Hyperparameter Tuning for AI Models

Fernando Berti Cruz Nogueira, Kelvin Mock

Abstract: Considering the growing trend in AI, and specifically computer vision, it is important to ensure models are trained and run with their most effective set of hyperparameters. While the convolutional neural network (CNN) is a popular choice as a backbone, we formulate our project to apply and compare against multiple metaheuristic search-based algorithms in the problem of image recognition with CNN, with reference to some papers (Bibaeva, 2018) (Catalin Stoean, Miodrag Zivkovic, AleksandraNebojsa Bacanin, Roma Strulak-Wójcikiewicz, Milos Antonijevic, and Ruxandra Stoean, 2023).

1. Introduction

(1) The Overall Idea of the Project

Artificial Intelligence now spans its applications across multiple domains, relying heavily on machine learning models. A typical model is a complex mathematical formula, such as a polynomial, that consists of weights that act as coefficients for each term. The process of training involves finding the most suitable vector of these weights (known as **parameters**) for a given formula and dataset. However, this training process itself is controlled by another set of variables called **hyperparameters**.

Hyperparameters dictate how a model learns and include critical settings like the learning rate, batch size per iteration, and the number of epochs. Tuning these hyperparameters is a demanding task, as it is very difficult to determine the optimal combination for a given model. Finding this optimal set through manual trial-and-error is often impractical, as it would require running the entire resource-intensive training and inference process for every possible combination. This challenge of finding the best hyperparameters is a significant bottleneck in developing effective AI models.

To address this challenge, our project will experiment with a variety of metaheuristics for searching and identifying optimal hyperparameters. Owing to resources limitations, a simple dataset and pre-built model architectures are used.

(2) Why the Project is Significant

Complex neural networks are now widely applied in critical domains, from energy generation and power prediction (Catalin Stoean, Miodrag Zivkovic, AleksandraNebojsa Bacanin, Roma Strulak-Wójcikiewicz, Milos Antonijevic, and Ruxandra Stoean, 2023) *to* bioinformatics and clinical diagnosis using medical imaging (Sajjad Nematzadeh, Farzad Kiani, Mahsa Torkamanian-Afshar, Nizamettin Aydin, 2021). For these sophisticated models, traditional hyperparameter tuning methods such as exhaustive grid search (EGS) are often too slow and computationally expensive. Research has shown that metaheuristics can deliver both better performance and faster convergence than these traditional techniques (Sajjad Nematzadeh, Farzad Kiani, Mahsa Torkamanian-Afshar, Nizamettin Aydin, 2021). Therefore, using metaheuristics could be a novel way to solve such a complex or even NP-hard problem.

Recent papers propose the use of metaheuristic algorithms to optimize hyperparameters, particularly for **convolutional neural networks (CNN)**. A CNN, like LeNet-5, AlexNet, VGG, and ResNet – is widely used for processing pixel-based images with kernel/filters. Typical hyperparameters include number of layers, filter size, type of activation function (namely, ReLu), pooling type, etc (Bibaeva, 2018). Given the large set of variables to be tuned, the process significantly requires automation. Metaheuristics are perfectly suited for this task, as they excel at navigating complex search spaces. While many studies have applied these methods to simpler architectures, demonstrating their effectiveness on complex models like CNNs highlights their significance in advancing the field of AI.

(3) Relevancy: how the project is relevant to SBSE or to the applications SMT/SAT solvers for ST or SE

Traditionally, either a randomized grid or an exhaustive grid is generated to search for the optimal set of hyperparameters. As suggested by the name, a random search typically generates k random numbers for each hyperparameter. An exhaustive grid search, in a nutshell, attempts to generate all possible combinations of hyperparameters. Aside from the expensive computation, one common problem is the cause of missing intermediate values where continuous values are expected. Like the learning rate, it is expected to be a decimal number. But those searches could only statically define (i.e., hardcode) a few numbers. Can we guarantee to obtain all possible combinations? If not, are they representative enough? These questions are difficult to answer definitely. Another issue specifically with the exhaustive grid search is the computation. For larger and complex neural networks, where a very wide variety of hyperparameters is involved, the computational time and power is significantly high, because it is obtaining the grid based on a brute-force approach.

