T.R.
SAKARYA UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DYNAMIC HEURISTIC APPROACH TO ENHANCE THE PERFORMANCE OF FEW-SHOT META-LEARNING

PhD THESIS

Ömer MİRHAN

Computer and Information Systems Engineering Department

FEBRUARY 2024

T.R.
SAKARYA UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DYNAMIC HEURISTIC APPROACH TO ENHANCE THE PERFORMANCE OF FEW-SHOT META-LEARNING

PhD THESIS

Ömer MİRHAN

Computer and Information Systems Engineering Department

Thesis Advisor: Prof. Dr. Numan ÇELEBİ

FEBRUARY 2024

The thesis work titled “Dynamic Heuristic Approach to Enhance the Performance of Few-Shot Meta-Learning” prepared by Ömer MİRHAN was accepted by the following jury on …./…. /……. by unanimously/majority of votes as a PhD THESIS in Sakarya University Graduate School of Natural and Applied Sciences, Computer and Information Systems Engineering Department.

Thesis Jury

Head of Jury :	**Prof. Dr. Devrim AKGÜN** Sakarya University
Jury Member :	**Prof. Dr. Numan ÇELEBİ** (Advisor) Sakarya University
Jury Member :	**Dr. Öğr. Üyesi Tuğrul TAŞCI** Sakarya University
Jury Member :	**Dr. Öğr. Üyesi Ertuğrul BAYRAKTAR** Yıldız Technical University
Jury Member :	**Doç. Dr. Zafer ALBAYRAK** Sakarya University of Applied Sciences

STATEMENT OF COMPLIANCE WITH THE ETHICAL PRINCIPLES AND RULES

I declare that the thesis work titled "Dynamic Heuristic Approach To Enhance The Performance Of Few-Shot Meta-Learning", which I have prepared in accordance with Sakarya University Graduate School of Natural and Applied Sciences regulations and Higher Education Institutions Scientific Research and Publication Ethics Directive, belongs to me, is an original work, I have acted in accordance with the regulations and directives mentioned above at all stages of my study, I did not get the innovations and results contained in the thesis from anywhere else, I duly cited the references for the works I used in my thesis, I did not submit this thesis to another scientific committee for academic purposes and to obtain a title, in accordance with the articles 9/2 and 22/2 of the Sakarya University Graduate Education and Training Regulation published in the Official Gazette dated 20.04.2016, a report was received in accordance with the criteria determined by the graduate school using the plagiarism software program to which Sakarya University is a subscriber, I have received an ethics committee approval document, I accept all kinds of legal responsibility that may arise in case of a situation contrary to this statement.

(……/……/20…..)

Ömer MİRHAN

Dedicated to my kids, please never stop learning

ACKNOWLEDGMENTS

Firstly, I would thank Allah for giving me the health and the capacity to successfully complete this long journey, Alhamdulillah.

I express my deepest gratitude to my supervisor, Prof. Dr. Numan ÇELEBI. Your unlimited support and guidance have been truly essential during this research. Without your expertise, wisdom, and patience, this research would not have been possible.

I would also like to thank the monitoring committee, Dr. Ertuğrul BAYRAKTAR and Dr. Tuğrul TAŞCI, for their valuable feedback, which helped shaping this work.

I am deeply thankful to my parents, Hala and Hisham, for their endless love and support. The trust you place in me has been my constant motivation in all my life.

To my beloved siblings, Huda, Nada, Anas and Asmaa. Thank you for always being there, cheering me on, this was always a source of strength.

A special appreciation and gratitude goes to my wife, Zeynep, and our kids, Shahd Aynur, Hisham Ahmet and Leyla Nur. Your patience, understanding, and unlimited support have been the cornerstone of my pursuit of knowledge. Your love has sustained me through late nights and tough days, encouraging me to overcome frustrations and difficulties throughout completing this hard journey.

Finally, I would like to extend my wishes for the quick recovery and stability of my home country Syria. Additionally, I express my deep appreciation to Türkiye including everyone helped me establishing a new life here to resume my academic and professional endeavors.

Thank you to everyone who played a role in my academic and professional life.

Ömer MİRHAN

TABLE OF CONTENTS

Sayfa

ABBREVIATIONS

ADAM : Adaptive Moment Estimation
AI : Artificial Intelligence
ANN : Artificial Neural Network
BP : Backpropagation
CNN : Convolutional Neural Network
DL : Deep Learning
DPO : Dynamic Population Optimization
GA : Genetic Algorithm
GD : Gradient Descent
MAML : Model Agnostic Meta Learning
ML : Machine Learning
MSE : Mean Square Error
PSO : Particle Swarm Optimization
SA : Simulated Annealing
SGD : Stochastic Gradient Descent
SI : Swarm Intelligence

LIST OF TABLES

Page

LIST OF FIGURES

Page

DYNAMİC HEURİSTİC APPROACH TO ENHANCE THE PERFORMANCE OF FEW-SHOT META-LEARNİNG

SUMMARY

Meta-learning has recently become an interesting topic for researchers, particularly in supervised learning problems with a lack of training data. Meta-learning has shown effectiveness in generalization and adapting to solve new tasks with only a few data points. To train a deep learning model in general or meta-learning specifically, an optimization function should be used to update parameters during each training cycle according to the calculated loss. The popular meta-learning models have used one of the traditional gradient-based optimizers. However, challenges introduced by meta-learning, such as performance considerations during the meta-training and the need for faster adaptation, might not be handled efficiently by those optimizers.

In this research, we propose a custom optimizer to train meta-learning models. Our proposal is a new optimizer based on combining a metaheuristic algorithm with traditional gradient-based techniques. The heuristic algorithm starts searching for the optimized values for the model's weights using some random initial candidate solutions. Then iteratively, according to the performance of each individual, dynamic population strategy will be applied to population members by either reproducing or eliminating members from the population. The learning process will then continue using classic gradient optimization starting with the optimal solution found via the heuristic algorithm.

The custom optimizer we have developed was first tested and tuned on five classification benchmark datasets and showed higher accuracy and faster convergence. Then the same approach -with slight enhancements- was applied in order to solve the meta-learning problem. Our experimental analysis shows that our optimizer could enhance the performance of training meta-learning models and enable the efficient finding of optimal parameters due to the dynamic characteristics of our proposed strategy.

AZ ÖRNEKLE META-ÖĞRENME'NİN PERFORMANSINI ARTIRMAK İÇİN DİNAMİK HEURİSTİK BIR YAKLAŞIM

ÖZET

Meta-öğrenme, öğrenmeyi öğrenme olarak da bilinen, derin öğrenme altında göreceli olarak yeni bir alt alan, özellikle sınırlı eğitim verileri ile karşılaşılan denetimli öğrenme senaryolarının zorluklarına çözüm olarak araştırmacılar arasında önemli bir ilgi kazanmıştır. Meta-öğrenmenin merkezi odak noktası, bir modelin genelleme kapasitesini artırmak ve görülmemiş görevlere uyum sağlamak üzerinedir. Bu, derin öğrenme modelinin, yinelemeli optimizasyon döngüleri geçirerek ve modele ait parametreleri buna göre güncelleyerek sezgisel bir eğitim platformu aracılığıyla elde edilir. Meta-öğrenme alanında, eğitim süreci, her meta-eğitim döngüsü sırasında elde edilen hesaplanmış kayıplara dayanarak modelin parametrelerini ince ayarlamayı içerir. Klasik yaklaşım, klasik gradyan tabanlı optimize edicileri kullanırken, bu yöntemleri meta-öğrenme problemleri tarafından tanıtılan ayırt edici zorluklara uyarlamak bazı verimsizliklere neden olur.

Bir anahtar zorluk, meta-eğitim sırasında, modelin sınırlı veri noktalarıyla çeşitli görevlerden öğrenme gerekliliğidir. Meta-öğrenmenin karmaşıklığı, genellikle mevcut veriden bilgi çıkarma ve görevler arasında öğrenme için yeni stratejileri keşfetme arasında doğru bir denge gerektirir. Bu denge, özellikle meta-öğrenme senaryolarının sınırlı veri karakteristiği ile karşılaşıldığında, mevcut veriden bilgi çıkarmak ve görevler arasında öğrenme için yeni stratejileri keşfetmek arasında doğru bir denge gerektirir. Özellikle kritik hale gelir. Ayrıca, görülmemiş görevlere hızlı uyum sağlama ihtiyacı, optimizasyon sürecini daha da karmaşık hale getirir. Klasik gradyan tabanlı optimize ediciler, özellikle sınırlı görev özgü veri ile karşılaşıldığında, model parametrelerini hızlı bir şekilde yeni bir görevin benzersiz özelliklerine uyum sağlamakta zorlanabilir. Bu, modelin hızlı bir şekilde adapte olması ve orijinal eğitim setinin bir parçası olmayan görevlerde iyi performans göstermesi beklenen uygulamalarda kritik bir düşünce noktasıdır.

Meta-öğrenme alanındaki araştırmacılar, bu zorlukları ele almak için alternatif optimizasyon stratejilerini aktif bir şekilde keşfetmektedirler. Model güncellemeleri, düzenleme ve optimizasyon fonksiyonlarına yenilikçi yaklaşımlar önererek, meta-öğrenme modellerinin verimliliğini ve etkinliğini artırmayı hedeflemektedirler. Bu ilerlemeler, sınırlı etiketli veri örnekleri ile elde edilebileceklerin sınırlarını zorlamak için kritiktir, bu da meta-öğrenmeyi klasik denetimli öğrenmenin gerçek dünya uygulamalarındaki kısıtlamaları aşma konusunda umut vadeden bir yol haline getirebilir.

Sezgisel algoritmalar, kesin bir çözüm elde etmek zor veya hesaplama maliyeti yüksek olduğunda karmaşık problemler için yaklaşık çözümler bulmak için kullanılan problem çözme yaklaşımlarıdır. Sezgisel algoritmalar, büyük çözüm alanlarında çözümleri bulmak için kuralcıklar, deneyim veya sezgiyi kullanır. Yapay Sinir Ağları'nı (YSA) eğitirken, sezgisel algoritmalar, gradyan tabanlı tekniklere alternatif

olarak değerli birer seçenek olarak hizmet eder. Her parametre ayarlaması için gradyanları hesaplamaya dayanan klasik yöntemlerin aksine, genetik algoritmalar, sürü zekâsı veya simüle edilmiş tavlama gibi sezgisel algoritmalar, çözüm alanlarını daha geniş bir şekilde keşfeder. Bu keşif, YSA'lerin yerel minimumlardan kaçınmasına yardımcı olur ve çeşitli ve potansiyel olarak üstün model konfigürasyonlarını keşfetme yeteneklerini artırır. Sezgisel algoritmalar, özellikle gradyan tabanlı yöntemlerin mücadele ettiği karmaşık optimizasyon alanlarında YSA'leri eğitmek için çok yönlü ve hesaplama açısından verimli bir yol sunar.

Bu çalışmada, meta-öğrenme modellerinin eğitimini artırmak amacıyla tasarlanmış yeni bir özel optimize ediciyi tanıtıyoruz. Optimize edicimiz, bir dinamik popülasyon tabanlı sezgisel algoritmayı klasik gradyan tabanlı tekniklerle sorunsuz bir şekilde entegre ederek, meta-öğrenme problemlerinin ortaya çıkardığı zorluklara dinamik ve verimli bir yaklaşım sunar. Metaheuristik algoritma, optimal model parametre değerlerini aramaya başlamak için arama alanında rastgele dağıtılan başlangıç aday çözümler üreterek başlar. Ardından, dinamik bir popülasyon stratejisi uygulanır ve bu, bireysel performansa dayalı olarak popülasyon üyelerinin üretilmesini veya elenmesini içerir.

Dinamik popülasyon stratejisi, her eğitim döngüsünün sonunda popülasyon üyelerinin performansını değerlendirmeyi içerir. İyi performans gösterenler çoğaltılır ve umut veren alanları keşfetmelerini simgelerken, en az performans gösteren bireyler elenir. Bu sürekli çoğaltma ve elenme süreci, umut veren bölgelerde aramayı yoğunlaştırmayı ve daha az verimli alanlarda daha fazla keşif yapmamayı amaçlar. Heuristik keşif aşamasının ardından, öğrenme süreci, metaheuristik algoritma tarafından belirlenen en iyi çözümü kullanarak klasik gradyan optimizasyonuna sorunsuz bir şekilde geçer. İki aşama arasındaki eğitim döngülerinin dağılımı, kritik bir giriş parametresi olan Heuristik Oranı tarafından kontrol edilir, bu da heuristik keşif aşamasına ayrılan döngülerin yüzdesini belirtir.

Özel optimize edicimizin performansını değerlendirmek amacıyla geniş kapsamlı test ve ayarlamalar gerçekleştirdik ve çeşitli ölçüm veri setlerinde bunları gerçekleştirdik. Bu veri setleri, Iris, MNIST, CIFAR-10, CIFAR-100 ve Fashion gibi klasik sınıflandırma görevlerini içermekte olup, görüntü ve desen tanıma alanında çeşitli karmaşıklıkları temsil etmektedir.

Klasik veri setlerinde elde edilen umut verici sonuçlar tarafından teşvik edilen, yaklaşımımızı meta-öğrenmenin zorluklarına çözüm üretecek şekilde genişlettik. Özel optimize ediciyi, Omniglot ve MiniImageNet gibi iki iyi kurulmuş meta-öğrenme veri setine uyguladık. Bu veri setleri, her sınıfta sınırlı örnek içermeleriyle karakterize edilir ve bu da onları bir modelin minimal veri ile yeni görevlere adapte olma yeteneğini değerlendirmek için özellikle uygun kılar.

Deneylerimiz, özel optimize edicinin üç önemli meta-öğrenme çerçevesine uygulanmasını içerdi: MAML, Reptile ve Meta-SGD. Bu çerçeveler, bir modelin meta-öğrenme senaryolarındaki genelleme ve uyum yeteneklerini değerlendirmek için yaygın bir şekilde kullanılmaktadır.

Hem MAML hem de Meta-SGD'nin elde edilen doğruluğu, Omniglot ve MiniImageNet veri kümelerinde %2-2,5 oranında iyileşme gördü. Ancak Reptile, hem veri kümelerinde hem de aynı sayıda dönemde %1'lik bir doğruluk artışı gösterdi. Hem MAML hem de Meta-SGD, ilgili meta eğitim aşamaları sırasında iki adımlı bir optimizasyon sürecini içerir. MAML ve Meta-SGD tarafından kullanılan iki aşamalı optimizasyon sürecinin aksine Reptile, daha hızlı yakınsamayı vurgulayan daha basit

bir yaklaşımı benimser. Eğitim sırasında Reptile, bir iç döngü içindeki belirli bir görev üzerinde yalnızca birkaç kademeli adım gerçekleştirir.

Genel olarak, deneylerimizden elde ettiğimiz sonuçlar, özel optimize edicimizin sadece meta-öğrenme modellerinin eğitim performansını artırmakla kalmadığını, aynı zamanda verimli parametre keşfini kolaylaştırdığını gösterdi. Önerdiğimiz stratejinin dinamik özellikleri, daha yüksek doğruluk, daha hızlı yakınsama ve görünmeyen görevlere hızlı uyum sağlama yeteneğine katkıda bulunarak, yenilikçi yaklaşımımızın meta-öğrenme optimizasyon alanındaki potansiyelini sergilemektedir.

1. INTRODUCTION

This thesis investigates how using heuristic algorithms can make few-shot meta learning work better. Through this study, we aim to make meaningful contributions to the field of meta learning optimization process.

1.1. Overview

Artificial Intelligence (AI) is a broad field dedicated to creating intelligent agents capable of replicating human-like cognitive functions. Under the umbrella of AI, Machine Learning (ML) focuses on developing algorithms that enable systems to learn from data, categorized into supervised, unsupervised, and reinforcement learning. Neural Networks (NNs), or Artificial Neural Networks (ANNs), the backbone of machine learning, simulating the complexity of the human brain. Built from layers, neurons, weights, and biases, these elements collaborate to process information, recognize patterns, and make predictions [1]. The network is organized into layers, including the input layer, hidden layers, and output layer. Neurons, the basic computational units, exist in each layer, with those in the input layer representing raw data features. This information will then processed by hidden layers this through connections, while the output layer generates the last prediction or classification [2].

Deep Learning (DL), a subset of machine learning, Figure 1.1, employs deep neural networks with multiple hidden layers. This depth allows automatic learning of hierarchical features from data, resulting in more sophisticated representations. The advancement of deep learning is evident in many areas such as image recognition (utilizing Convolutional Neural Networks), Natural Language Processing (leveraging Recurrent Neural Networks and Transformers), healthcare (for medical image analysis and diagnosis), and autonomous vehicles (interpreting environments for safe navigation) [3]. One fundamental aspect of training neural networks is optimization, often achieved through Gradient Descent. This iterative algorithm minimizes the error or loss function by adjusting the weight vector that reduces the error [4]. Stochastic Gradient Descent (SGD) is a usually utilized variant, randomly selecting subsets of training data for efficiency.

The learning rate is a critical hyperparameter in Gradient Descent, influencing the size of steps taken during optimization. Selecting an appropriate learning rate is crucial for convergence and stability during training. While standard gradient-based optimizers like SGD are common, modern optimizers like Adam, RMSprop, and Adagrad are developed to address specific challenges in training deep neural networks.

