Meta-Heuristics vs. Backpropagation: A Fresh Look at CNN Model Parameter Optimization for Image Classification

Utkarsh Mathur
umathur, Pixel Swarm
CSE 573 Project Milestone 2
University at Buffalo
umathur@buffalo.edu

Mahammad Iqbal Shaik
mahammad, Pixel Swarm
CSE 573 Project Milestone 2
University at Buffalo
mahammad@buffalo.edu

***Abstract*—Most of the Neural Network models are trained using gradient based backpropagation technique to optimize model parameters which are prone to be stuck at local optimal value rather than reaching the globally optimal parameters. There are various techniques to improve the simple Stochastic Gradient Descent (SGD) like Learning Rate Scheduling and Momentum, but these techniques does not resolve the aforementioned limitation. In this project, we aim to compare Backpropagation with Meta-Heuristic Optimization Algorithms by analyzing their performance on training CNN models for Image Classification tasks. There are a few population-based metaheuristic algorithms like Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO) which can achieve globally optimal parameters and so, we aim to check the feasibility of such optimization techniques in CNN model architectures. https://github.com/datamathur/pixel_swarm_CV_project**

I. Introduction

Since the first milestone we have made significant progress at the cost of compromise with respect to project method details. After doing some more literature review we tailored our method and execution details to accommodate lack of computational resources, insufficiency of time (with respect to the scope of the project), and executional bottlenecks.

The aim of this project is to analyze the performances of meta-heuristic optimization algorithms in contrast to gradient-based backpropagation techniques for Convolutional Neural Network model training.

Since we are trying to change the model optimization techniques we intend to perform analysis on the following two grounds:

1. Quality of model training – Based on the model inferencing performed on models trained with different optimization techniques, we will be comparing the qualities of trained model.

2. Computational Cost – Since most of the deep-CNN models are computationally expensive we will be analyzing the computational resources required by the novel optimization techniques in contrast to gradient-based backpropagation techniques.

We have performed an preliminary analysis on both these fronts and we plan to perform a few more experiments to complete our analysis.

A. Modified Method Details

1) Meta Heuristic Algorithms: In the initial project proposal we aimed to implement 4 metaheuristic algorithms as optimizers for image classification modes however we are only using 2 optimizer. We decided not to use Genetic Algorithms and Ant Colony Optimization owing to algorithm complexity, insufficient library support, and time limitation. We have developed the following optimizers:

i. Particle Swarm Optimization
ii. Grey Wolf Optimization

2) Training Dataset: The original aim was to train and test the model on ImageNet but owing to the lack of computational resources we have switched our attention to the following datasets:

i. MNIST
ii. CIFAR-10

3) Model Architectures: The original aim was to train several models like AlexNet, VGG-16, and ResNet-50 but as our datasets is that of small size images we are using LeNet-5 architecture to experiment model training for MNIST and CIFAR-10.

B. Project Status

1. We have completed our literature review by studying relevant survey papers, articles, and books.

2. We have completed writing the code of metaheuristic optimizations for model training in PyTorch.

3. We have completed checking model training for metaheuristic optimizations on a small subset of hyperparameters for each algorithm.

4. We are currently working on analysing the performance of PSO and GWO.

GitHub Repository for our project can be found at https://github.com/datamathur/pixel_swarm_CV_project

II. Methodology

Most of the deep learning libraries like PyTorch, TensorFlow, and MxNet donot have functions available for metaheuristic algorithms. Therefore, we developed these optimizers for model training in PyTorch. The literature around metaheuristic algorithms deals with execution of these optimizers for scalars and 1D vectors, however most of the Convolutional Neural Networks have parameters in 4-dimensional tensors. We address this by using model parameter and candidate parameters as collection of tensors depicting parameters of a single convolution layer.

Since we aimed to develop metaheuristic optimizers for PyTorch, we inherited torch.optim.Optimizers module to create optimizers that functions similar to the in-built PyTorch optimizers like Stochastic Gradient Descent and Adam.