As an effective alternative to streamline the optimization process, many studies support the use of a metaheuristic search-based algorithm. The mind map below illustrates all possible types of algorithms that we could take into consideration.

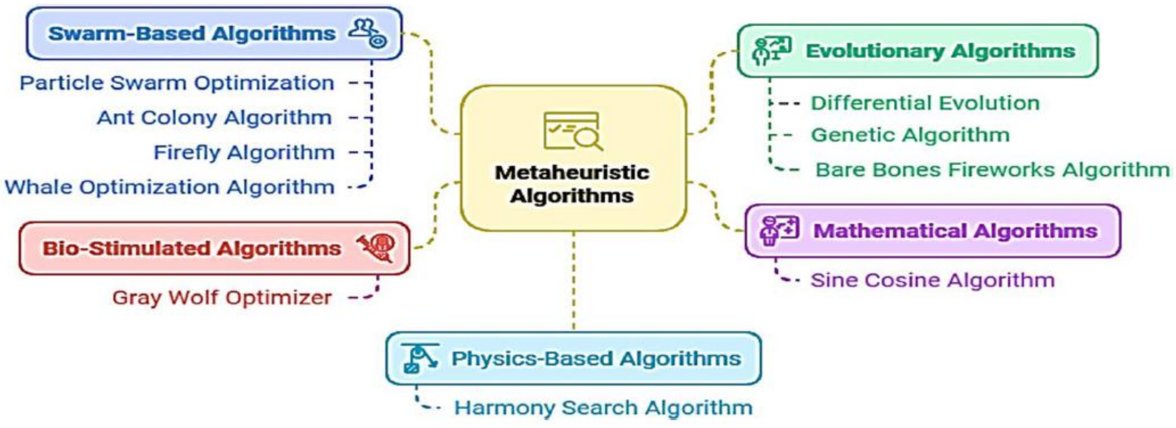

Figure 1 Metaheuristics Overview (Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad, 2025)

A strategy which uses no further models to find an optimal set of hyperparameters is called the **model-free algorithm** (Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad, 2025). It is generally exploring an environment, like the model's nature, by trial-and-error. A grid search, as mentioned above, robustly explores a finite set of values and calculates their Cartesian product. It repeatedly happens such that the search space is reduced until an optimal set is found. Random search, on the other hand, is believed to be computationally more friendly, but the chances of getting to the optimal set depends on pure luck. A novel way as a "model-free" algorithm is by using gradient descent. It works by continuously minimizing the gradient of the model's loss function, by differentiation. Theoretically, the gradients, at some point, tend to be 0, referring to the minimum point, and not on a changing slope. It requires a lot of hand calculations while those calculations are believed less applicable to real-world models. Thus, those kinds of models could be useful as a baseline in comparison with other advanced ones. In comparison to those choices of baseline, we need to identify several types of **metaheuristics**. Interestingly, **swarm-based optimization**, namely particle swarm and ant colony, attempts to spread in a stochastic manner to find an optimal solution. However, convergence could be an issue, since it could happen endlessly without being able to locate a suitable candidate, and thus, an algorithm like this should be bounded by time and solution quality with a proper metric (Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad, 2025). Similarly, firefly algorithm and whale optimization algorithm are also borrowing the idea from how fireflies attract one another and how humpback whales are being hunted, respectively. From a different perspective, the **grey wolf optimizer** is an algorithm inspired by the biological processes of how wolves encircle their victims, as a result of finding the optimal solution. It is susceptible to being stopped at a local optimum without progressing to the true global optimal solution. The **harmony search algorithm** mimics how a musician discovers superior harmonic tone, either randomly or by memory. A mathematically inspired one – **sine cosine algorithm** – has been proved in an experiment about its ability to explore interesting enough candidates from a search space, and to steer to local and even the global optimum by generating and changing random solutions. This is very similar to population-based evolutionary algorithms, like the **genetic algorithm** and **differential evolution**, which firstly selects a subset of candidates (by for example, elitism), crossovers a part of a pair of candidates, and finally mutates a random pair of candidate solutions. All those metaheuristic algorithms are outlined clearly in the abovementioned paper (Mohammed Q Ibrahim, Nazar