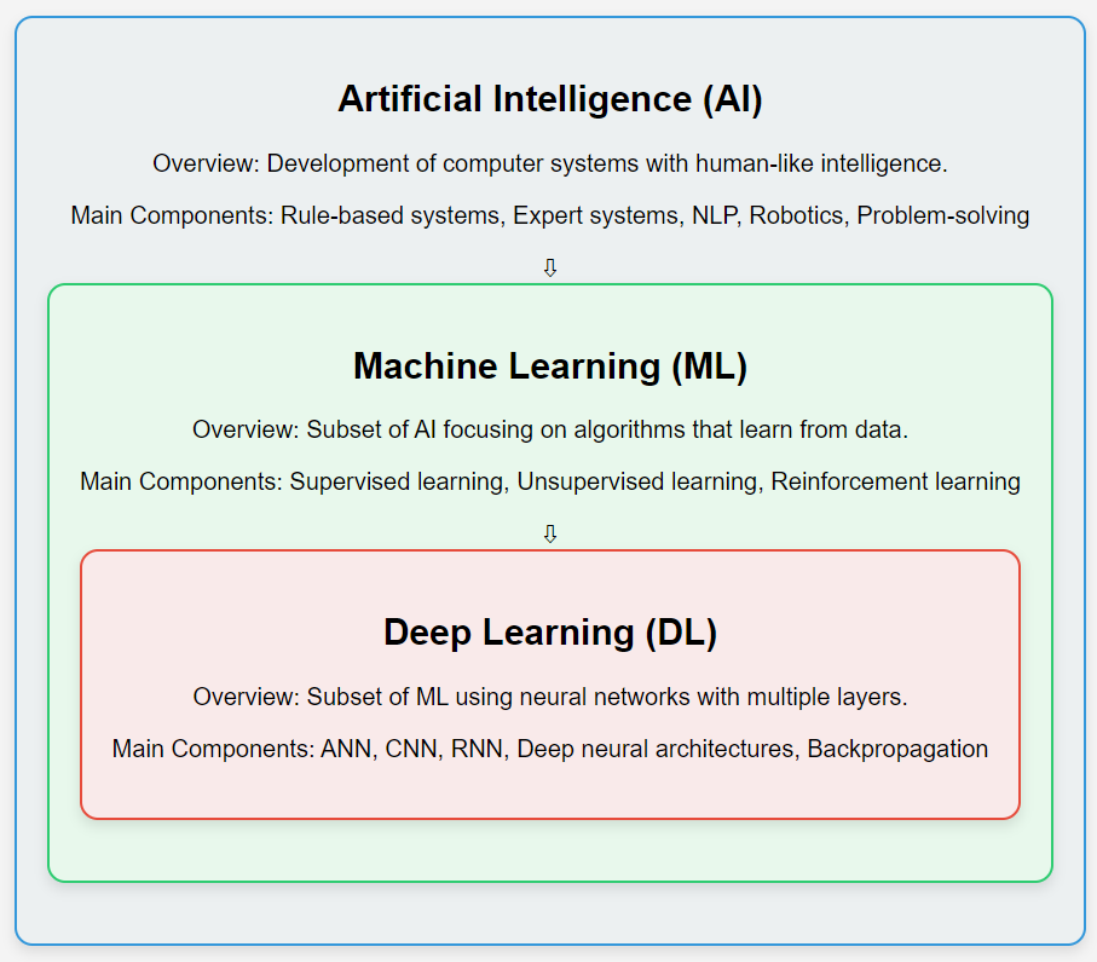


Figure 1.1. Deep learning family.

Innovations in optimization also explore the combination of traditional gradient-based techniques with metaheuristic algorithms. Metaheuristics, such as genetic algorithms or simulated annealing, introduce randomness and global search strategies, enhancing the exploration of the parameter space [5].

Traditional machine learning techniques usually solve problems by undergoing intensive training cycles to learn from scratch. However, this often requires a massive amount of training data, which might not always be available or may come at a high cost. Human brains have the ability to use accumulated knowledge to quickly learn

new things. For example, let's assume that a child has never seen a zebra before. By showing the child a few pictures of a zebra, they will be able to recognize zebras easily. In this case, we cannot assume that the child learned from scratch and performed the recognition process by looking at just a few examples. Instead, they used their previously accumulated knowledge and their ability to recognize animals they were familiar with, such as dogs or cats, to quickly identify a new kind of animal by only seeing few samples. This is exactly how meta-learning works: we first train the model on various tasks to create a generic model. Then, to solve new tasks, only a few data points are needed to fine-tune the model's parameters [6].

Meta-learning, also known as learning to learn, is a powerful framework within the field of deep learning that focuses on enabling models to learn how to learn from limited data [6]. Unlike traditional machine learning approaches that rely on large datasets for training, meta-learning aims to create models that can adapt and generalize to unseen tasks with the need for only few data points. At its core, meta-learning involves training a model on a variety of tasks in such a way that it becomes adept at quickly adapting to new, unseen tasks [7]. This is achieved by exposing the model to a diverse set of tasks during its training phase. Each task consists of a small dataset, and the model learns not only the specifics of each task but also a more general set of parameters or initial conditions that facilitate rapid adaptation [8].

Heuristic algorithms are smart problem-solving methods that focus on being practical and efficient rather than aiming for the best possible solution. Unlike exact algorithms that guarantee the best solution, heuristics aim to find a good enough solution within a reasonable amount of time, making them particularly useful for solving complicated problems where finding an best solution is computationally infeasible [9]. Swarm Intelligence, including algorithms like Particle Swarm Optimization PSO [10], mimics the collective behavior of groups of entities to find solutions. In Particle Swarm Optimization, a population of particles iteratively adjusts its position in a search space based on its own experience and that of its neighbors [10]. In population-based heuristic algorithms, the population size remains constant throughout the optimization process. Each member of the population, often represented as a solution candidate or individual, is granted equal importance and opportunity to influence the search for the optimal solution, regardless of an individual's past performance or fitness score, it continues to participate in the search process [11]. In simpler terms, objects that excel

and are near a good solution will receive equal opportunities as those exploring areas distant from a good solution. In contrast, the heuristic algorithm we have developed is based on a dynamic members where the performance of each individual will be evaluated at the end of each training cycle, then new members might join the next cycles or individuals with unsatisfactory performance will be dismissed from the population.

1.2. Problem Statement

Although the advancements in meta-learning and its proven efficacy in few-shot scenarios, a critical bottleneck remains—the optimization process. Standard gradient-based optimizers, which have shown success in conventional machine learning settings, may underperform in addressing the unique challenges raised by meta-learning.

The core problem lies in the necessity for meta-learning models to rapidly adapt to new tasks with limited data. Traditional optimizers, designed for smooth and continuous optimization landscapes, may struggle in the dynamic environment characteristic of meta-learning. Moreover, the importance of task-specific adaptation often demands a more focusing approach to parameter updates.

This research identifies the limitations of existing optimizers in the meta-learning context and aims to address them through the development of a heuristic based optimizer.

1.3. Objectives of the Research

This research aims to address the identified challenges within the platform of meta-learning optimization. The overall goal is to develop a custom optimizer that optimally balances the exploration of novel solutions and the exploitation of known information, specifically tailored to the few-shot nature of meta-learning. The objectives can be outlined as follows:

1. Custom Optimizer Development: Design and implement a novel custom optimizer that integrates metaheuristic algorithms with traditional gradient-based optimization techniques. Also, empirically ensure the optimizer's adaptability to the nature of meta-learning tasks, focusing on efficient parameter updates and fast convergence.

2. Efficiency Enhancement for Benchmark Datasets: Firstly, the performance of the developed custom optimizer on standard classification benchmark datasets will be evaluated by comparing the accuracy and convergence speed of the custom optimizer against traditional gradient-based optimizers, demonstrating its efficacy in conventional machine learning scenarios.
3. Meta-Learning Model Optimization: Apply the custom optimizer to few-shot meta-learning scenarios, where adaptation to new tasks with limited data is crucial. Then assess the performance of meta-learning models trained with the custom optimizer in terms of accuracy, convergence speed.
4. Empirical Validation and Analysis: Perform a comprehensive empirical examination of the experimental outcomes, addressing both the strengths and limitations of our approach. Additionally, compare our approach with four state-of-the-art meta-learning models through statistical analysis and visualizations, ensuring its effectiveness in enhancing meta-learning performance.

1.4. Thesis Organization

We have organized this thesis as follows:

- **Chapter 2:** Background and Literature Review: This chapter provides a comprehensive overview of the foundational aspects relevant to the study. It begins by delving into the fundamentals of deep learning, establishing a theoretical background for subsequent discussions. Following this, the chapter shifts its focus to meta-learning, offering an in-depth exploration of its theoretical foundations, applications, and the challenges it faces. The discussion then extends to heuristic algorithms, where a detailed examination of principles and methodologies behind heuristic approaches is presented.
- **Chapter 3:** Proposed Model: This pivotal chapter introduces the novel hybrid optimizer designed for meta-learning. The chapter outlines the conceptual framework, detailing how the metaheuristic algorithm is combined with traditional gradient-based techniques. It also elaborates on the optimization process and the rationale behind the design choices made in building the proposed model.

- **Chapter 4:** Empirical Analysis: Present the empirical analysis of the custom optimizer. Initial testing on classification benchmark datasets is discussed, highlighting the achieved higher accuracy and faster convergence. The application of the proposed model to solve the meta-learning problem is thoroughly examined through experimental analysis.
- **Chapter 5:** Conclusion and Outlook: The final chapter provides a comprehensive conclusion of the research findings, summarizes key contributions, and discusses implications. Additionally, the chapter outlines potential avenues for future research, highlighting areas where further exploration and refinement of the proposed approach could be undertaken.

2. BACKGROUND AND LITERATURE REVIEW

This chapter navigates the foundational landscapes of deep learning, meta-learning, and heuristic algorithms, providing the essential background to contextualize our novel approach to enhancing meta-learning performance. The evolution of these domains sets the stage for understanding the difficulties and challenges that our proposed approach aims to address.

2.1. Deep Learning: A Brief Summary

The field of deep learning, illustrated in Figure 2.1 [1], characterized by its neural network architectures and sophisticated training methodologies, forms the cornerstone of modern machine learning applications. This section offers a brief synthesis, emphasizing advanced concepts that directly influence the development and performance of deep-learning models. By identifying key components and recent trends and advances, we establish a baseline understanding for the subsequent exploration of deep learning.

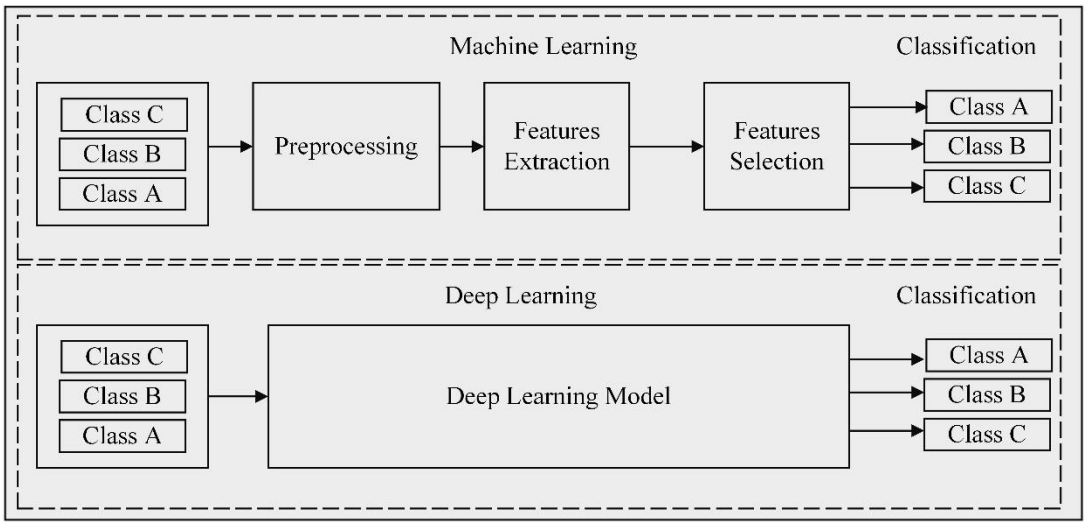


Figure 2.1. Traditional machine learning vs. deep learning.

2.1.1. Categories of deep learning approaches

Deep learning methodologies can be classified into three primary categories: supervised learning, unsupervised learning, and reinforcement learning, Figure 2.2.

Supervised Learning: In this category of deep learning, the model undergoes training using a labeled dataset, in which every input is matched with a corresponding output or target. The objective is to grasp a relationship between inputs and outputs, enabling the algorithm to predict outcomes for fresh data that the model did not see during the training process [12], [13]. Classification and regression problems are common in supervised learning. For example, checking whether an message is spam or not (classification) or forecasting of the price of a house based on its features (regression).

The algorithm undergoes training using a labeled dataset, during that the model fine-tunes weights to minimize the distance between its predictions and the actual data [14].

Unsupervised Learning: Handling unlabeled data, unsupervised learning involves the algorithm in the exploration of patterns, relationships, or structures within the data without explicit instructions on what to do with data [15]. Unsupervised learning involves two key tasks: clustering and dimensionality reduction. Clustering is about grouping similar data points together, while dimensionality reduction aims to preserve important data while decreasing the total number of attributes. These techniques play vital roles in data analysis and pattern recognition. Clustering helps in identifying natural groupings within data, while dimensionality reduction simplifies complex datasets for easier analysis. Both are essential tools in uncovering insights and patterns hidden within large datasets [12]. The algorithm explores the input data to identify inherent structures or patterns, with no exact target labels to guide the learning [16].

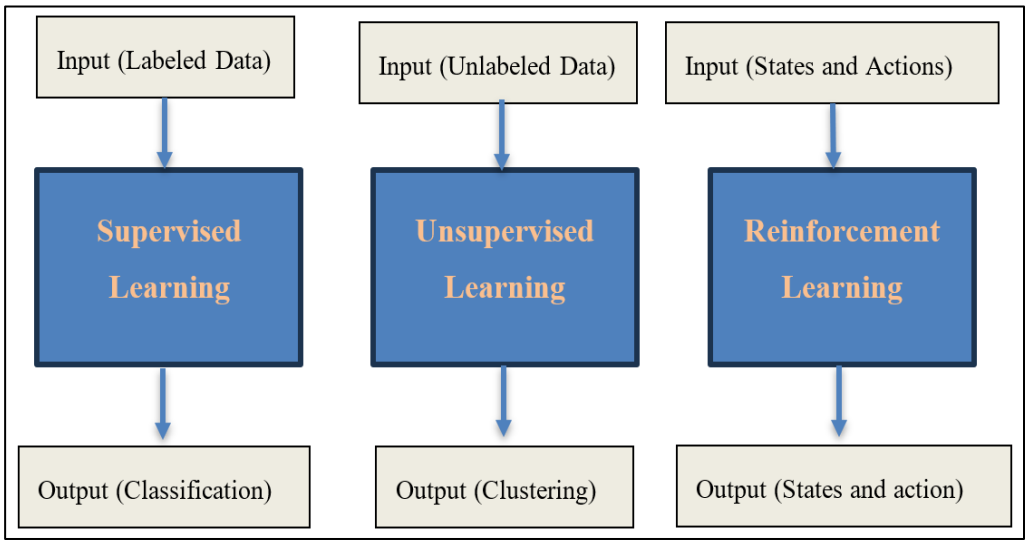


Figure 2.2. Classification of deep learning models.

Reinforcement Learning: Reinforcement learning involves an agent that will develop its decision-making skills through iterative interactions with its environment [1]. The agent's learning process involves obtained feedback, either a kind of rewards for favorable actions or penalties for unfavorable ones. Its primary aim is to develop a policy that increase the total reward accumulated over time as much as possible. This process is fundamental in reinforcement learning, where the agent learns through trial and error to achieve its objectives efficiently. [17]. Playing games, robotics control, and autonomous systems are common applications of reinforcement learning [12]. The agent learns by trial and error, adjusting its actions to reach the maximum possible awards. The agent learns through exploration and exploitation, receiving feedback from the environment in the form of rewards or penalties. The learning process involves finding a balance between exploring new actions and exploiting known actions that lead to positive outcomes [18].

2.1.2. Neural networks structure

Neural networks can be considered as the cornerstone in the field of artificial intelligence (AI) and machine learning. These computational models draw inspiration from the functioning of biological neural networks within the human brain [2].

A typical neural network consists of layers of interconnected nodes, also known as artificial neurons. These layers usually include the input layer, hidden layer(s), and output layer, forming the three primary types of layers in a neural network. [19].

The input layer's role is to receive the initial data into the neural network. Each node within the input layer is connected to an attribute or input variable. The quantity of nodes in the input layer is determined by the number of dimensions of the input data. Between the input and output layers, the network might include also one or multiple hidden layers. The nodes in the middle layers operate the input data by utilizing weighted connections and activation functions. The term "hidden" comes from the fact that the outputs of these nodes are not explicitly seen in the training samples. The output layer generates the final outcomes of the neural network's computation. The quantity of nodes within the output layer varies depending on the nature of the problem at hand [20].

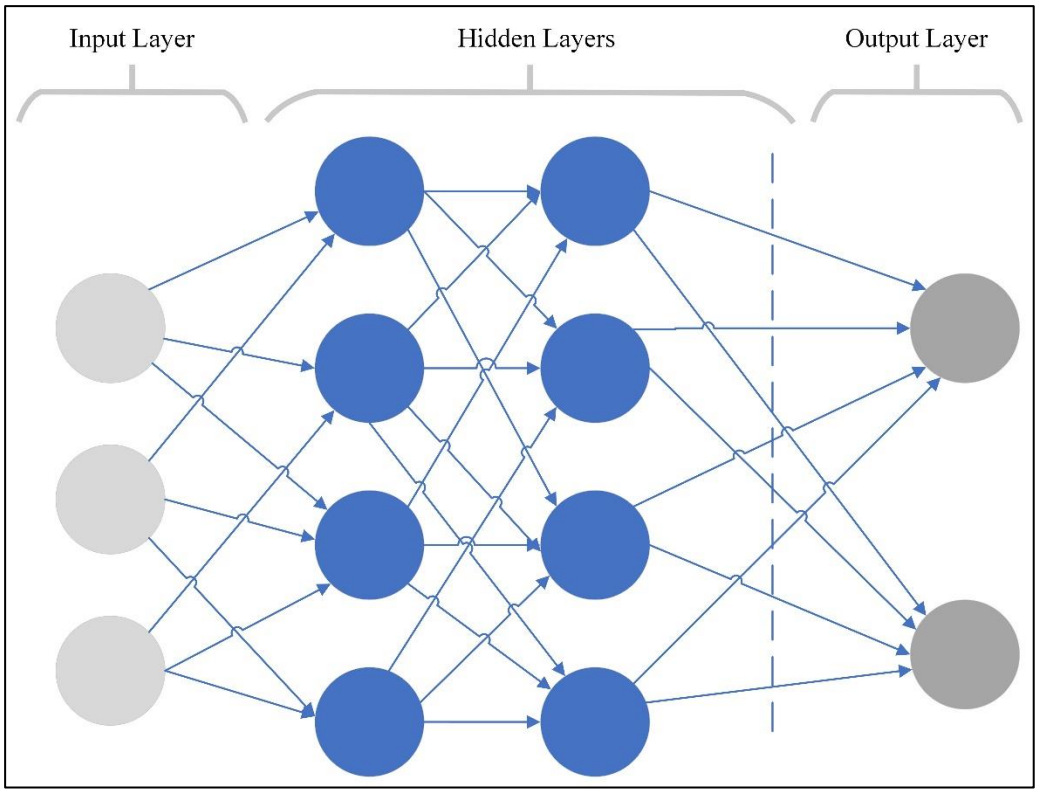


Figure 2.3. Feedforward neural network structure.

