.

5. CONCLUSION

This research aimed to improve the model performance of a CNN for brain-tumor multi-class identification by optimizing the model hyper-parameters. As we observed from the results above, the GA-optimized model was able to achieve 93% validation accuracy over 30 epochs. While it is hard to generalize that improvements in hyper-parameters for a given architecture will always be significant, using GAs to evolve quality hyper-parameters is effective at reducing the search space of your fine-tuning and can lead to very promising results. Had this experiment been done via fine tuning, the individual would have to try a total of 36 combinations of hyper-parameters manually to achieve the same results. This is a very time consuming task which requires the researcher to be online. The use of a GA for improved fine-tuning allows one to work offline while the GA fine-tunes the model. But does this warrant the added computational overhead?

While using GAs for large model architectures would be very time-consuming, for improving smaller CNN networks in applications on the edge, this approach can actually be quite useful. This is because, we often fail to exploit the full abilities of a smaller network because we do not have enough insight on the hyper-parameter search space we are working in. Even with the added overhead of running the GA for a small network, for every subsequent run of the model, we can be more confident that we are making use of computational resources efficiently because the quality of the hyper-parameters will be more vetted. For applications of deep learning models which we know will be run several times by a given client for example, we want to make sure that computation is efficient. GAs can serve as a good one time vetting of a model before it goes out in production.

The added computational overhead can be hard to justify for many real-life applications, therefore I would not recommend using this approach for all problem types. This is one metaheuristic technique for optimization and even GAs perform a lot of redundant work. For larger networks, even more so. As a future direction of this work, I would try and parallelize some of the GA computation to see if I can get significant runtime improvements. Also, I'd like to try and make the model architecture a much larger component of the hyper-parameter search space so that we know that a given architecture is efficient to begin with. The hyper-parameters can do as good as the model allows it, and this work makes the assumption that a given model architecture is capable and effective at the given task. Only after this has been confirmed does trying to reduce the hyper-parameter search space serve its utility.

6. RELATED WORK

[31] Studies performed by Tahir et al. (2021) utilized PSO optimization to automate the classification task in brain tumor identification. This study made use of GAs and found good success with this approach. My project aims to utilize these findings and make use of transfer learning and data augmentation in the process.

[17] Bahadure et al.(2018) studied a comparative approach similar to the one I wish to perform on different segmentation techniques using genetic algorithms (GA). This paper used feature extraction using GAs and this is a stretch goal for my experiment.

[9] Deepak et al.(2019) had much success using transfer learning in the classification of brain tumors. They were able to get 98% accuracy with their predictions for a 3-class classification problem. This is one motivation for integrating transfer learning in this project.

[33] Sajjad et al.(2019) "showed convincing" performance in the use of data augmentation for multi-grade classification tasks of brain tumors. Using a pre-trained CNN, they were able to attain over 95% test accuracy.

REFERENCES

[1] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694

[2] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117

[3] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression

Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)

[4] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, et al., "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection", The Cancer Imaging Archive, 2017. DOI: 10.7937/K9/TCIA.2017.KLXWJJ1Q

[5] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, et al., "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-LGG collection", The Cancer Imaging Archive, 2017. DOI: 10.7937/K9/TCIA.2017.GJQ7R0EF

[6] Abiwinanda, N., Hanif, M., Hesaputra, S. T., Handayani, A., & Mengko, T. R. (2019). Brain tumor classification using convolutional neural network. In World Congress on Medical Physics and Biomedical Engineering 2018: June 3-8, 2018, Prague, Czech Republic (Vol. 1) (pp. 183-189). Springer Singapore.

[7] Seetha, J., & Raja, S. S. (2018). Brain tumor classification using convolutional neural networks. Biomedical & Pharmacology Journal, 11(3), 1457.

[8] Paul, J. S., Plassard, A. J., Landman, B. A., & Fabbri, D. (2017, March). Deep learning for brain tumor classification. In Medical Imaging 2017: Biomedical Applications in Molecular, Structural, and Functional Imaging (Vol. 10137, pp. 253-268). SPIE.