K. Hussein, David Guinovart, Mohammed Qaraad, 2025) which makes reproducing in specifically for hyperparameter tuning much easily.

In comparison to those choices of baseline, we need to identify several types of **metaheuristics**. Interestingly, **swarm-based optimization**, namely particle swarm and ant colony, attempts to spread in a stochastic manner to find an optimal solution. However, convergence could be an issue, since it could happen endlessly without being able to locate a suitable candidate, and thus, an algorithm like this should be bounded by time and solution quality with a proper metric (Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad, 2025). Similarly, firefly algorithm and whale optimization algorithm are also borrowing the idea from how fireflies attract one another and how humpback whales are being hunted, respectively. From a different perspective, the **grey wolf optimizer** is an algorithm inspired by the biological processes of how wolves encircle their victims, as a result of finding the optimal solution. It is susceptible to being stopped at a local optimum without progressing to the true global optimal solution. The **harmony search algorithm** mimics how a musician discovers superior harmonic tone, either randomly or by memory. A mathematically inspired one – **sine cosine algorithm** – has been proved in an experiment about its ability to explore interesting enough candidates from a search space, and to steer to local and even the global optimum by generating and changing random solutions. This is very similar to population-based evolutionary algorithms, like the **genetic algorithm** and **differential evolution**, which firstly selects a subset of candidates (by for example, elitism), crossovers a part of a pair of candidates, and finally mutates a random pair of candidate solutions. All those metaheuristic algorithms are outlined clearly in the abovementioned paper (Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad, 2025) which makes reproducing in specifically for hyperparameter tuning much easily.

2. Related Work

In your final course project report, you are expected to have a detailed discussion about related work. For the proposal stage, the related work section will be short (1 paragraph) and can be limited to the key one or two papers that you will build on, alongside the relationship between your proposal and these papers.

One paper (Bibaeva, 2018) suggests using a specialized population-based genetic algorithm, given its versatility in all sorts of datasets and model architectures but not just a very particular one. As a preparation stage, they provided a **representation** of a binary encoding of the entire CNN architecture, essentially encapsulating hyperparameters needed. The length of a layer is calculated from the count of layers and the size of the input image. Then, certain validation rules were applied on the architecture. For example, its filter size should be odd; step size not exceeding filter size; etc.

To evaluate the quality of each candidate solution, Bibaeva used a multi-objective **fitness function** that combined the model's loss and accuracy, placing a heavier weight on accuracy to prioritize predictive performance. Their work proposed a memetic algorithm which integrated a genetic algorithm with a local search component to balance global exploration and local refinement.

The study (Bibaeva, 2018) compares the memetic algorithm to simulated annealing, highlighting their trade-offs: The memetic algorithm's cubic complexity allows for a more robust search that can converge to a global optimum. In contrast, simulated annealing offers a faster, linear complexity but is limited to a predefined search space and risks getting trapped in local optima.

The work provides a good foundation, albeit limited to solely the memetic algorithm versus simulated annealing. Our research will expand on this by evaluating a broader range of metaheuristics drawn from different families against their approach. By doing so, we aim to provide a more comprehensive answer to which metaheuristic strategies are most effective for the complex task of CNN hyperparameter tuning.

3. Research Questions

Please list the research question(s) (1-2 sentences for each question) that your final project is going to tackle. You will be allowed to update your research questions later on, but I expect the research questions at the proposal stage to be valid, meaningful and interesting in relation to the project you would like to conduct.