The structure of a neural network can vary based on the specific task and architecture chosen. However, here are the common neural network architectures:

- Feedforward Neural Networks (FNN): Information flows in one direction, from input to output, with no feedback loops, Figure 2.3 is a sample structure [20].
- Recurrent Neural Networks (RNN): Accommodates sequential data by incorporating feedback loops, allowing information persistence [20].
- Convolutional Neural Networks (CNN): Specialized for image processing, utilizing convolutional layers for feature extraction [13, 21].

2.1.3. Connections and weights

Every connection between nodes in adjacent layers is accompanied by a weight, signifying the connection's strength. These weights are fine-tuned during the training phase to enhance the network's performance [20]. The weighted total of input data, coupled with a bias, undergoes evaluation through an activation function to ascertain the output of each node. Normally, each node within a layer is associated with a bias term—a constant that's incorporated into the weighted sum before applying the

activation function. The inclusion of bias enables the network to adapt and modulate its output [2].

For a multi-layer neural network with L layers (which includes the input and output layers) and N_l neurons in each layer l, the output equation for a neuron in layer l can be expressed as follows [20]:

$$y_i^{(l)} = f\left(\sum_{j=1}^{N^{(l-1)}} W_{i,j}^{(l)} \cdot y_j^{(l-1)} + b_i^{(l)}\right) \tag{2.1}$$

Where $y_i^{(l)}$ is the result of the $i-th$ node in layer l, f is the applied activation function to the weighted sum. $N^{(l-1)}$ is the number of neurons in the previous layer ($l\ -1$).

$W_{i,j}^{(l)}$ is the value associated with the connection between the $j-th$ node in layer l -1 and $i-th$ node in layer l. $b_i^{(l)}$ is the bias value for the $i-th$ node in layer l.

2.1.4. Activation function

An activation function injects non-linearity into the network, empowering it to grasp complex patterns. Typical activation functions comprise::

- Sigmoid: S-shaped function, often used in the output layer for binary classification, illustrated in Figure 2.4.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

- Hyperbolic Tangent (tanh): Similar to the sigmoid but ranging from -1 to 1.

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.3}$$

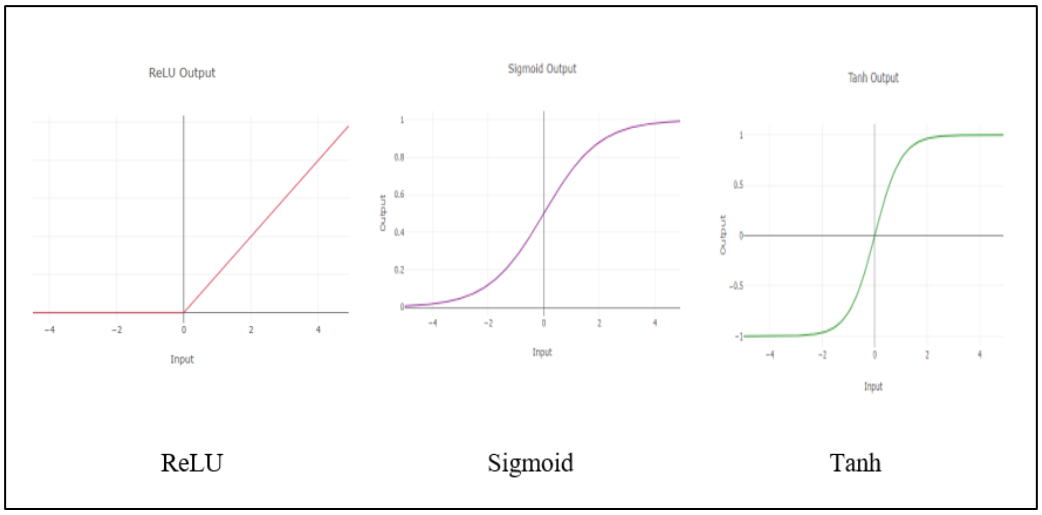


Figure 2.4. Common activation functions.

– Rectified Linear Unit (ReLU): Commonly used in hidden layers, allowing only positive values. Figure 2.4.

$$\text{ReLU}(x) = \max(0, x) \tag{2.4}$$

2.1.5. Loss function

The loss function measures the discrepancy between the predicted output and the actual one. Throughout training, the objective is to diminish this loss as much as possible [2].

– Mean Squared Error (MSE): Usually utilized for regression problems, quantifying the disparity between generated and actual values. The MSE is calculated using the this formual [23]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

Where n is the number of sample data, y_i is the real (observed) value for the i-th sample point, y_i is the calculated value for the i-th sample point.

– Cross-Entropy Loss: Prevalent in classification tasks, calculating the divergence between calculated and true probability distributions [24].

$$H(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij}\log(\widehat{y_{ij}}) \tag{2.6}$$

Where K is the total of output classes in multi-class classification.

2.1.6. Network training

Training involves adjusting the values of nodes according to the difference between the calculated output and the actual output. Backpropagation is the algorithm used to propagate this error backward through the network, updating the weights and biases [2].

2.1.7. Optimization function

Optimizing an Artificial Neural Network involves adjusting the model's parameters to minimize a certain cost or loss function. The optimization function holds significant importance in training and fine-tuning the model to achieve optimal performance. It serves as a guide for turning the model's weights in the course of the learning process, helping the network converge to a solution that minimizes the distance between calculated and desired outputs. Here are some details about optimizing ANNs, including popular optimization algorithms and alternative approaches [25]:

- **Gradient Descent (GD):** The basic optimization algorithm that adjusts model parameters in the direction opposite to the gradient of the cost function. The concept of gradient descent dates back to the early days of optimization and is a fundamental building block for many optimization algorithms [26].

$$\theta_{i+1} = \theta_i - \alpha \nabla J(\theta_i) \tag{2.7}$$

Where θ_{i+1} is the updated parameter values. θ_i is the current parameter values. α is the learning rate, determining the size of the step taken in each iteration. $\alpha \nabla J(\theta_i)$ is the derivation of the loss function J with respect to the parameters θ_i.

- **Stochastic Gradient Descent (SGD):** Optimizes parameters using the gradient computed from a single training example, leading to faster updates. SGD is commonly employed in training large-scale neural networks and has been a key optimization method in machine learning [27].

$$\theta_{j+1} = \theta_j - \alpha \nabla J_i(\theta_j) \tag{2.8}$$

Where the derivation of the cost function is computed for one training sample i. In practice, SGD is often used with mini-batches, which are small groups of training examples. This can further improve the efficiency of the algorithm.

- **Mini-Batch Gradient Descent:** An intermediate approach between Gradient Descent (GD) and Stochastic Gradient Descent (SGD) entails updating parameters based on a small batch of training examples. Mini-batch gradient descent is a standard optimization approach in deep learning due to its efficiency in utilizing both parallelism and stochasticity [27, 3].

$$\theta_{i+1} = \theta_i - \alpha \cdot \frac{1}{m} \sum_{j=1}^{m} \nabla_\theta J\left(\theta; x^{(i_j)}, y^{(i_j)}\right) \tag{2.9}$$

Where m is the batch size and $\nabla_\theta J\left(\theta; x^{(i_j)}, y^{(i_j)}\right)$ is the gradient of the cost function computed on the mini-batch $(x^{(i_j)}, y^{(i_j)})$.

- **Momentum:** Momentum was introduced to address the issue of high variance in Stochastic Gradient Descent (SGD) and to smooth the convergence process. It speeds up convergence in the pertinent direction while dampening fluctuations in irrelevant directions [28].
- **RMSprop**: An adaptive learning rate method that normalizes the gradient using a moving average of squared gradients. Introduced by Hinton in lecture notes, RMSprop is designed to address the sensitivity of learning rates in different dimensions [29].
- **Adam (Adaptive Moment Estimation)**: An adaptive learning rate optimization algorithm that integrates concepts from both momentum and RMSprop. Introduced by Kingma and Ba in 2014, Adam has become a popular optimization choice due to its robustness and efficiency across various tasks [30].
- **Adagrad**: Modifies the learning rates of each parameter independently by considering historical gradient details. Proposed by Duchi et al. in 2011, Adagrad is suitable for sparse data and has been influential in optimization research [31].

– **Nadam**: An extension of Adam that incorporates Nesterov momentum. Nadam, introduced by Dozat in 2016, combines the strengths of Adam and Nesterov accelerated gradient, offering improved convergence properties [32].

The Adam optimizer showcases exceptional adaptability across a wide range of tasks, owing to its dynamic learning rates and momentum. This flexibility enables efficient navigation through various loss function landscapes. A standout feature of Adam is its effective management of sparse gradients, particularly beneficial in situations with high-dimensional and sparse data. The integration of momentum and adaptive learning rates in Adam plays a pivotal role in achieving rapid convergence, enabling neural networks to swiftly attain satisfactory solutions and outperform traditional optimization methods. Numerous studies, consistently demonstrate Adam's superiority over other adaptive learning rate mechanisms.

2.1.8. Challenges of training algorithms

Several issues arise during the training of neural networks. Here, we will outline the key challenges:

Local Minima: Gradient-based optimization methods may get stuck in local minima, demonstrated in Figure 2.5. preventing them from finding the global minimum of the loss function, resulting in suboptimal solutions and slower convergence in complex, non-convex landscapes.

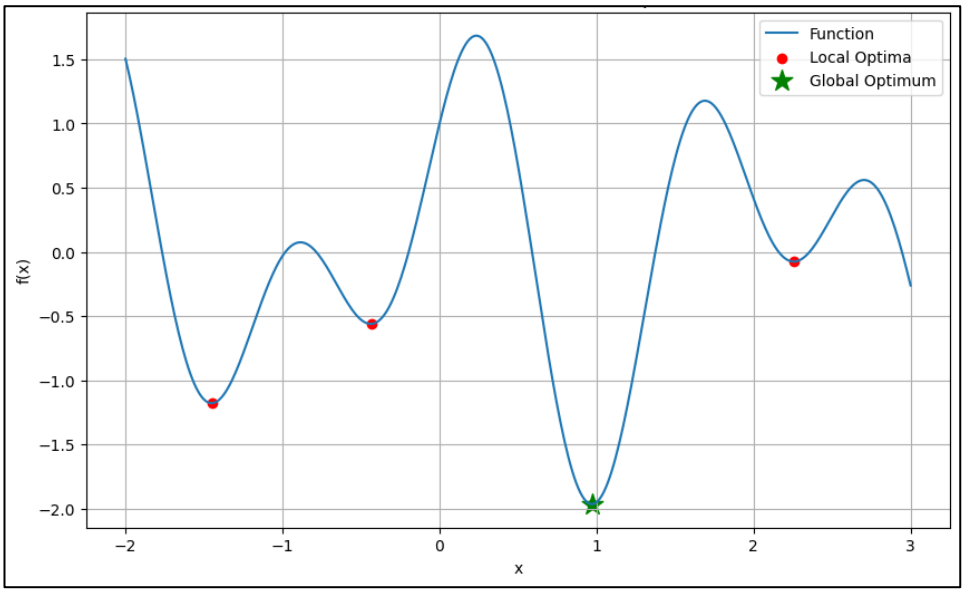


Figure 2.5. Global vs. local minima.

As descripted in [33], local minima is mathematically outlined as the following:

- Assume $c_f: S \rightarrow R_{\geq 0}$, with $S \subset R^n$ is compact and nonempty [41].
- A solution $w^* \in S$ is named global optima when $c_f(w^*) \leq c_f(w)$ at any w $\in S$ exists.
- A solution $w^* \in S$ is named local optima if there exists $\varepsilon > 0$, and ε-neighborhood $B_\varepsilon(w^*, \varepsilon)$ near w^* where exists $c_f(w^*) \leq c_f(w)$ for any w $\in S \cap B_\varepsilon(w^*, \varepsilon)$.

Vanishing and Exploding Gradients: Gradients can be extremely tiny (vanishing) or too big (exploding) within the backpropagation, affecting the training stability. Result can be difficulty in learning long-term dependencies (vanishing gradients) or unstable training (exploding gradients) [1, 34].

Overfitting: Models may memorize the training samples instead of being able to generalize to new data, unseen samples. This might reduce the performance on new data, limiting the model's capability for generalization. Figure 2.6 [1, 35, 36].

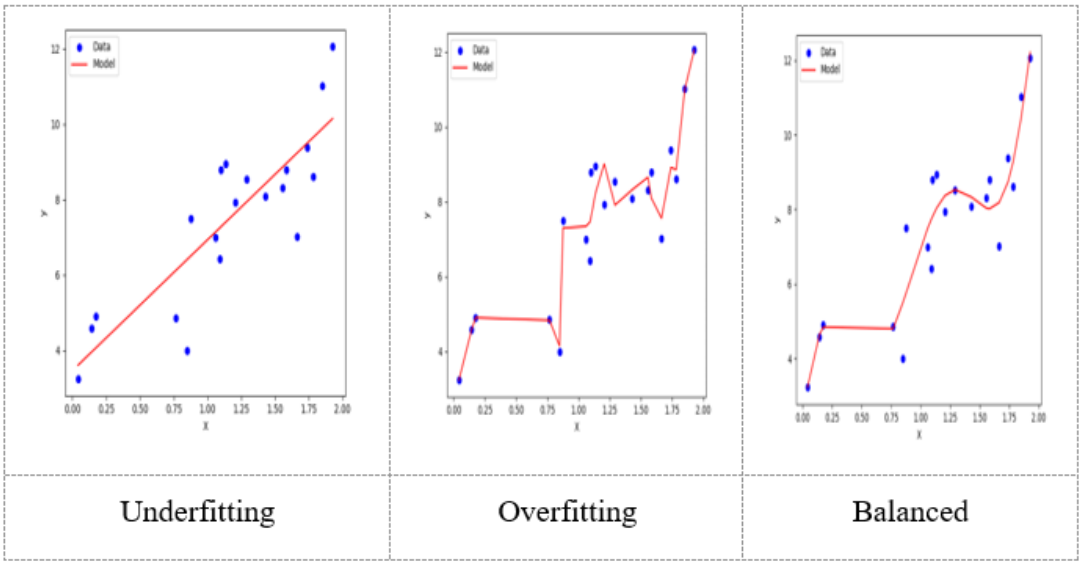


Figure 2.6. Over-fitting and under-fitting issues.

Computational Complexity: Deep neural networks with numerous parameters can be computationally expensive to train. Increasing training time, resource requirements, and potential limitations on deployment in resource-constrained environments. [1]

Limited Transferability: Models trained on one task may not generalize well to related tasks. Requiring extensive retraining for new tasks, limiting the transferability of learned features [1, 37].

2.2. Meta Learning

Meta-learning, often regarded as learning to learn, was initially introduced by [38] and then has garnered substantial attention in recent years for its capacity to address challenges posed by limited training data. This section delves into the meta-learning landscape, delineating its types, applications, and notable models. A concise review lays the groundwork for understanding the complications of meta-learning, setting the scene for the exploration of our novel dynamic heuristic approach.

Meta-learning includes training a network to learn how to learn, enabling it to adapt quickly to unseen tasks with few data points, Figure 2.7. [40] It employs a two-step process: in the first step, the model learns from a diverse set of tasks to acquire general knowledge or "meta-knowledge." In the second step, this meta-knowledge is used to rapidly adapt to new tasks.[6] The model typically leverages a meta-learning algorithm, such as MAML (Model-Agnostic Meta-Learning) [39], to update its parameters in a way that facilitates quick adaptation. Meta-learning is beneficial in scenarios where limited data is available for new tasks, as it enables models to generalize effectively and perform well with fewer examples [7].

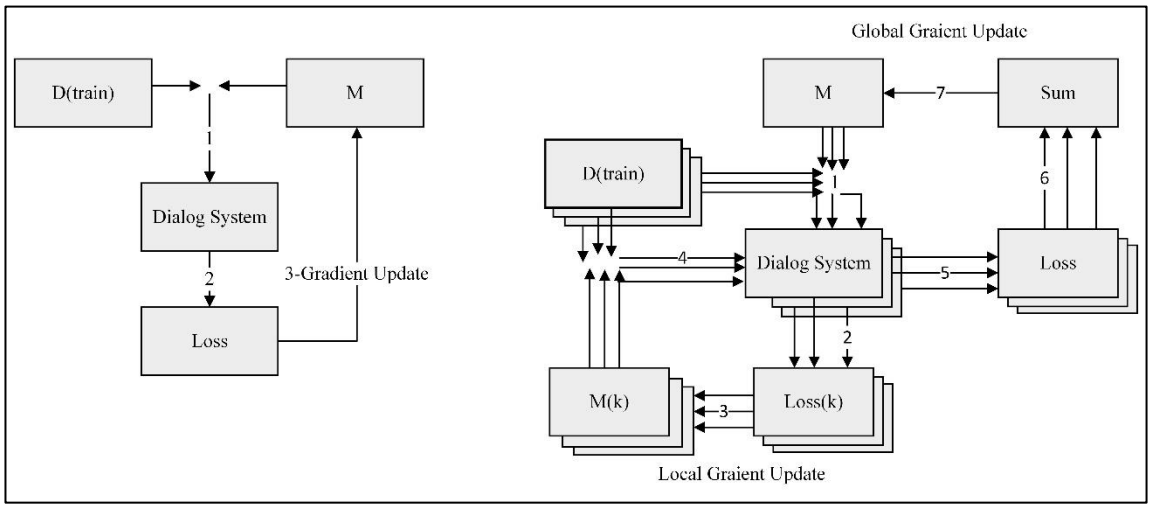


Figure 2.7. Classic gradient update vs. MAML update.

Meta-learning, transfer learning and multitask learning are unique methodologies crafted to facilitate learning across a spectrum of tasks. [37]. Multitask learning seeks to enhance generalization against a group of tasks by simultaneously learning them. Transfer learning involves refining a pre-trained model for a unseen task using limited

data. In contrast, meta-learning involves extracting valuable insights from prior tasks and utilizing them to efficiently learn new tasks [7, 41].

Meta-learning strives to overcome the constraints of conventional transfer learning by embracing a more advanced strategy that explicitly prioritizes transferability. Unlike traditional transfer learning, which entails pre-training a model on source tasks and fine-tuning it for a new task, meta-learning involves training a network to adeptly learn or adapt to new tasks with minimal examples [7, 42]. During meta-training, the focus is on acquiring the ability to learn various tasks, while at meta-test time, the emphasis is on efficiently mastering a new task.

2.2.1. Meta-learning: task-distribution view

In meta-learning, we often have a meta-training phase where the model learns a good initialization from a set of tasks. The model can then quickly adapt to new tasks during the meta-testing phase [7]. Here's a generic representation:

Meta-Training Objective: The meta-training objective aims to find model parameters that generalize well across different tasks. Given a set of tasks $\mathcal{T}$, the target is to reduce the total loss across all tasks [8]:

$$\min_{\theta} \sum_{T \in \mathcal{T}} L_T(\theta) \tag{2.10}$$

Where $L_T(\theta)$ is the loss on a task T calculated for the model parameters θ. This objective guides a model to learn a set of parameters which perform good on a several number of tasks.