[11] Swati, Z. N. K., Zhao, Q., Kabir, M., Ali, F., Ali, Z., Ahmed, S., & Lu, J. (2019). Brain tumor classification for MR images using transfer learning and fine-tuning. Computerized Medical Imaging and Graphics, 75, 34-46.

[12] Arbane, M., Benlamri, R., Brik, Y., & Djerioui, M. (2021, February). Transfer learning for automatic brain tumor classification using MRI images. In 2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH) (pp. 210-214). IEEE.

[13] Holland, J. H. (1992). Genetic algorithms. Scientific american, 267(1), 66-73.

[14] Shin, H. C., Tenenholtz, N. A., Rogers, J. K., Schwarz, C. G., Senjem, M. L., Gunter, J. L., ... & Michalski, M. (2018). Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In Simulation and Synthesis in Medical Imaging: Third International Workshop, SASHIMI 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings 3 (pp. 1-11). Springer International Publishing.

[15] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48.

[16] Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S. H., & Patton, R. M. (2015, November). Optimizing deep learning hyper-parameters through an evolutionary algorithm. In Proceedings of the workshop on machine learning in high-performance computing environments (pp. 1-5).

[17] Bahadure, N. B., Ray, A. K., & Thethi, H. P. (2018). Comparative approach of MRI-based brain tumor segmentation and classification using genetic algorithm. Journal of digital imaging, 31, 477-489.

[18] Chandra, G. R., & Rao, K. R. H. (2016). Tumor detection in brain using genetic algorithm. Procedia Computer Science, 79, 449-457.

[19] Anaraki, A. K., Ayati, M., & Kazemi, F. (2019). Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. biocybernetics and biomedical engineering, 39(1), 63-74.

[20] Atia N, Benzaoui A, Jacques S, Hamiane M, Kourd KE, Bouakaz A, Ouahabi A. Particle Swarm Optimization and Two-Way Fixed-Effects Analysis of Variance for Efficient Brain Tumor Segmentation. Cancers. 2022; 14(18):4399. https://doi.org/10.3390/cancers14184399

[21] Sivanandam, S. N., Deepa, S. N., Sivanandam, S. N., & Deepa, S. N. (2008). Genetic algorithms (pp. 15-37). Springer Berlin Heidelberg.

[22] Hosseini, M., Powell, M., Collins, J., Callahan-Flintoft, C., Jones, W., Bowman, H., & Wyble, B.

(2020). I tried a bunch of things: The dangers of unexpected overfitting in classification of brain data. Neuroscience & Biobehavioral Reviews, 119, 456-467.

[23] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3(1), 1-40.

[24] Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

[25] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).

[26] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2008). Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transactions on Evolutionary Computation, 13(2), 398-417.

[27] Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-aided design, 43(3), 303-315.

[28] Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., ... & Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. Medical image analysis, 35, 18-31.

[29] Cherkassky, V., & Ma, Y. (2004). Practical selection of SVM parameters and noise estimation for SVM regression. Neural networks, 17(1), 113-126.

[31] A. Bin T. Tahir, M. Attique Khan, M. Alhaisoni, J. Ali Khan, Y. Nam et al., "Deep learning and improved particle swarm optimization based multimodal brain tumor classification," Computers, Materials & Continua, vol. 68, no.1, pp. 1099–1116, 2021.

[32] Sajjad, M., Khan, S., Muhammad, K., Wu, W., Ullah, A., & Baik, S. W. (2019). Multi-grade brain tumor classification using deep CNN with extensive data augmentation. Journal of computational science, 30, 174-182.

[33] Maxfield, M. (2020, July 9). When genetic algorithms meet Artificial Intelligence. EEJournal. Retrieved April 18, 2023, from https://www.eejournal.com/article/when-genetic-algorithms-meet-artificial-intelligence/