1. Which of the selected metaheuristics algorithm demonstrates the best performance in optimizing the hyperparameters of any kinds of model architecture?
2. How do hyperparameters found by metaheuristic algorithms compared to those found by traditional methods like grid search and random search in terms of model accuracy and training time?

4. Data Set and Replication plan (if your project is about replication or if you plan to use any existing or new data sets)

Clearly describe what data source(s) you will be using for your course project. This is applicable for the projects that you want to replicate an existing study using a new set of data or the same set of data used in the replication package.

To test the effectiveness of different metaheuristic algorithms, this project will use two standard image classification datasets: **MNIST** and **CIFAR-10**. These datasets were chosen because they are widely used benchmarks, which allows a straightforward comparison of our results with existing studies. This approach is also consistent with the methodology used by (Bibaeva, 2018) , which serves as reference for our work.

The datasets are:
- The MNIST dataset (MNIST classification using multinomial logistic + L1, n.d.): This collection consists of 70,000 grayscale images of handwritten digits (0-9).
- The CIFAR-10 dataset (Krizhevsky, 2009): This dataset contains 60,000 32×32 color images across 10 object classes (eg: airplane, cat, truck)

By using both datasets, we can evaluate how each metaheuristic performs on tasks of varying difficulty. Both datasets involve a 10-class classification problem, providing a consistent framework for assessing the methods.

5. Overview

Provide an overview of your technical approach. How do you want to conduct your study? How do you want to solve the problem you are proposing? How do you want to conduct your replication study?

Our image classification problem is formulated with a representation, fitness function, and operators. A state, or a candidate solution in the search space, is represented as an image of any kind. To reduce noise, it is preprocessed under specific procedures before running down a model. The following preprocessing steps are inspired by Kelvin's previous work (Mock, Texture-Image-Comparison, 2024) at his previously taken course – CSI5341 Learning-based Computer Vision at the University of Ottawa:

1. The image is read into a digital array of pixel values with the scikit-image library.
2. To align with images from both datasets, the image resolution is resized to 32×32 pixels.
3. The image is converted to grayscale if it is currently not.
4. The image is normalized to a range of [0, 1] both inclusive, by dividing by 255 from each pixel value in the image's array.
5. Gamma correction, as a point processing step is applied to the image to adjust its brightness, so that the model can recognize objects easier.
6. Adaptive histogram equalization is applied to the image to enhance its contrast.
7. A sobel-edge filter (Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and open-source contributors, 2015) is applied to detect edges of an image.
8. Gaussian blur (Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and open-source contributors, 2015) is applied for smoothing.

A multi-objective fitness function will then be used to formulate the quality of a certain set of hyperparameters on a particular model and a particular dataset. Traditionally, we measure the accuracy, precision, recall, and (macro and micro) F1 scores of a certain array of predictions from a model. Those metrics will be a significant part of the fitness function.

Operators depend on algorithms. Evolutionary algorithms like the genetic algorithm or memetic algorithm uses *selection*, *crossover*, and *mutation*. Simulated annealing uses an exponential cooling schedule to adjust the temperature, which is an important parameter to evaluate a solution with a fitness function. Other choices worth investigating its performance could be those about swarm optimization, namely the particle-based one and ant colony one. Particle Swarm Optimization (PSO) takes a *velocity update* to adjust each particle's velocity using inertia, cognitive, and social coefficients; and position update to move each particle according to its updated velocity (Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad, 2025). Ant Colony Optimization uses mainly Pheromone deposition and Pheromone evaporation.

6. Analysis or implementation Procedure

Clearly but briefly explain what analytical or implementation steps you will take to answer your research questions in Section 3. If you are running a survey, you should still design your survey in such a way that it answers some research questions that you have identified in Section 3.