Gradient Descent Updates (First Order): During meta-training, the model parameters are updated using gradient descent. The first-order meta-training update is expressed as [7]:

$$\theta' = \theta - \alpha \nabla_{\theta} \sum_{T \in T} L_T(\theta) \tag{2.11}$$

Here, α represents the meta-training learning rate, while the gradient term represents the sum of gradients of the losses on all tasks calculated against the model parameters.

Meta-Testing Objective: In the meta-testing phase, the model parameters θ' are adapted to a new task. The target is to reduce the loss on the new task L_{new} with respect to the adapted parameters [7]:

$$\theta'' = \theta' - \beta \nabla_{\theta} L_{\text{new}}(\theta') \tag{2.12}$$

Where β is the meta-testing learning rate. This equation captures the idea of quick adaptation to a new task using a few additional gradient steps.

Higher-Order Gradients: In certain meta-learning algorithms, higher-order gradients are computed to facilitate faster adaptation. For instance, in MAML framework, the second-order update is given by [39]:

$$\theta'' = \theta - \beta \nabla_{\theta} \left(\sum_{T \in T} L_T \left(\theta - \alpha \nabla_{\theta} L_T(\theta) \right) \right) \tag{2.13}$$

This involves calculating the derivative of the meta-objective with regard to the model parameters and is particularly useful where the network needs to adapt fast with minimal data.

2.2.2. Meta learning approaches

In meta-learning, various learning strategies can be organized into three types: metric-based, model-based, and optimization-based techniques, These categories represent distinct strategies for enabling models to learn how to learn efficiently across diverse tasks [7].

Metric-based meta-learning:

Also known as distance-based or similarity-based meta-learning, aims to is to enable a model to quickly adapt to new tasks by learning a metric or loss function that calculates the similarity or dissimilarity between examples [42]. In metric-based meta-learning, the idea is often to embed examples into a metric space in such a way that examples from the same task are close to each other, and examples from different tasks are far apart. This learned metric is then used during the adaptation phase to quickly identify relationships between examples in a new task.

Siamese networks, a groundbreaking contribution by Koch et al. in 2015 [43], represent the base of this category of meta-learning strategies within few-shot learning

domains. This pioneering approach involves estimating classes by matching inputs of both support and query sets. The significance of Siamese networks lies in their introduction of the fundamental idea that subsequently evolved in various directions summarized in Figure 2.8 [7].

Graph neural networks (GNNs), as advanced by Hamilton et al. in 2017 [44] and Garcia and Bruna in 2017 [45], expanded upon the Siamese network concept. GNNs introduced a parametric information flow between support and query inputs, offering a more flexible framework for metric-based meta-learning techniques.

Matching networks, a direct offshoot inspired by Siamese networks, were introduced by Vinyals et al. in 2016 [46]. While retaining the core idea of comparing inputs for predictions, matching networks departed by training directly in the few-shot setting. Notably, they employed cosine similarity as a metric, omitting the auxiliary binary classification task utilized by Siamese networks.

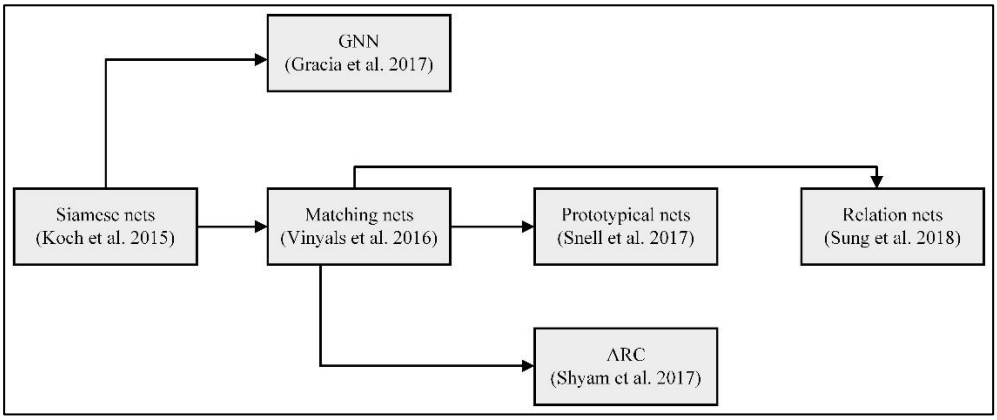


Figure 2.8. The evolution of covered metric-based meta-learning strategies.

Prototypical networks, proposed by Snell et al. in 2017 [47], further refined the methodology of input matching. Rather than matching each query set data point into individual support set examples, prototypical networks introduced the innovative concept of comparing with a class prototype. This strategic adjustment reduced the number of necessary input comparisons.

Relation networks, as detailed by Sung et al. in 2018 [48], marked a notable evolution by replacing fixed, predefined similarity metrics. Instead, Relation networks substituted these metrics with a neural network, allowing to the learning of domain-specific similarity functions. This adaptation enhanced the adaptability and performance of the model.

Attentive Recurrent Comparators (ARCs), a distinctive approach introduced by Shyam et al. in 2017 [49], took a biologically plausible perspective. ARCs focused on several interleaved glimpses at different sections of the inputs during the comparison process, departing from the conventional practice of comparing entire inputs. This nuanced approach in ARCs offered a detailed and sophisticated assessment during the meta-learning process.

The main benefits of employing metric-based techniques include the simplicity of the underlying concept in similarity-based predictions and the potential for fast test-time execution in small tasks, as the network does not require task-specific tuning. Nevertheless, as tasks at meta-test time deviate further from those used during meta-training, metric-learning techniques struggle to assimilate new task information into the network weights. Consequently, this limitation sometimes results in performance drop.

Model-based meta-learning:

Model-based meta-learning is a technique within the broader field of meta-learning that involves learning a model's internal representations or dynamics during meta-training in order to facilitate rapid adaptation to new tasks during meta-testing [7].

In model-based meta-learning, the focus is on learning how to update a model's parameters quickly for a new task. Instead of directly learning the parameters that make the model effective for adaptation, the algorithm learns a model that can predict how the parameters should be updated based on a few examples from a new task.

Memory-augmented neural networks (MANN), introduced by Santoro et al. in 2016 [50], represent a significant advancement in the field of deep model-based meta-learning techniques. This pioneering approach involves sequentially feeding the entire support set into the model, leveraging its internal state to make predictions for the query set inputs. Several research extended MANN summarized in Figure 2.9:

The model-based paradigm, characterized by the sequential entry of inputs, found resonance in recurrent meta-learners (RMLs), a concept explored by Duan et al. [51] in 2016 and Wang et al. [52] in the same year, particularly in the context of reinforcement learning. RMLs share a similar strategy of processing inputs in a sequential manner.

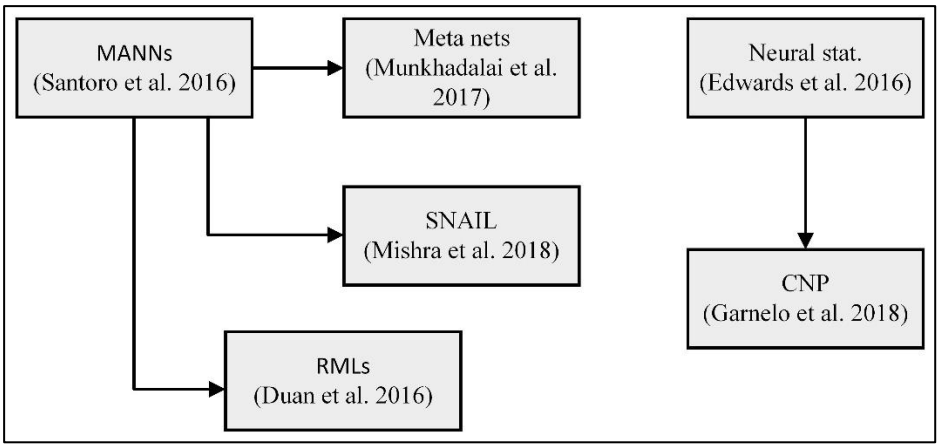


Figure 2.9. The history of model-based meta-learning techniques[7].

Meta networks, proposed by Munkhdalai and Yu in 2017 [53], marked another evolution in model-based meta-learning. In contrast to memory-augmented neural networks, meta networks employed a big black-box technique but innovatively generated task-specific values for each encountered task, showcasing the adaptability of model-based techniques.

SNAIL (Mishra et al. 2018) [54] emerged as a distinctive effort to augment memory capacity and refine the ability to pinpoint memories, addressing inherent limitations in recurrent neural networks. This improvement was achieved through the incorporation of attention mechanisms coupled with special temporal layers, representing a noteworthy refinement in the model-based meta-learning approach.

Furthermore, the neural statistician and conditional neural processes introduced two innovative techniques aiming to learn meta-features of datasets in an end-to-end fashion. The neural statistician, relying on the distance between meta-features, made class predictions, while the conditional neural process conditioned classifiers on these features, showcasing the versatility within the model-based meta-learning landscape [7].

While model-based approaches offer advantages such as flexibility in internal system dynamics and broader applicability, it is important to note their comparative performance. Garcia and Bruna (2017) [45] demonstrated that metric-based techniques, specifically in supervised settings, often outperform model-based approaches, as exemplified by graph neural networks. Additionally, challenges such as suboptimal performance with larger datasets, as highlighted by Hospedales et al. in

2020 [6], and a reduced ability to generalize to several variant tasks compared to optimization-based techniques, will be discussed in the subsequent section.

Optimization-based meta-learning:

Optimization-based meta-learning is a category of meta-learning algorithms that focuses on training a model to adapt fast to unseen tasks through optimization procedures. The key idea is to learn a set of model parameters that can be efficiently fine-tuned for a unseen tasks with a limited amount of data points [7]. The optimization-based meta-learning approach is versatile and can be applied to various types of models and tasks. It provides a framework for learning an initialization that guarantees a quick adaptation to new and unseen tasks, making it particularly useful in scenarios where data for new tasks is limited [39].

The LSTM optimizer, introduced by Andrychowicz et al. in 2016 [55], serves as the foundational concept for optimization-based meta-learning techniques. This pioneering approach replaces handcrafted optimization procedures, such as gradient descent, with a trainable LSTM network. The LSTM meta-learner, as proposed by Ravi and Larochelle in 2017 [56], extends this paradigm into the few-shot learning setting. Beyond learning the optimization procedure, this meta-learner also acquires a set of initial weights, enabling its versatile application across various tasks.

MAML (Finn et al. 2017) [39] simplifies the LSTM meta-learner by replacing the trainable LSTM optimizer with handcrafted gradient descent, Figure 2.10. Widely recognized in the deep meta-learning field, MAML has inspired numerous subsequent works, showcasing its impact and relevance in the research community.

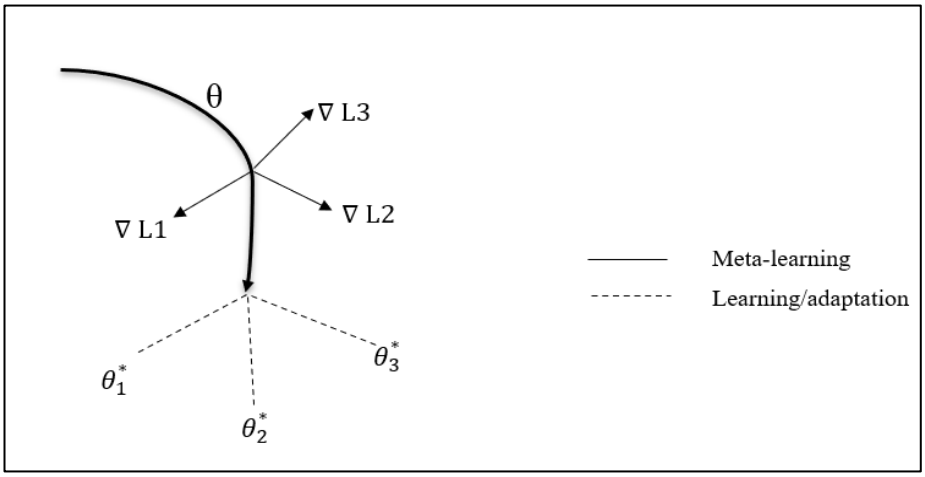


Figure 2.10. MAML diagram.

MAML has garnered significant interest in the domain of Deep Meta-Learning, Figure 2.11, likely attributed to its: (i) simplicity, requiring just two parameters, (ii) broad implementations, and (iii) robust performance [39, 7]. However, a drawback of MAML, as noted earlier, is its potential computational expense, both in with regard to performance represented in memory utilization and total running time needed, particularly when optimizing a base-learner for each task and computing higher-order derivatives shaping the optimization paths. [7, 8]

Meta-SGD [57], an enhancement of MAML introduced by Li et al. in 2017, takes a step further by incorporating the learning rates into the optimization process. This refinement demonstrates an additional layer of sophistication in fine-tuning the optimization-based meta-learning approach.

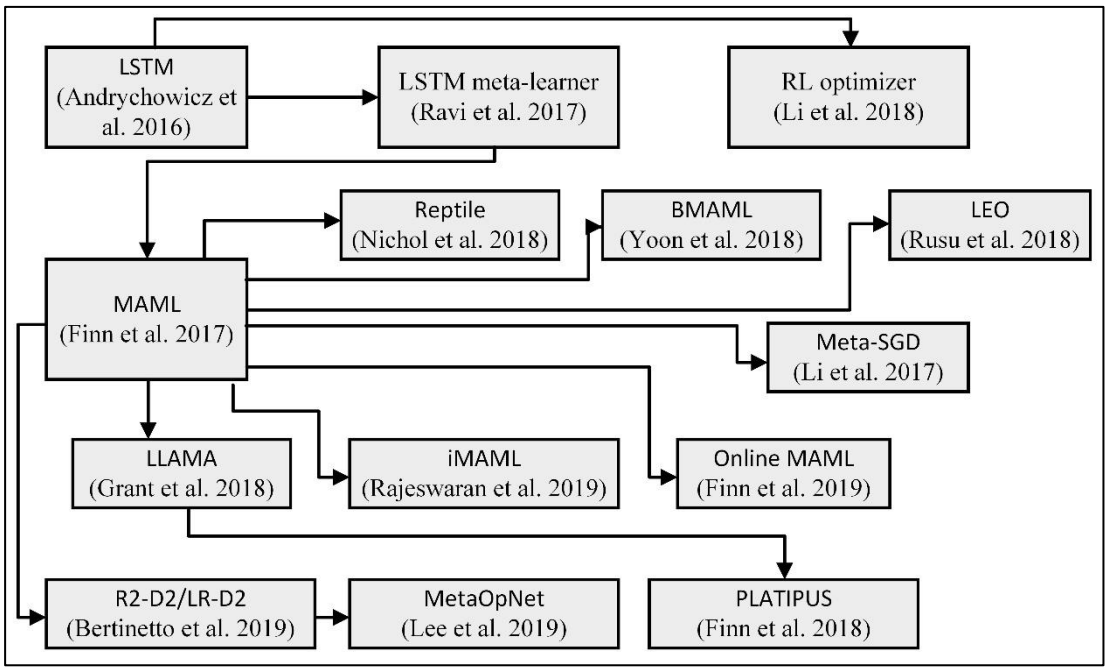


Figure 2.11. The covered optimization-based meta-learning techniques.

LLAMA (Grant et al. 2018) [58], PLATIPUS (Finn et al. 2018), and online MAML (Finn et al. 2019) [59] extend MAML's applicability to active and online learning settings. LLAMA and PLATIPUS introduce probabilistic interpretations of MAML, allowing for the sampling of several solutions for a specific task and quantifying uncertainty. These extensions contribute to a more nuanced understanding and utilization of MAML in diverse learning scenarios [7].

BMAML (Yoon et al. 2018) [60] takes a distinctive approach by jointly optimizing a discrete set of M initializations. This discrete optimization strategy represents a departure from traditional continuous optimization, demonstrating the diversity and creativity within optimization-based meta-learning research.

iMAML (Rajeswaran et al. 2019) [61] addresses the computational challenges associated with the computation of second-order derivatives needed by MAML. Through implicit differentiation, iMAML not only overcomes computational expenses but also enables the utilization of derivation-free inner loop optimization cycle. This innovative solution opens avenues for more efficient and versatile optimization-based meta-learning techniques.

Reptile (Nichol et al. 2018) [62] presents an elegant first-order meta-learning algorithm that finds a set of initial parameters without the need for computing higher-order derivatives. This streamlined approach contributes to the efficiency and scalability of optimization-based meta-learning.

LEO (Rusu et al. 2018) [63] seeks to enhance the performance of MAML by making improvements in a lower-dimensional parameter domain using an encoder-decoder architecture. This architectural innovation showcases the exploration of novel design principles within the optimization-based meta-learning framework.

Lastly, (Bertinetto et al. 2019) [64], and Lee et al. (2019) [65] leverage traditional deep learning techniques (ridge regression, logistic regression, SVM, respectively) as classifiers on top of a learned feature extractor. These approaches integrate classical methods into the meta-learning framework, demonstrating the adaptability and fusion of traditional techniques with contemporary meta-learning concepts [7].

The main benefit of optimization-based techniques, highlighted by Finn and Levine in 2018 [58, 66], is their ability to achieve superior performance on wider task distributions compared to model-based approaches. However, it is essential to acknowledge the computational expenses associated with optimization-based techniques, as emphasized by Hospedales et al. in 2020 [6], particularly in optimizing a base-learner for each task and learning the optimization procedure.

When the target is to implement heuristic approach to optimize the meta learning model, it will be more efficient to apply on optimizer-based meta-learning approaches rather than the other two, due to their ability to efficiently explore solution spaces,

adapt to diverse tasks, and incorporate prior knowledge. Unlike metric-based approaches, which rely heavily on similarity measures and may struggle with diverse tasks, and model-based approaches, which require explicit modeling of the underlying process and may lack flexibility, optimizer-based methods directly optimize the search process, allowing for robust adaptation and effective exploration of solution spaces, thus offering superior performance and applicability across various domains.

2.2.3. Meta learning challenges

While meta-learning has demonstrated its potential to revolutionize learning processes across diverse domains, unlocking the ability to rapidly adapt to new tasks and acquire generalized knowledge, it also comes with its set of challenges. Some of the key challenges in meta-learning include:

- **Time Complexity:** There is often a trade-off between the performance of a meta-learning algorithm and its time complexity. Striking the right balance is crucial, especially in scenarios where quick adaptation is essential [6, 7]. Meta-learning models often involve two phases: meta-training and task-specific adaptation. The time complexity of meta-learning includes the time required to train the meta-model and the adaptation time for new tasks [42].
- **Model Complexity:** The complexity of the meta-learning model itself can impact both performance and time complexity. More complex models may offer better performance but might require more time for training and adaptation [41].
- **Algorithmic Innovations:** Researchers continually work on developing new meta-learning algorithms that improve both performance and time complexity. Innovations in optimization techniques, model architectures, and training strategies contribute to advancements in this field [8].