By defining a callable closure() function, we we able to pass a function to calculate loss associated with each set of parameters as and when required by the optimization process. This reduces memory usage by avoiding making unnecessary instances of model class.

A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based meta-heuristic algorithm inspired by swarm behavior observed in nature such as fish and bird schooling. PSO is a simulation of a simplified social system.

The PSO class inherits Optimizers module from PyTorch and defines the step() function which is used in defining the process of selecting the best particle from a swarm of particles based on its fitness (loss) over the given model and dataset. These particles are defined by another class Particle which also has its own step() function whose aim is to perform one iteration of PSO on a single particle in the swarm.

B. Grey Wolf Optimization

Grey wolf optimizer (GWO) is a population-based meta-heuristics algorithm that simulates the leadership hierarchy and hunting mechanism of grey wolves in nature, and it's proposed by Seyedali Mirjalili et al. in 2014.

The GWO class defines Grey Wolf Optimization for PyTorch models by inheriting Optimizers module from PyTorch. The step() in GWO function is used to define and perform one iteration of training based on the fitness (loss) of all the member of the pack. In the end we store the 3 best positions instead of just one as it is the requirement of the algorithm (this depicts the social hierarchy of wolfs).

C. Model Training

Models are trained in PyTorch library using the CPU as device and Critical Cross Entropy as loss function to optimize parameters of LeNet-5 architecture.

III. Experiments & Results

A. Particle Swarm Optimization

We performed grid search hyperparameter tuning with 4 hyperparameters of PSO:

1. **Population Size** (N): Population size determines the size of the swarm of particles. We assumed [10, 20, 50, 100] values for population size.

2. **Inertial Weight** (w): Inertial weight is a hyperparameter that determines the weightage or momentum carried by position and velocity of the particle at previous step. We used [0.5, 0.75, 0.9] values for inertial weight which is recommended in the range of 0.1-1.

3. **Cognitive Coefficient** (c1): It is the hyperparamter that determines the amount of influence of current position of the particle in the next position. We assumed [1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0] for cognitive coefficient which is recommended in the range of 1.0-2.

4. **Social Coefficient** (c2): This hyperparameter determines the amount of influence of the global best position (position of the fittest particle in the swarm) in the new position of the particle. We assumed [1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0] values for Social coefficient which is recommended in the range of 1.0-2.0.

By controling these 4 hyperparameters we wanted to experiment with search space exploration v/s exploitation tradeoff. However, the best validation accuracy for MNIST dataset was 26.37% (N=100, w=0.9, c1=1.1, c2=1.2) and for CIFAR-10 was 18.97% (N=20, w=0.9, c1=1.5, c2=1.2) which is very low in comparison to SGD optimizer. (Table I and Table II)

B. Grey Wolf Optimization

We performed Grid search Hyperparamter tuning with 2 hyperparamters for GWO:

1. **Encircling Coefficient** (a = 2*(1-(i/num_epochs)): This hyperparamter determines the rate at which each wolf in the pack encircles or searches the search space for solution. It is determined by the number of epochs such that its value decreases uniformly from 2 to 0 over the training process.

2. **Pack Size** (N): This hyperparamter determines the population size of the pack and is extremely crucial in the exploration and exploitation of search space.

The best validation accuracy for MNIST dataset was 16.52% (num_epochs = 50, N = 5) and for CIFAR-10 was 12.35% (num_epochs = 20, N = 10) which is very low in comparison to SGD optimizer. (Table I and Table II)

TABLE I
Comparing Validation Loss for PSO, GWO, and SGD

Dataset	PSO Loss	GWO Loss	SGD Loss
MNIST	1.9981	2.303	1.4722
CIFAR-10	2.370	2.337	1.0433

TABLE II
Comparing Validation Accuracy for PSO, GWO, and SGD

Dataset	PSO Acc.	GWO Acc.	SGD Acc.
MNIST	26.37%	16.52%	98.99%
CIFAR-10	18.97%	12.35%	64.4%

IV. Plan for Future Work

We have completed a developing codes for optimizers and models in PyTorch and performed a few experiments which has result given us a few opportunites to understand the workings of Meta Heuristic Optimization Algorithms and Neural Networks and Convolutional Neural Networks.