We expect a tabular analysis of different types of models, datasets (MNIST and CIFAR-10), and different metaheuristics. In addition to the metaheuristics, it is also necessary to plan what models we should try. For simplicity, we can use models from scikit-learn with prebuilt structures. To ensure a robust analysis, there are several types of models: tree-based, probabilistic, linear or polynomial based, kernel-based, neural networks, and ensembles. We choose representative ones among all those types. However, due to limited time, we have to narrow down the scope. Inspired by Kelvin's work about explainability AI (Mock, Drug-

Consumption-Machine-Learning-analysis, 2024), from the course CSI5155 Machine Learning at the University of Ottawa last year, there are 4 main types: tree-based, linear-based, permutation-based, and kernel-based. The simplest tree-based model is a decision tree. Techniques like random forest and gradient boosting are their respective bagging and boosting variations and thus we need not to repeatedly analyze them. K-Nearest-Neighbor forms a basis of machine learning as a kernel-based model. According to our work in assignment 1 of this course, it works well with simulated annealing. Since we do not know how simple or complex a custom neural network should seem representative, a prebuilt multi-layer perceptron (MLP) model from scikit-learn would be desirable. Finally, if time allows, linear regression seems conceptually simpler as an implementation of linear or polynomial based models.

7. Teamwork Plan (applies only to teams)

If you are doing the course project by yourself, this section does not apply to you. If you are a team of two, explain how you will divide the work and who is going to be responsible for each envisaged activity.

	Fernando	**Kelvin**
Phase 1: Project Setup	Extract and clean datasets.	Implement machine learning models and their evaluations.
	Implement base classes for metaheuristics algorithms.	
Phase 2: Experiments	Implement Simulated Annealing.	Implement the baselines by leveraging random search and grid search from scikit-learn.
	Implement Swarm Optimization algorithm(s).	Implement evolutionary-based algorithm(s).

References

Bibaeva, V. (2018). USING METAHEURISTICS FOR HYPER-PARAMETER OPTIMIZATION OF CONVOLUTIONAL NEURAL NETWORKS. *2018 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2018, AALBORG, DENMARK.* Hamburg.

Catalin Stoean, Miodrag Zivkovic, AleksandraNebojsa Bacanin, Roma Strulak-Wójcikiewicz, Milos Antonijevic, and Ruxandra Stoean. (2023, March 4). Metaheuristic-Based Hyperparameter Tuning for Recurrent Deep Learning: Application to the Prediction of Solar Energy Generation. *Advances in Mathematics for Applied Machine Learning*.

Krizhevsky, A. (2009). *The CIFAR-10 dataset*. Retrieved from Alex Krizhevsky: https://www.cs.toronto.edu/~kriz/cifar.html

MNIST classification using multinomial logistic + L1. (n.d.). Retrieved from Scikit-Learn: https://scikit-learn.org/stable/auto_examples/linear_model/plot_sparse_logistic_regression_mnist.html

Mock, K. (2024, December 9). *Drug-Consumption-Machine-Learning-analysis*. Retrieved from GitHub: https://github.com/kmock930/Drug-Consumption-Machine-Learning-analysis

Mock, K. (2024, October 23). *Texture-Image-Comparison*. Retrieved from GitHub: https://github.com/kmock930/Texture-Image-Comparison

Mohammed Q Ibrahim, Nazar K. Hussein, David Guinovart, Mohammed Qaraad. (2025). Optimizing Convolutional Neural Networks: A Comprehensive Review of Hyperparameter Tuning Through Metaheuristic Algorithms. *Computational Methods in Engineering*.

Sajjad Nematzadeh, Farzad Kiani, Mahsa Torkamanian-Afshar, Nizamettin Aydin. (2021). Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases. *Computational Biology and Chemistry*.

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and open-source contributors. (2015). *Edge operators*. Retrieved from scikit-image: https://scikit-image.org/docs/stable/auto_examples/edges/plot_edge_filter.html

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and open-source contributors. (2015). *skimage.filters*. Retrieved from scikit-image: https://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.gaussian