2.3. Heuristic Algorithms

Heuristic algorithms, rooted in optimization principles, play a pivotal role in our proposed methodology. This section provides a focused exploration of heuristic algorithms, emphasizing their relevance in addressing the characteristics of deep learning in general and meta-learning specifically. By examining how heuristic algorithms bridge the gap between optimization and learning, we go through their

importance in our explore of enhancing the adaptability and efficiency of meta-learning models [11].

2.3.1. Characteristics of heuristics algorithms

- **Simplicity:** Heuristic algorithms are designed to be straightforward and easy to apply. They provide quick decision-making without intricate calculations or exhaustive exploration of all possible solutions [9, 11].
- **Efficiency:** Heuristics prioritize speed in generating solutions. They aim to provide reasonably good outcomes within a limited amount of time, making them well-suited for scenarios where computational resources are constrained [67].
- **Adaptability:** Heuristics often present adaptability, allowing them to be applied across various problem domains. They rely on general problem-solving principles, making them important techniques in diverse contexts [67, 68].
- **Approximate Solutions:** Rather than the possibility of finding optimal solutions, heuristics offer approximate solutions that are often "good enough" for the specific problem at hand. This characteristic is particularly valuable in situations where exact solutions are computationally expensive or even infeasible [69].

These algorithms, designed to navigate complex problem spaces efficiently, present diverse strategies that can be systematically organized into distinct categories. Categorizing those algorithms by source of inspiration serves to encapsulate common principles and approaches shared by various heuristic methods, facilitating a clearer understanding of their operational frameworks, summarized in Figure 2.12.

2.3.2. Evolutionary algorithms

Darwinian principles, specifically the concepts of natural selection and survival of the fittest, serve as the foundation for Evolutionary Algorithms (EAs). These algorithms initiate with a group of individuals, orchestrating simulated processes of reproduction and mutation to generate a new offspring generation. This iterative approach ensures the preservation of genetic traits that enhance an individual's adaptability to a given environment, while concurrently eliminating characteristics that render it less resilient.

The theories articulated by Charles Darwin, particularly those pertaining to natural evolution, serve as a driving force behind Genetic Algorithms (GAs) and Differential

Evolution (DE). Genetic Programming (GP), on the other hand, draws inspiration from the biological evolution paradigm. EAs encompass a diverse array of methodologies, such as Gene Expression Programming (GEP), Learning Classifier Systems (LCS), Neuroevolution (NE), and Evolution Strategy (ES), each reflecting the assimilation of evolutionary principles into computational frameworks [9, 11, 67, 68].

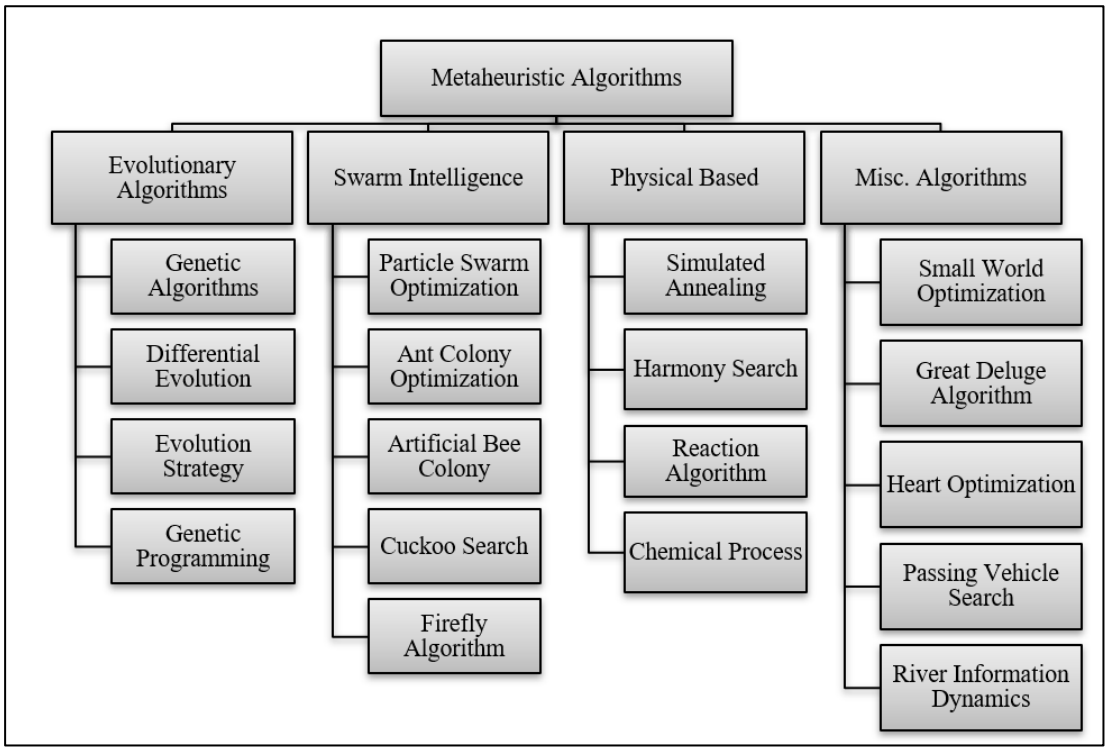


Figure 2.12. Classification of heuristic algorithms based on the source of inspiration with including popular algorithms under each class.

Beni and Wang (1993) [71] introduced the term 'Swarm Intelligence' in 1989 within the realm of cellular robotic systems. Since then, Swarm Intelligence (SI) has garnered significant attention across various industries. SI entails the collective behavior of decentralized and self-organized systems. Key attributes of swarm systems include adaptability, characterized by learning through action, along with robust communication and knowledge-sharing capabilities [11]. In the wild, creatures take on tasks such as defending themselves against formidable predators or hunting for sustenance independently, yet they exhibit a strong reliance on swarming. Even in their search for food, they exhibit this swarming behavior. Swarm intelligence (SI) has served as inspiration for numerous methodologies. For instance, the intelligent social interactions observed in birds have motivated the development of algorithms like particle swarm optimization (PSO) [10]. The process of monkeys climbing trees in

search of food has inspired the development of the monkey search (MS) algorithm [72]. Moreover, grey wolf leadership hierarchy and hunting strategy motivates grey wolf optimizer (GWO), and so on. SI examples include, but are not limited to, ant lion optimizer (ALO) [73], bat algorithm (BA) [74], firefly algorithm (FA) [75], ant colony optimization (ACO) [76], cuckoo search (CS) [77], artificial bee colony (ABC) [78]. Polar Bear Optimization [79].

While the implementation details vary from one algorithm to another, we can provide a general formulation that highlights key concepts often found in swarm intelligence algorithms as summarized in [9, 11] and [67].

Population: Let N be the number of agents in the swarm. The swarm population is represented by:

$X = \{X_1, X_2, \dots, X_N\}$ where X_i is the state of agent i, each X_i typically represents a candidate solution in the search space.

Distribution of Agents: The distribution of agents in the search space is governed by a probability density function $P(X)$ which describes the likelihood of finding an agent at a particular state.

Objective Function: For optimization problems, there exists an objective function f that maps each agent's state to a real value:

$$f(X_i) \to R \tag{2.14}$$

Local Search: Agents perform local search around their current positions to explore the neighborhood. Let L_i denote the local search space for agent i, and $f_{\text{local}}(X_i)$ be the local objective function:

$$X_i(t+1) = \text{argmin}_{X \in L_i} f_{\text{local}}(X) \tag{2.15}$$

Global Search: Agents share information to perform a global search. The global search space is denoted as G, and $f_{\text{global}}(X)$ is the global objective function:

$$X_i(t+1) = \text{argmin}_{X \in G} f_{\text{global}}(X) \tag{2.16}$$

Update Equations: Agents update their positions based on their local search, global search, and possibly historical information:

$$X_i(t+1) = A(X_i(t), \text{local info}, \text{global info}, \text{historical info}) \quad (2.17)$$

2.2.3. Physical law-based algorithms

Algorithms that draw inspiration from principles rooted in the physical and chemical realms belong to this specific category. Furthermore, they can be categorized as follows:

(i) Physics-based algorithms [11]:

This subcategory draws inspiration from key cosmic phenomena such as gravitation, the big bang, black holes, galaxies, and fields. Concepts underlying this category stem from phenomena like the devouring of stars by black holes and the genesis of new beginnings, which have inspired algorithms like the Black Hole Algorithm (BH). Harmony Search (HS) [80], on the other hand, is crafted from the improvisational methods employed by musicians. Simulated Annealing (SA) [81] is grounded in the metallurgical annealing process, where metal is rapidly heated and then slowly cooled, enhancing strength and facilitating ease of manipulation.

(ii) Chemistry-based algorithms:

Metaheuristic Algorithms (MAs) inspired by the principles of chemical reactions, including molecular reaction, Brownian motion [82] falls into this category.

2.3.4. Miscellaneous algorithms

This category encompasses algorithms rooted in a wide array of concepts including human behaviors, game strategies, mathematical theorems, politics, artificial thoughts, and various other topics. For instance, the movement and propagation of clouds have inspired the development of the Atmosphere Clouds Model Optimization Algorithm (ACMO) [83], while activities in the stock market, particularly the trading of shares, drive the Exchange Market Algorithm (EMA) [84].

2.4. Heuristic Algorithms and Deep Learning

Neural network optimization is a multifaceted challenge involving various aspects such as weight optimization, hyperparameter tuning, and architecture design. This

literature review categorizes heuristic algorithms based on specific optimization areas, providing insights into their applications and effectiveness in addressing distinct challenges within neural network optimization [70, 5].

More and more researchers are leaning towards crafting novel hybrid optimization tools. These tools combine two or more metaheuristic algorithms to fine-tune the training parameters of artificial neural networks. With the introduction of numerous metaheuristic algorithms, the choice of an appropriate algorithm for hybridization emerges as a crucial factor in crafting superior algorithms. The performance of optimization tools is generally contingent on two essential components: exploitation and exploration, synonymous with intensification and diversification [70].

The search process efficiently traverses the search domain, conducting a global search to evade local optima but may encounter slow convergence. Conversely, the exploitation process tends to yield very high convergence rates but risks being confined to a local optimum. Achieving an optimal algorithm performance necessitates striking a delicate balance between these two components. Currently, no algorithm can assert having attained such equilibrium in the existing literature [5].

Among the developed metaheuristic algorithms, certain algorithms excel in convergence rates, while others prove adept at avoiding local minima. For instance, Particle Swarm Optimization (PSO) achieves faster convergence in optimizing ANN models [85], yet for multimodal functions, both Differential Evolution (DE) and Harmony Search (HS) exhibit superior convergence rates compared to PSO [86, 87]. Furthermore, Artificial Bee Colony (ABC) demonstrate faster convergence than Genetic Algorithms (GA) for several benchmarks in training Feedforward Neural Networks (FNN) for classification purposes [88].

Generalized Simulated Annealing (GSA) may suffer from a slow convergence behavior in the final cycles but has been proven to converge much quicker than GA in dynamic neural network identification [70]. Conversely, Simulated Annealing (SA) exhibits slower convergence in solving machining optimization problems [89]. Tabu Search (TS) is demonstrated to converge quicker than GA in resolving both quadratic kind of tasks and vehicle routing problems [5].

Cuckoo Search (CS) and GSA are widely employed due to their proficiency in finding the global optimum. CS outperforms FA and PSO in optimizing retaining wall design

and is efficient in achieving global optima with higher success rates than PSO and GA for solving optimization problems [70]. Additionally, both PSO and ABC achieve higher accuracy compared to FA in optimizing ANN models. However, FA shows a higher tendency to be trapped in local minima compared with ABC in training FNN for classification problems [5].

ABC's strong exploration ability results in a lower tendency to be trapped in local optima compared to PSO, which exhibits a weaker exploration ability [88]. In stock price prediction, both PSO and DE can avoid local minima, but DE provides better accuracy, particularly in fluctuated time series [5]. DE also showed better performance than GA, PSO, and SA methods in solving various problems. For multimodal functions, both HS and DE exhibit higher performance than PSO, while HS proves more efficient in finding optimal solutions than PSO. TS is commonly used for hybridization with other algorithms to overcome their weakness of being trapped in local optima [70].

2.5. Current Landscape and Gaps in the Literature

Meta-learning has emerged as a compelling area of study, particularly in addressing challenges that come with supervised learning problems characterized by limited training data. A substantial body of research attests to the effectiveness of meta-learning in facilitating generalization and rapid adaptation to new tasks with minimal data points. To train these meta-learning models, conventional practice involves the use of gradient-based optimizers. However, the unique challenges presented by meta-learning, such as the demand for swift adaptation and the necessity of task-specific adjustments, suggest that standard optimizers may not be optimally suited to address these issues due to the need of intensive training cycle in order to achieve the desired accuracies.

Despite the promising experiments made in the field of meta-learning, a critical examination of the existing literature reveals significant performance issues that persist. The reported shortcomings encompass challenges related to the convergence speed, accuracy, and adaptability of meta-learning models, particularly in scenarios with lack of training data. These limitations underscore the need for innovative approaches to enhance the efficiency of meta-learning.

A comprehensive review of the literature results a common trend in the use of heuristics and traditional gradient-based optimizers in the training of deep learning models. However, it is noteworthy that existing studies generally focus on general deep learning models, and there is a conspicuous dearth of attention directed specifically at optimizing meta-learning models. While heuristics have demonstrated efficacy in certain contexts, their application has not been systematically explored in the realm of meta-learning, leaving a significant gap in the literature.

This research seeks to address these gaps by proposing a novel hybrid optimizer tailored explicitly for meta-learning models. Recognizing that standard optimizers may not efficiently handle the unique demands of meta-learning, we introduce an innovative approach. Our custom optimizer combines a metaheuristic algorithm with traditional gradient-based techniques. This hybrid approach involves an initial search for optimal model parameters using random candidate solutions, followed by iterative adjustments based on individual performance within the population. The subsequent learning process incorporates classic gradient optimization, to continue with the most promising solution identified by the heuristic algorithm.

To prove the efficacy of our proposed custom optimizer, we first conducted experiments on five classification benchmark datasets. The results demonstrated superior accuracy and faster convergence compared to conventional approaches. Moreover, we extended our methodology to address the meta-learning problem, illustrating how our custom optimizer enhances the training of meta-learning models and facilitates the efficient identification of optimal parameters. Our experimental analysis focuses on the dynamic characteristics of our proposed strategy, emphasizing its potential to overcome the performance issues observed in traditional meta-learning models.

3. DYNAMIC POPULATION OPTIMIZATION

In this section, we present the proposed methodology for our research, providing a comprehensive framework for the design details of our hybrid algorithm.

3.1. Introduction

We will explain the fundamentals of our Dynamic Population Optimization (DPO) algorithm. DPO represents a combination of metaheuristic algorithms and gradient descent optimization, summarized in Figure 3.1, which was specifically developed for training neural networks in the context of classification problems.

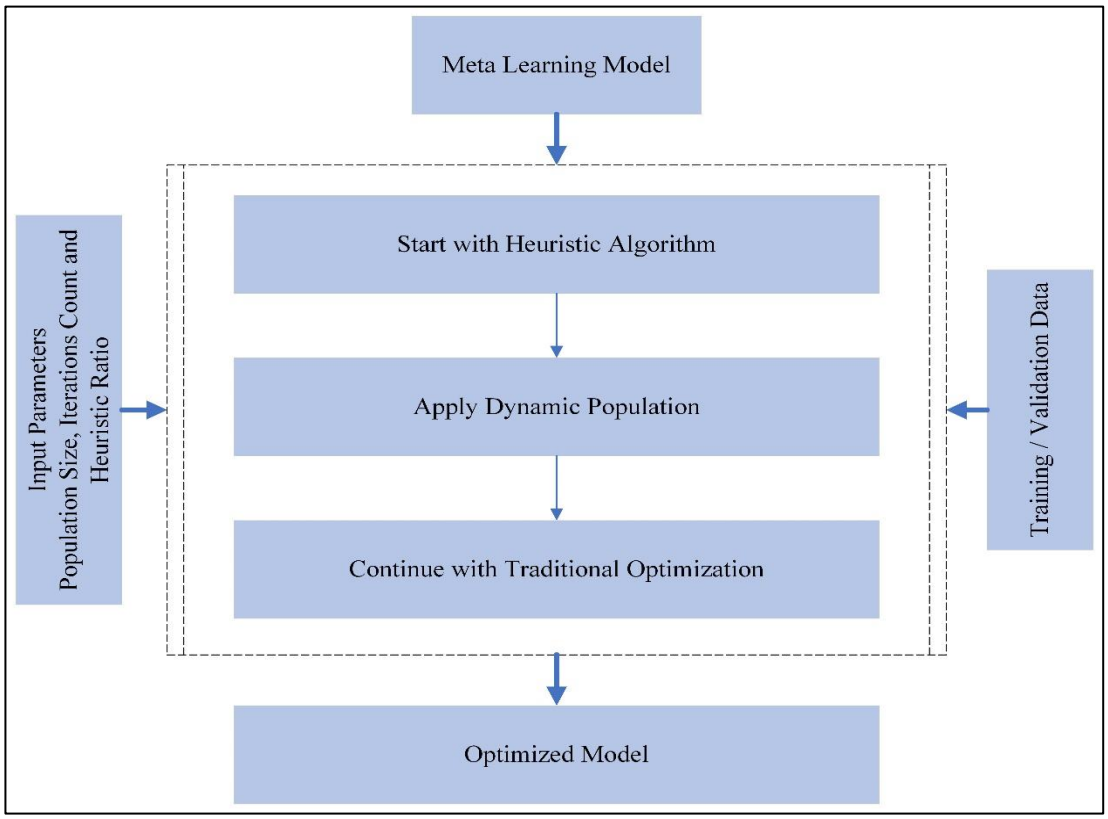


Figure 3.1. Research Framework.

In the ever-evolving landscape of machine learning, finding an efficient and adaptable optimization strategies has become essential, especially as we deal with the complexity of meta learning explained in the previous chapter.

Meta learning, characterized by its ability to enable models to learn and adapt across various tasks, introduces a level of complexity that demands a strategic and systematic approach.