Over the span of 1 week (deadline for final project submission) we aim to perform the following analysis.

1. Complete the time complexity analysis for model training and inference with PSO and GWO.

2. Hyperparameter tuning and subsequent analysis for PSO and GWO.

3. Develop an intuition behind why backpropagation work better than these global optimizers.

References

[1] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. doi: https://doi.org/10.1007/978-3-662-03315-9.

[2] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," Multimedia Tools and Applications, vol. 80, no. 5, Oct. 2020, doi: https://doi.org/10.1007/s11042-020-10139-6.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995, doi: https://doi.org/10.1109/icnn.1995.488968.

[4] T. Alam, S. Qamar, A. Dixit, and M. Benaida, "Genetic Algorithm: Reviews, Implementations, and Applications," Jun. 2020, doi: https://doi.org/10.48550/arxiv.2007.12673.

[5] A. G. Gad, "Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review," Archives of Computational Methods in Engineering, vol. 29, no. 5, pp. 2531–2561, Apr. 2022, doi: https://doi.org/10.1007/s11831-021-09694-4.

[6] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: https://doi.org/10.1109/MCI.2006.329691.

[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, Mar. 2014, doi: https://doi.org/10.1016/j.advengsoft.2013.12.007.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2009, doi: https://doi.org/10.1109/cvpr.2009.5206848

[9] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, Nov. 2012, doi: https://doi.org/10.1109/msp.2012.2211477.¡/li¿

[10] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," undefined, 2009, Accessed: May 11, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Learning-Multiple-Layers-of-Features-from-Tiny-Krizhevsky/5d90f06bb70a0a3dced62413346235c02b1aa086¡/li¿

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998, doi: https://doi.org/10.1109/5.726791.¡/li¿

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2012, doi: https://doi.org/10.1145/3065386.¡/li¿

[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv.org, Apr. 10, 2015. https://arxiv.org/abs/1409.1556

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv.org, Dec. 10, 2015. https://arxiv.org/abs/1512.03385

[15] R. Mohapatra, "rohanmohapatra/torchswarm," GitHub, Apr. 20, 2024. https://github.com/rohanmohapatra/torchswarm (accessed Apr. 20, 2024).

[16] A. P. Sansom, "Torch PSO," GitHub, Aug. 01, 2022. https://github.com/qthequartermasterman/torch_pso (accessed Apr. 20, 2024).

[17] H. Faris, "7ossam81/EvoloPy," GitHub, Apr. 20, 2024. https://github.com/7ossam81/EvoloPy/tree/master (accessed Apr. 20, 2024).

[18] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, Nov. 2012, doi: https://doi.org/10.1109/msp.2012.2211477.

[19] A. Krizhevsky, "CIFAR-10 and CIFAR-100 datasets," Toronto.edu, 2009. https://www.cs.toronto.edu/ kriz/cifar.html (accessed Apr.20, 2024)

[20] "pytorch/pytorch," GitHub, Mar. 22, 2021. https://github.com/pytorch/pytorch (accessed Apr.20, 2024)

[21] "Grey wolf optimization - Introduction," GeeksforGeeks, Mar. 16, 2021. https://www.geeksforgeeks.org/grey-wolf-optimization-introduction/ (accessed Apr. 20, 2024).

[22] "Particle Swarm Optimization (PSO) - An Overview," GeeksforGeeks, Apr. 22, 2021. https://www.geeksforgeeks.org/particle-swarm-optimization-pso-an-overview/