Recognizing the challenges posed by meta learning, we propose a two-phase strategy for the development and evaluation of an optimization function based on heuristic principles. The significance of this approach lies in the acknowledgment of the complexity posed by meta learning and the performance constraints of our optimizer before navigating the complexities of meta-learning scenarios.

However, given the complexity of meta learning extends beyond the diversity of tasks. Meta learning often involves handling limited labeled data, requiring innovative solutions. These difficulties emphasize the need for a robust optimization framework capable of navigating the details of meta learning scenarios.

The Rationality of Two Phases Development: Given the increased challenges associated with meta learning, we propose a that begins with applying the optimization of our framework on traditional classification problems. This initial phase serves as a basic test, ensuring that the optimizer is not only effective but also adaptable across a spectrum of benchmark classification datasets. By validating and fine-tuning the optimization function in traditional classification scenarios, we allow smooth transition into the complex realm of meta learning with the need only for minor tweaks. This phased methodology aligns with a wise strategy of "mastering the basics before tackling the complexities."

Ensuring Improved Performance: Before delving into the challenges of meta learning, it is imperative to establish a solid foundation. The first phase allows us to gauge the optimizer's efficacy in handling diverse classification tasks, providing insights into its adaptability and robustness. Only once the optimizer demonstrates consistent and stable performance across various datasets can we confidently transition to the nuanced domain of meta learning.

In effect, this two-phased approach aligns with a principle of cautious progression, ensuring that our optimization function is not only sophisticated but also reliable. By doing so, we aim to contribute a solution that stands resilient against the multifaceted challenges of meta learning, ultimately advancing the frontier of machine learning optimization strategies.

3.2. General Classification Perspective

In the effort of refining the classification process within supervised learning problems, our objective is to locate the optimum values for the network parameters (weights vector). To accomplish this, we introduce a novel approach, a hybrid model that integrates a heuristic derivative-free heuristic technique with traditional gradient descent-based optimization. In the landscape of supervised learning, where the model relies on labeled training data to make predictions, the significance of parameter optimization is the core function of the training process. The proposed technique enrich the reliance on gradient information, allowing our model to navigate the parameter space with agility and adaptability. Concurrently, the incorporation of dynamic population optimization adds a layer of intelligence, enabling the model to dynamically adjust its parameters based on evolving patterns within the training data. Integrating heuristic approaches with traditional gradient-based training offers a solution to the optimization challenges in high-dimensional search domains. By combining the global exploration capabilities of heuristics with the local search proficiency of gradient-based methods, this hybrid approach aims to overcome the limitations inherent in each technique. Heuristic algorithms excel in navigating complex and expansive search spaces but may struggle to find precise solutions, while gradient-based methods are adept at fine-tuning solutions in local regions but can be trapped in local optima. By leveraging the strengths of both paradigms, hybrid optimization strategies can be designed that strike a balance between exploration and exploitation, leading to more robust and effective optimization outcomes across a wide range of complex problems.

Gradient-based optimization techniques proceed iteratively, sequentially refining the solution from an initial point within the search space. However, the efficacy of these techniques hinges greatly on the selection of the starting point, as it can determine whether the algorithm converges to a global optimum or gets stuck in a local optimum [3, 4]. Recognizing this, a comprehensive exploration of the search domain is conducted initially to identify a promising starting point. This exploration involves systematically traversing the search space to evaluate various starting points, assessing their potential to lead to optimal solutions. By doing so, the optimization process aims to mitigate the risk of being confined to suboptimal solutions and enhance the chances of achieving a globally optimal outcome.

During the initialization phase, a heuristic algorithm efficiently navigates the search space to identify a promising starting point for the subsequent training phase. This process involves systematically evaluating candidate points within the space, leveraging heuristic techniques to prioritize regions likely to contain optimal solutions. By swiftly exploring the search space in this manner, the initialization phase aims to expedite the convergence towards a high-quality solution in the subsequent stages of training.

Every individual in the population P holds values of the weight vector of a neural network:

$$P = p1, p2, p3, \dots, p_n \tag{3.1}$$

$$\theta = (w_1, w_2, w_3, \dots, w_m) \tag{3.2}$$

Where n and m represent the number of individuals and the total number of parameters –weights- that should be optimized respectively. The first stage within our algorithm is starting by dispersing randomly the members of population P within the search domain, subsequently, each object will have the opportunity to explore its surrounding area in search of an improved solution. Evaluations will be conducted based on Equation 3.3.

$$Fitness(w_i) = \frac{1}{1 + C_i} \tag{3.3}$$

Since this fitness will represent the performance of one individual -absolute performance-, however, to guarantee fair comparison with other members of the population we will be calculating the normalized fitness [70]:

$$NormalizedFitness(w_i) = \frac{Fitness(w_i)}{\sum_{j=1}^{n} Fitness(w_j)} \tag{3.4}$$

The population after calculating the fitness for each member will be represented as following matrix:

$$P = \begin{bmatrix} \theta_1 & f_1 \\ \theta_2 & f_2 \\ \theta_3 & f_3 \\ \vdots & \vdots \\ \theta_n & f_n \end{bmatrix} \tag{3.5}$$

Where f_i represents the normalized fitness value of the i^{th} individual.

Selecting a good local search function is crucial when designing a heuristic algorithm, because it directly influences the algorithm's ability to efficiently explore the solution space and find the desired solutions. Local search functions guide the exploration of the solution space by focusing on promising regions. A good local search function helps the algorithm navigate through the solution space more efficiently, reducing the computational effort required to find solutions.

Based on that and after evaluating several local search strategies reported in [69] we empirically noticed that the local search strategy introduced by [79] showed good performance in optimizing the parameters of the network. Thus, in every cycle, values of all parameters will be adjusted as mentioned in Equations 3.6 and 3.7 which was introduced by [79] a local search method initiated from the existing solution, which involves navigating the surrounding reagon:

$$r = 4a \ cos\,\phi_0 \ sin\,\phi_0 \tag{3.6}$$

$$W_{new} = W_{actual} \pm [r\,sin(\phi_k) + r cos\,(\phi_n)] \tag{3.7}$$

Here $a \in \{0, 0.3\}$ represents a randomly determined number that governs the diference over which an object can perceive its surroundings, while $\phi \in \left\{0, \frac{\pi}{2}\right\}$ the angle of moving direction. k and n are the angular values selected randomly for each point withing the range $\in \{0, 2\pi\}$.

In the Polar Bear Optimization (PBO) algorithm [79], local search is performed by adjusting the coordinates of the bear's position. The algorithm uses local search strategies to explore the solution space effectively.

- Adjusting Angle (ϕ): The angle component of the coordinates represents the direction in which the search agent is moving. During local search, the

algorithm may make small adjustments to this angle, allowing the agent to explore the nearby regions in the solution space.

- Adjusting Distance (r): The distance component of the coordinates represents the distance from the origin. Local search may involve small variations in this distance, enabling the individual solutions to explore other solutions that are close to its current position.

By adjusting both the angle and distance, solutions can navigate the solution space more finely, exploring local regions around its current position. These local search operations aim to refining the solutions in the population and potentially converging to a better optimum.

Following the completion of each cycle, during which every object has taken a step, or attempted to do so, towards the target, we enact a dynamic population strategy inspired by the methodology described in [79]. Subsequent to each cycle, a comprehensive evaluation of the entire population occurs. According to a random parameter generated at each cycle, a decision regarding if to remove an object from the population or introduce a new one is made. In instances where the decision leans towards removal, the poorest-performing solution within the population is selectively eliminated. This approach aims to mitigate the exploration of futile regions within the search domain, thus enhancing the efficiency of the optimization process. Otherwise, when creating a new object, we select the best object and we clone it. We assume here that the best object is searching in a promising area and having more objects searching in that area will result in faster convergence. Figure 3.2 illustrates a depiction of solutions represented by white dots within the search space, with the objective being to locate the optimal solution represented by the dark blue point. According to our dynamic strategy, the object situated on the most far left should be discarded, given its considerable distance from the target. Conversely, the object positioned near the best solution should be replicated. The rationale behind the dynamic approach stems from grappling with a high-dimensional search space, where achieving comprehensive coverage with objects is exceedingly challenging. Consequently, this strategy facilitates the exploration of a broader spectrum of areas within the search space, leveraging a smaller number of search agents. Additional elaboration on this strategy will be thoroughly discussed in the experimental analysis chapter, providing comprehensive insights and details.

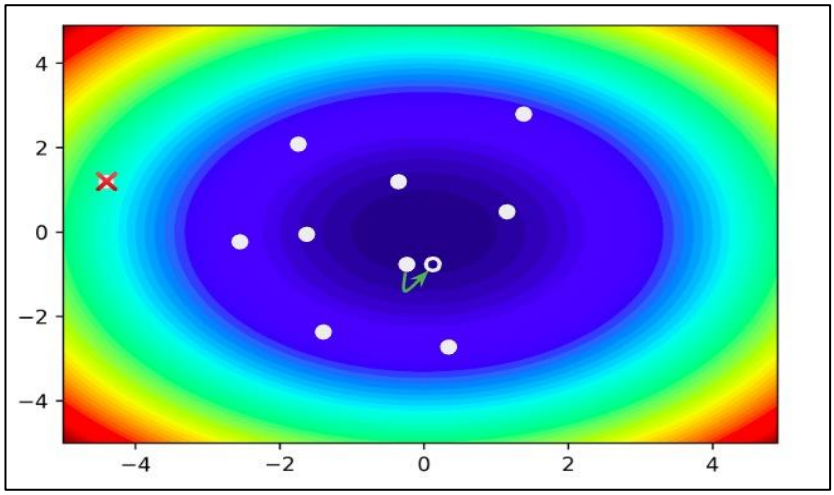


Figure 3.2. Dynamic population strategy.

To prevent getting stuck in local optima, we won't duplicate the object with higher accuacy and remove the lowest one at each cycle. But, we introduce a new variable $k \in \{0, 1\}$ generated randomly at the end if every loop. When the generated random variable's value is below 0.75, then we will be reproduce a new solution; otherwise, worst solution will be dismissed.

$$Reproduction \quad if\ 0 \leq k \leq 0.75 \tag{3.8-a}$$

$$Remove \quad if\ 0.75 < k \leq 1 \tag{3.8-b}$$

In simpler terms, the strategy of being three times more likely to duplicate than to remove also offers members that are not performing well the opportunity to find better solutions, as they are not immediately removed at each cycle. In essence, we can approximate that even the poorest solution will likely have three opportunities to improve itself before being removed.

When the initialization phase concludes, we choose the best solution, typically the one with the maximum accuracy, to serve as the initialization point for the next training phase, which will utilize classic backprobagation methods. The accuracy is calculated according to Equation 3.4. However, given the very high number of dimensions of search domain. Here we do not anticipate the heuristic algorithm to reach the optimal values for the weight vector by itself, unlike various studies aiming to entirely substitute gradient-based optimizers with heuristic approaches. Instead, our proposition is that the heuristic algorithm swiftly navigates the search space, offering

a robust initial position for the gradient-based optimizer to proceed with the rest of the neural network training. Additionally, we expect, supported by our upcoming experiments, that the dynamic population strategy will enable the exploration of an extensive search space with a restricted number of individuals.

In addition to specifying the model for optimization, the algorithm needs three key input parameters: population size, cycle count (epochs), and heuristic ratio. The heuristic ratio governs the allocation of training cycles among heuristic and gradient-based techniques. Employing a larger number of objects enhances the likelihood of discovering optimal values for the weight vector but may potentially lead to decreased performance. After initialization, the algorithm proceeds through a specified number of iterations for each potential solution. The number of iterations constitutes another critical input parameter that warrants careful selection. Increasing the number of iterations boosts the probability of converging closer to the optimal solution through local search. However, this enhancement comes with the trade-off of prolonged runtime for the algorithm. In every cycle, every object explores by taking steps while considering both the positive and negative sides, represented by positive and negative values respectively in Equation 3.7. Subsequently, both generated potential solutions are evaluated, and will move only if a superior solution is generated.Our dynamic population optimization approach is summarized by Algorithm-1 and Figure 3.4.

```
Algorithm 1 Dynamic Population Optimization – DPO
Require: Model: Neural Network Model to   Optimize. S, N:
Population Size and Max Iterations. h: heuristic ratio
1: randomly initialize S objects
2: calculate heuristic cycles H according to (3.9)
3: while Loop Counter < H do
4:   for all Population Members do
5:   Generate new solutions according to Eq. (3.6 & 3.7)
6:   Evaluate new solution according to Eq. (3.4) and move
     if better
7:   Generate random value ϕ
8:   If ϕ < 0.75
9:        Duplicate the best solution
10:  else
11:       Remove Worst Solution
12:  select best model M
13:  calculate gradient cycles G according to (3.10)
14: End While
15: continue the training using gradient descent with G
cycles starting with initial weights M
```

Figure 3.3. Dynamic population optimization – DPO.

Following the completion of a iteration when individuals of the population have the opportunity to improve their current values of weight vector, the dynamic population strategy comes into play. As outlined in lines 7-11 of Algorithm 1, a decision is made based on a randomly generated variable regarding either to duplicate or eliminate an individual from the population. If the value of the randomly generated variable is below 0.75, reproduction is chosen; else, removal occurs. In this context, the value 0.75 indicates that the re-production possibility is almost three times than the removal decision. While this ratio remains fixed in this study, it could optionally be set as an input parameter. When the decision is reproduction, we select the best solution and we create an identical one, this allows more intensive search in that area. In other words, we suppose that the optimal solution -so far- is close to the optimal solution we are looking for and that area is worth more agents to search for. However, when the choice is to remove, we select the worst solution in the solution domain, and we dismiss assuming that the solution is searching in a wrong region and to continue searching in that area is a waste of time.

The final parameter is the Heuristic Ratio, responsible for allocating number of iterations among the heuristic phase H and the gradient phase approach G.

$$H = \frac{h_{ratio}}{100} * N \tag{3.9}$$

$$G = N - H \tag{3.10}$$

In simpler terms, this parameter determines the proportion of time dedicated to initialization. Setting this parameter to zero implies that no initialization will be performed, and the optimization process will rely solely on gradient descent. Conversely, setting it to 100 indicates that the entire training process will be conducted using heuristic methods without traditional gradient descent. Intermediate values represent a combination of both approaches, with a portion of the training time allocated to initialization and the remaining time devoted to gradient descent. Adjusting this parameter allows for fine-tuning the tradeoff between exploration (through heuristic initialization) and exploitation (through gradient descent) during the training process.

Gradient-based techniques iteratively update the weights of a neural network according to the computed loss, aiming to reduce this loss over successive iterations. By iteratively modifying the parameters in the direction that reduces the error, these techniques gradually converge towards optimal solutions. The error represents the disparity among the calculated value and the netowrk actual output. This difference quantifies how well the model's predictions align with the ground truth or target values. To compute this error, various cost functions, or loss functions, can be employed. One commonly used cost function is the mean squared error (MSE), which calculates the average squared difference between the predicted and actual values across all samples. Mathematically, the MSE cost function is defined as:

$$c = \frac{1}{n}\sum_{p}^{n}\sum_{k}^{m}\left(\widehat{y_{pk}} - y_{pk}\right)^2 \tag{3.11}$$

Once the loss is computed, the parameners will be adjusted by calcualtuing the gradient for all weights in the network. In this context, n is the total number of training data points, while m signifies the count of output parameters. The derivation of the loss with respect to each node in the model is calcuated to determine the direction and magnitude of weight adjustments during the optimization process.

$$\Delta_\theta = -\eta \frac{dC}{d\theta}. \tag{3.12}$$

Here θ is the weight vector and η denotes the learning rate.

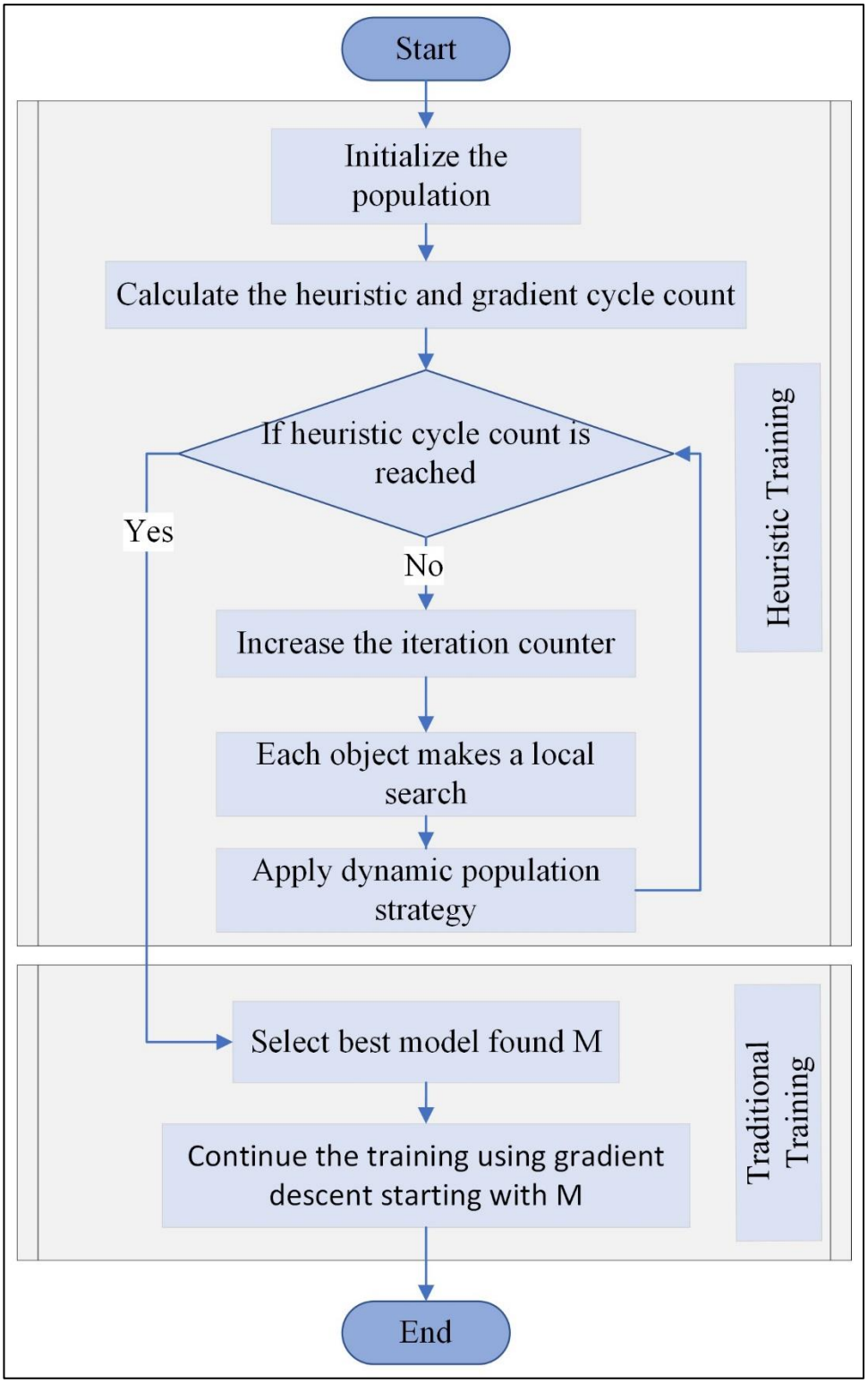


Figure 3.4. Procedure of DPO.

3.3. Meta Learning Perspective

Optimizers reliant on gradients, like stochastic gradient descent (SGD) or adaptations such as Adam, are designed to update the model's parameters in a way that minimizes

a given loss function for a specific task. However, in the realm of meta-learning, since the goal is to let a network learn how to learn across a wide range of tasks, standard optimizers may not be sufficient, and custom optimizers or meta-optimization techniques become essential.

In meta-learning, the model is trained on a distribution of tasks, and each task might require a different optimization strategy. Standard optimizers have fixed hyperparameters and update rules that may not adapt well to the varying requirements of different tasks. Custom optimizers can be tailored to the specific needs of each task, allowing for faster adaptation.

Neural networks used to train meta-learning models are usually high dimensional networks, gradient based techniques might need a lot of gradient steps to reach the minima or might even stuck in local minima [2]. We propose a hybrid model of meta-heuristic algorithm consists of dynamic population heursitic approac followed by gradient descent to be used as an optimization function while training the meta-learning models. In this research we have applied our proposed optimization function, Algorithm-1, using benchmark datasets Omniglot and MiniImage which can be considered as the most popular datasets for meta-learning problems due to high number of classes with limited number of samples for each.

Meta-learning, a paradigm where models learn to rapidly adapt to new tasks with minimal data, has witnessed significant advancements with the emergence of recently developed algorithms. Among these, Model-Agnostic Meta-Learning (MAML), Reptile, and Meta-SGD have established themselves as powerful platforms across diverse domains. MAML and Reptile excel in learning versatile initializations for quick adaptation, Meta-SGD dynamically adjusts learning rates for fine-tuning. Their collective contributions have moved the field of meta learning forward, demonstrating efficacy and versatility in addressing a wide range of meta-learning challenges across several real world problems.

Table 3.1. Meta-learning algorithms comparison.

Aspect	MAML	Reptile	Meta-SGD
Key Idea	Learns model initialization for fast adaptation	Finds a good initialization for fast adaptation	Adapts learning rates for task-specific fine-tuning
Training Procedure	Meta-optimization over a distribution of tasks	Closest-to-initialization optimization	Meta-optimization of learning rates across tasks
Flexibility	Model-agnostic, applicable to various neural network architectures	Model-agnostic, emphasizes a straightforward update rule	Adapts learning rates, providing flexibility to existing models
Main Emphasis	Initialization of model parameters for efficient task adaptation	Iterative updates that bring the model closer to task initialization	Dynamic adaptation of learning rates for optimal fine-tuning
Training Scenario	Meta-optimization over a distribution of diverse tasks	Closest-to-initialization updates for rapid task adaptation	Optimization of learning rate dynamics across a variety of tasks

3.3.1. DPO – MAML adaptation

Model-Agnostic Meta-Learning (MAML) can be considered as a cornerstone in the field of meta-learning, playing a pivotal role in shaping the landscape of adaptive machine learning algorithms. At its core, MAML optimizes a model's parameters in a way which will result in a quickly adapt to a diverse array of tasks, making it agnostic to the specifics of the underlying neural network architecture. This model-agnostic approach, not tied to any specific model, has sparked numerous expansions and modifications in diverse fields, spanning from computer vision to voice recognition and natural language processing. In essence, MAML's influence has transcended its initial introduction, producing a flow in research efforts dedicated to refining and extending meta-learning methodologies. The importance of MAML in meta-learning resides in its capacity to facilitate quick adaptation to new tasks through a gradient-based optimization process. MAML allows a model to quickly adapt to new tasks with only a few examples. It achieves this by training the model's parameters in a way that they are more conducive to fast adaptation during the fine-tuning phase on new tasks.

MAML can be used to solve regression, supervised learning and reinforcement learning problems, however, in the below section we will only discuss the supervised learning part and how we adopt our optimizer into MAML:

- **Initialization:** The model is initialized with random parameters. A meta-batch of tasks is sampled, each consisting of a support set and a query set.
- **Inner Loop (Adaptation):** The model is trained on the support set of each task for a few iterations to adapt its parameters. The calcualted error on the support set is used to adjust the network's parameters through backpropagation.
- **Outer Loop (Meta-Training):** The model's parameters are updated based on the aggregated performance on the query sets of all tasks. This involves computing the meta-gradient, which represents the overall performance across tasks, and updating the model's parameters to improve its generalization.

$$\theta_{0_i} = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta) \tag{3.13}$$

- **Meta-Testing (Adaptation to New Tasks):** During meta-testing, the model is fine-tuned on new tasks using a few examples from the support set, demonstrating its ability to quickly adapt to unseen tasks.

Algorithm 2 MAML + DPO
Require: p(T): distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $T_i \sim p(T)$
  for all T_i **do**
    Evaluate $\nabla_\theta LT_i(f_\theta)$ with respect to K examples
    ~~Compute adapted parameters with gradient descent: $\theta_i^{`} \leftarrow \theta - \alpha \nabla_\theta LT_i(f_\theta)$~~
    Model Optimization with DPO
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum T_i \sim p(T) LT_i(\theta_i^{`})$
end while

Figure 3.5. MAML + DPO.

3.3.2. DPO – Reptile adaptation

Reptile is designed to improve the model's ability to generalize across tasks by repeatedly exposing it to different tasks and updating its parameters in a specific manner. The training procedure can be summarized as follows:

- **Initialization:** The model starts with random parameters. In Reptile, there's no explicit separation between support and query sets during meta-training.
- **Inner Loop (Adaptation):** The model is trained on a task for a fixed number of iterations. Unlike MAML, Reptile focuses on updating the network's weights towards the end of the inner loop, emphasizing faster adaptation.
- **Outer Loop (Meta-Training):** The network's weights are adjusted based on the accumulated changes made within the adaptation phase across multiple tasks. Reptile aims to find a set of parameters that allows the model to quickly adapt to various tasks.

$$\phi \leftarrow \phi + \frac{\varepsilon}{n}\sum_{i=1}^{n}(\phi e_i - \phi) \tag{3.14}$$

- **Meta-Testing (Adaptation to New Tasks):** During meta-testing, the model is fine-tuned on new tasks using a few examples, showcasing its ability to rapidly adapt to novel tasks based on the learned meta-knowledge.

After iterating through multiple tasks, the model is expected to have learned a more generalized set of parameters that enable it to quickly adapt to new tasks.

Algorithm 3 Reptile + DPO
initialize θ, the vector of initial parameters
for iteration = 1, 2, … **do**
    Sample task T, corresponding to loss L_T on weight vector $\tilde{\theta}$
    ~~Compute $\tilde{\theta} = U_T^k(\theta)$, denoting k steps of SGD or Adam~~
    Model Optimization with DPO
    Update $\theta \leftarrow \theta + \epsilon(\tilde{\theta} - \theta)$
end for

Figure 3.6. Reptile + DPO.

The key idea behind Reptile is that by iteratively exposing the model to different tasks and updating its parameters based on the observed task-specific losses, the model becomes more capable of rapid adaptation to unseen tasks with limited number of training samples. This meta-learning approach helps enhance the network's capacity to generalize across a wide range of tasks.

3.3.3. DPO – Meta-SGD adaptation

Meta-SGD focuses on adapting the learning algorithm itself, specifically the update rule or optimization strategy, rather than just the model parameters. The goal is to train a meta-learner that can quickly adapt to new tasks by learning an effective update rule during meta-training.

Here's a brief overview of the Meta-SGD algorithm:

- **Initialization:** The model begins with random parameters. Similar to Reptile, Meta-SGD doesn't explicitly distinguish between support and query sets during meta-training.
- **Inner Loop (Adaptation):** The model is trained on a task for a fixed number of iterations. During this adaptation phase, the model's parameters are updated to better fit the specific task using standard gradient descent.
- **Outer Loop (Meta-Training):** Meta-SGD introduces a meta-optimizer, which is itself optimized during the outer loop. The meta-optimizer adapts the learning rate or other parameters of the base optimizer (e.g., SGD) enhancing the model's capacity for rapid adaptation to unseen tasks during the training phase.

$$\theta_0 = \theta - \alpha \circ \nabla L_T(\theta) \tag{3.15}$$

- **Meta-Testing (Adaptation to New Tasks):** In meta-testing, the meta-trained optimizer is used to fine-tune the model on new tasks. The goal is to demonstrate the effectiveness of the learned meta-optimizer in facilitating fast adaptation across a variety of tasks.

The idea behind Meta-SGD is to learn a set of meta-parameters that guide the optimization process, making the algorithm more adaptive and capable of efficient learning on a wide range of tasks.

Algorithm 4 Meta-SGD + DPO

Input: task distribution $p(T)$, learning rate β

Output: θ, α

Initialize θ, α;

while not done do

  sample batch of tasks $T_i \sim p(T)$

  for all T_i do

    $L_{train(T_i)}(\theta) \leftarrow \frac{1}{|train(T_i)|} \sum_{(x,y) \in train(T_i)} \ell(f_\theta(x), y)$;

    ~~$\theta_i' \leftarrow \theta - \alpha o \nabla L_{train(T_i)}(\theta)$;~~

    Model Optimization with DPO

    $L_{test(T_i)}(\theta_i') \leftarrow \frac{1}{|test(T_i)|} \sum_{(x,y) \in test(T_i)} \ell\left(f_{\theta_i'}(x), y\right)$;

  End

  $(\theta, \alpha) \leftarrow (\theta, \alpha) - \beta \nabla_{(\theta,\alpha)} \sum_{T_i} L_{test(T_i)}(\theta_i')$

end

Figure 3.7. Meta-SGD + DPO.

By explicitly considering the learning algorithm itself as a parameterized entity, Meta-SGD contributes to the field of meta-learning, enabling models to learn not only what to learn but also how to learn.

4. EXPERIMENTAL ANALYSIS

Within this chapter, our evaluation of the Dynamic Population Optimization (DPO) algorithm will be divided into two distinct sections. The first section will explore experiments conducted on benchmark classification datasets, providing insights into the algorithm's performance with traditional deep learning classification. Here, we assess its efficiency in handling traditional classification tasks -not meta learning- and establish a baseline for comparison.

The second section of the chapter will go into detailed exploration of the DPO algorithm optimizing of meta learning problems. This section aims to evaluate the algorithm's behavior and adaptability in the face of meta-learning scenarios, where the complexity extends beyond conventional classification tasks.

4.1. Experimental Analysis on Classification Datasets

Evaluating our proposed approach, we've chosen five benchmark datasets detailed in Table 4.1. Each dataset includes information on the total layers count in addition to the count of model’s parameters to be adjusted. The count of those parameters holds significant importance in terms of performance, serving as a metric for the size of the solution space, as it signifies the size of the weight vector and eventually number of dimensions. This parameter greatly influences the complexity and search space of the optimization problem, thus affecting the algorithm's efficiency and effectiveness in finding optimal solutions. Details of ANN model used for each dataset are listed in Table 4.1., However, our algorithm, as detailed in Algorithm-1, is not confined to any particular architecture. It's designed to be versatile and adaptable, capable of being applied across various neural network architectures. This flexibility enables its utilization in a diverse range of scenarios, accommodating different model complexities and requirements. By being architecture-agnostic, our approach can address a wide array of optimization challenges encountered in neural network training, providing a scalable and robust solution.

The algorithm accepts the network that should be optimized as an input parameter without any restriction or pre-assumption on the structure of the model such as the number of layers nor number of trainable parameters Figure 4.1. presents an overview of the model structure for Cifar100. All datasets are divided into two sets training and test, with the count of data points used during testing specified in Table 4.1. The chosen train/test rates are commonly accepted by the deep learning community in several implementations and have been employed in a lot of studies. These ratios were deemed logical as they achieve a balance among having a sizable dataset for training and ensuring the test dataset remains representative. This ensures that the model is adequately trained on diverse data while also being rigorously evaluated on unseen samples, facilitating robust performance assessment.

To ensure fair comparison among outcomes, it's essential to address potential influences from several random variables, such as the starting weight settings and the production of random values during solution creation. To mitigate these effects, we've adopted consistent practices across all experiments on the same model. Specifically, we've utilized identical startup weights for all tests considering the same network and maintained uniformity by using the same seed when generating random values. This approach helps to minimize variability stemming from random factors, thereby enhancing the reliability and validity of our comparisons.

In our experimental analysis, we conducted training processes for all models by recording the obtained accuracy at epochs (20, 30, and 40). The Heuristic Ratio parameter allows for adjustment between traditional gradient descent, heuristic approaches, or a hybrid one. In our evaluation, we considered the values of 0, 20, 50, 70, and 100 for this parameter. A setting of 0 implies no heuristic initialization, relying solely on gradient optimization, while 100 indicates exclusive use of the heuristic approach for to optimize the model. Intermediate rates at (20, 50, and 70) denote a hybrid model, starting the optimization by heuristic approach with subsequent gradient descent training. For example, when running 400 epochs with a heuristic ratio of 20, this signifies that 80 cycles are allocated to the heuristic algorithm for initialization, with the remaining 320 cycles utilized by traditional training. We utilized the Adam optimizer [30] to represent the backpropagation-based training, as it has demonstrated superior performance compared to similar gradient-based optimizers [3, 30]. However, in our subsequent analysis and discussions, we will refer to SGD as the base of

gradient-based optimizers, recognizing that any other gradient-based algorithm could be used instead of Adam in our approach.

Table 4.1. List of classification datasets.

Dataset	Layers #	Trainable Parameters	Attributes #	Instance #	Test#	Output
FASTION	7	1,163,330	784	70,000	10,000	10
IRIS	3	55	5	150	30	3
CIFAR-10	7	1,632,080	1,024	60,000	10,000	10
MNIST	7	1,256,080	784	60,000	10,000	10
CIFAR-100	8	3,042,546	1,024	60,000	10,000	100

From Table 4.2. and graphs in Figure 4.2, it's evident from our findings that our algorithm excelled consistently among all of the models and nearly all stages of training cycles when the heuristic rate was set to 20. However, when this rate was increased to 50 and more, this count not yield improved results. When drawing a comparison between the outcomes obtained from heuristic ratios of 0 and 20 indicates that swiftly navigating the search space and subsequently proceeding with gradient-based technique indeed produces best outcomes. This suggests that a balanced approach, combining exploration and exploitation, is crucial for achieving superior performance in neural network training.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 30, 30, 32)        896

max_pooling2d_4 (MaxPoolin   (None, 15, 15, 32)        0
g2D)

conv2d_5 (Conv2D)            (None, 15, 15, 50)        40050

max_pooling2d_5 (MaxPoolin   (None, 7, 7, 50)          0
g2D)

flatten_2 (Flatten)          (None, 2450)              0

dense_6 (Dense)              (None, 1000)              2451000

dense_7 (Dense)              (None, 500)               500500

dense_8 (Dense)              (None, 100)               50100

=================================================================
Total params: 3042546 (11.61 MB)
Trainable params: 3042546 (11.61 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.1. Summary of Cifar100 model.

Because regular methods that use Stochastic Gradient Descent (SGD) initiates the search processes from a random point, our experiments showed that spending a few rounds to find a good starting point gives us enhanced outcomes. This highlights the importance of taking some time in the beginning to get higher accuracy at the end.

Table 4.2. Accuracy per heuristic ratio.

		Heuristic Ratio				
Epoch	Dataset	0	20	50	70	100
20	FASHION	0.8583	**0.8941**	0.8796	0.7589	0.4045
30	FASHION	0.9144	**0.9370**	0.8855	0.7740	0.4237
40	FASHION	0.9255	**0.9317**	0.9139	0.8514	0.4745
20	IRIS	0.6187	0.6203	0.6537	0.6474	**0.7213**
30	IRIS	0.7378	0.7207	0.7348	0.7512	**0.7691**
40	IRIS	0.8237	0.8486	0.8570	0.8602	**0.8680**
20	CIFAR10	0.7145	**0.7594**	0.7479	0.6605	0.4139
30	CIFAR10	0.8073	**0.8482**	0.7484	0.6943	0.5186
40	CIFAR10	0.8598	**0.8977**	0.8847	0.8377	0.6498
20	MNIST	0.9530	**0.9744**	0.9673	0.8934	0.8064
30	MNIST	0.9677	**0.9843**	0.9797	0.9147	0.8605
40	MNIST	0.9942	**0.9959**	**0.9953**	0.9533	0.8944
20	CIFAR100	0.6181	**0.6546**	0.5427	0.4946	0.4536
30	CIFAR100	0.7375	**0.7772**	0.6980	0.6404	0.5823
40	CIFAR100	0.9243	**0.9536**	0.8155	0.8685	0.6282

Based on our experiments, it's evident that our algorithm effectively initializes the training process of a neural network. Allocating approximately 20% from the total training cycles prior to commencing gradient training leads to enhanced accuracy and faster convergence. This conclusion underscores the efficacy of our approach in optimizing the training process and improving overall performance.

Gradient-based optimizers leverage derivatives of the loss function to determine the direction of movement, making them potent tools for local search. However, they can encounter challenges, particularly when starting from suboptimal solutions, potentially converging to local optima. Despite advancements in stochastic gradient descent (SGD) algorithms that mitigate this issue [4], achieving high accuracy still often necessitates lengthy training cycles. Moreover, employing a heuristic ratio of 100 yielded unsatisfactory outcomes across most datasets, except for Iris. This discrepancy may stem from the complexity of the models utilized, which comprise a substantial number of parameters to optimize. Consequently, the sheer volume of combinations becomes insurmountable for stochastic methods to traverse effectively.

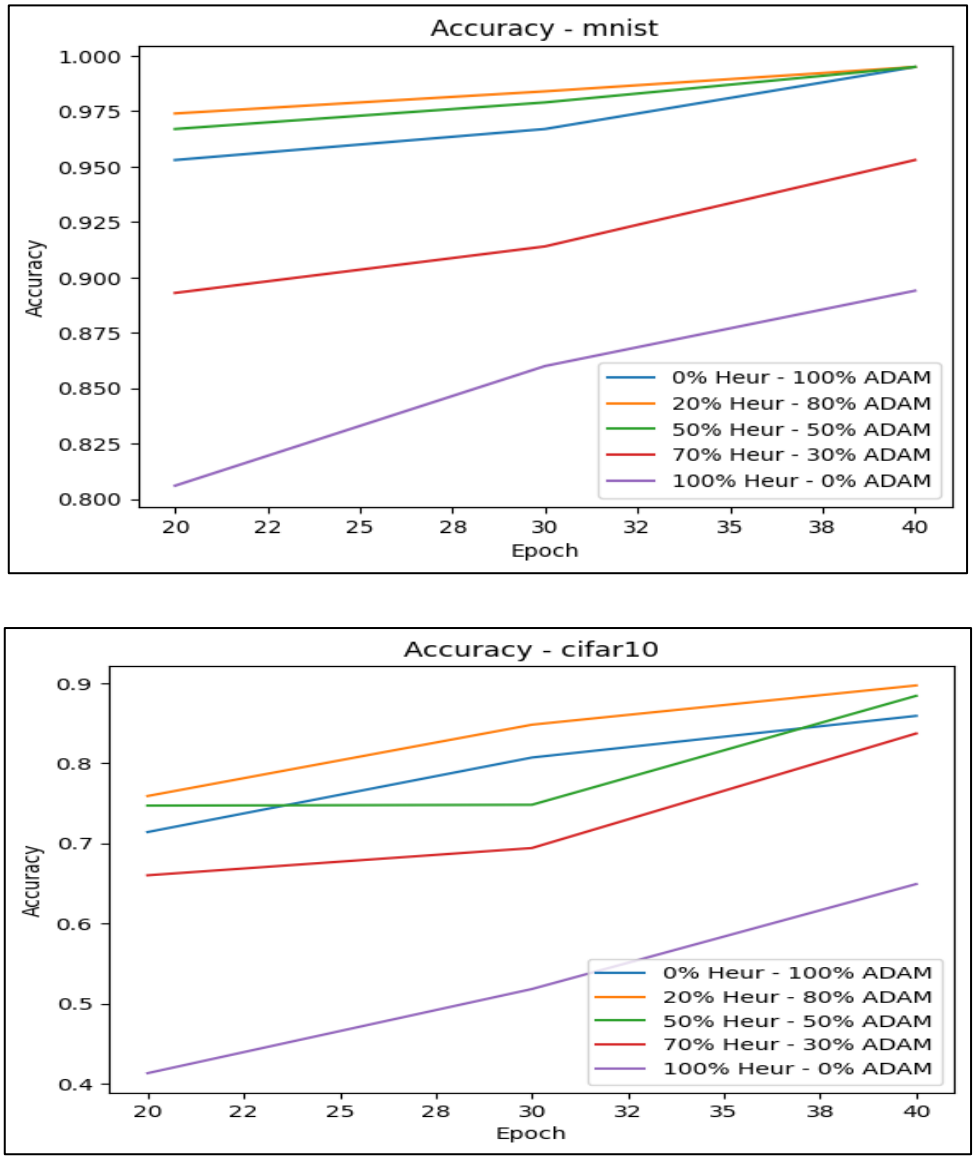
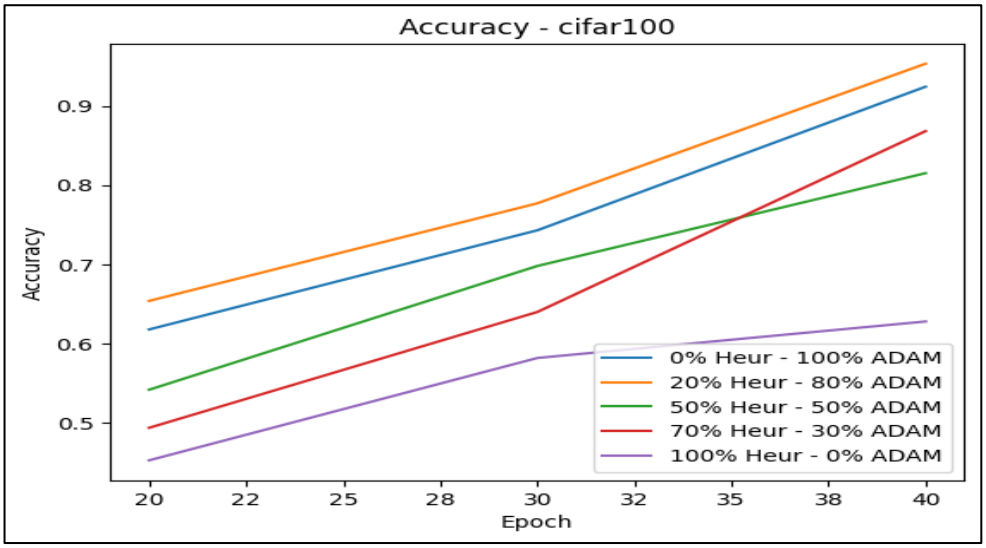
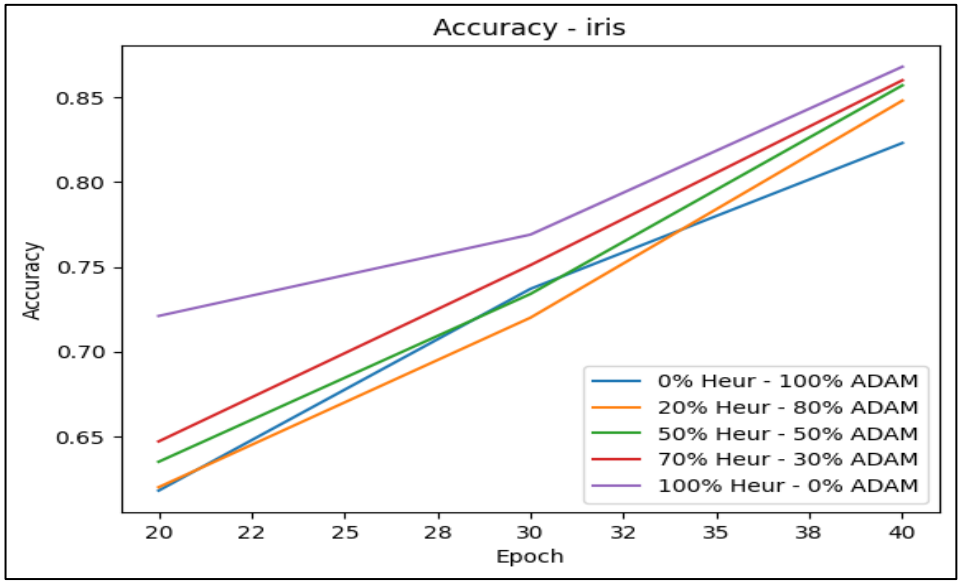


Figure 4.2. Accuracy per dataset according to heuristic ratio.





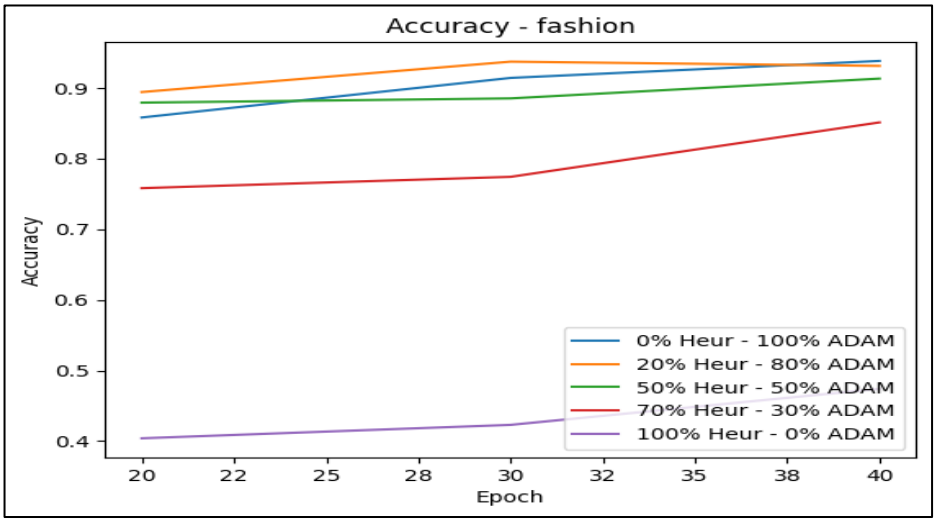


Figure 4.2. (Continued) Accuracy per dataset according to heuristic ratio.

As per [90], the time complexity of model training depends primarily on the total count of training cycles, number of training samples, and number of trainable parameters. Since our tests listed in Table 4.2 are conducted on the same model and dataset, achieving identical performance by running the exact number of training iterations using our hybrid approach would indicate greater efficiency with respect to time complexity. This is because the count of trainable parameters and training samples can be assumed as a fixed constant across all experiments.

Although the evaluations were conducted with a population size set to 20, however, the parameter is crucial and should be selected judiciously as it impacts both runtime and accuracy. The selection of population size should align with the total number of weights in the network. While rising the size of the weight vector will eventually expand the search space, necessitating more agents to explore effectively. Further detailed experiments are required to validate this relationship.

Table 4.3. Obtained accuracy by population size.

	Population size				
Model	10	20	40	60	80
FASHION	0.8423	0.8428	0.9312	0.9374	**0.9438**
IRIS	0.8155	0.8415	0.8487	0.8515	**0.8537**
CIFAR10	0.7560	0.8507	0.8977	0.9026	**0.9070**
MNIST	0.8847	0.9012	0.9656	0.9850	**0.9942**
CIFAR100	0.8277	0.9036	0.9530	0.9837	**0.9941**

In this experiment it was implemented the optimal heuristic ratio 20 from prior tests and assessed the approach across various population sizes: 10, 20, 40, 60, and 80. Outcomes are summarized within Table 4.3 and illustrated in Figure 4.3. The accuracy achieved for all datasets and population size is recorded, with the heuristic rate held constant as 20.

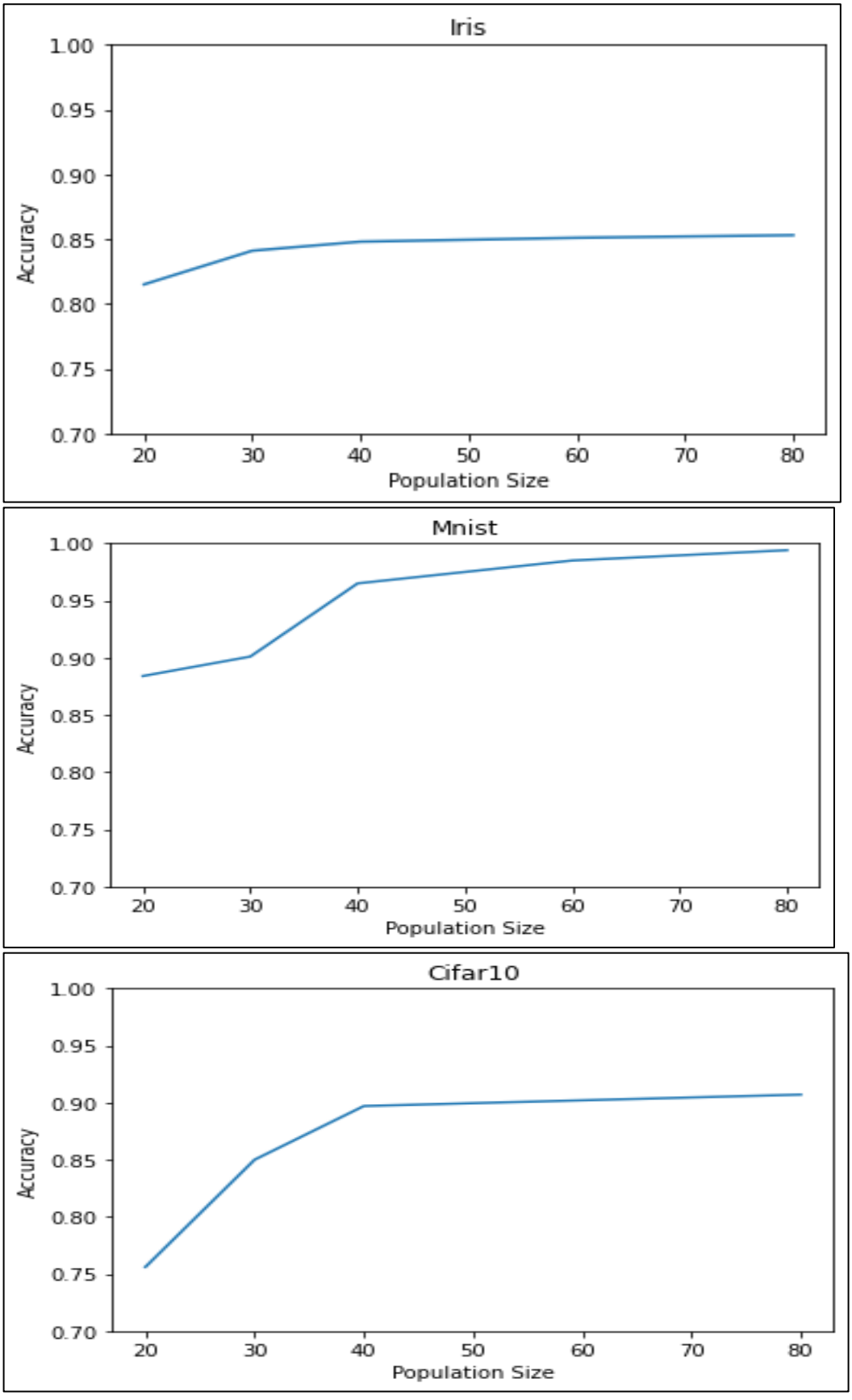


Figure 4.3. Obtained accuracy according to population size.

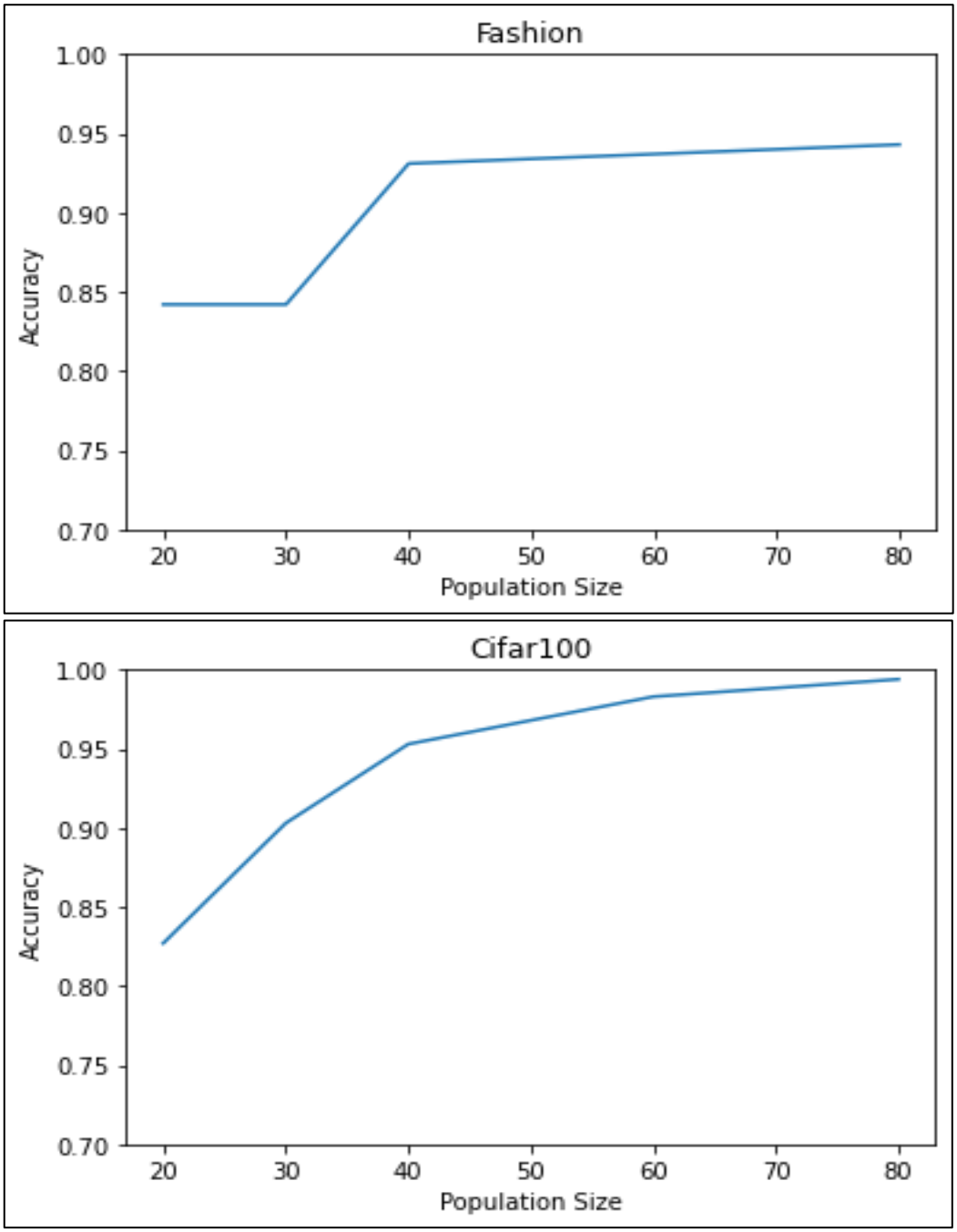


Figure 4.3. (Continued) Obtained accuracy by population size.

This approach aims to locate a better starting point within a vast search space rather than starting from random point, despite constraints on iterations and search agents. We assume that the improved performance compared with similar heuristic approaches is connected to the dynamic population strategy applied at the end of each iteration that is efficiently exploring the search domain by concentrating on promising regions and avoiding futile ones. To validate this, we conducted another experiment. We selected optimal input parameters based on prior experiments and ran the

algorithm without the dynamic population component (lines 7, 8, 9, 10 and 11 of Algorithm-1). Outcomes, as presented in Table 4.4, indicated that removing the dynamic behavior led to reduced accuracy, highlighting its crucial role in exploring the expansive search domain effectively.

Table 4.4. Effects of dynamic strategy on obtained performance.

Model	Dynamic Strategy	
	With	Without
FASHION	**0.9312**	0.9250
IRIS	**0.8484**	0.8437
CIFAR10	**0.8977**	0.8593
MNIST	**0.9954**	0.9946
CIFAR100	**0.9531**	0.9245

The outcomes closely resemble those in the initial field of Table 4.2, which the heuristic rate considered as zero. Using simpler terms, our recent experiments tell us that when dealing with complicated and big search spaces, like those found in complex models, the dynamic population strategy does a great job of finding the best starting points. On the other hand, for simpler models, the traditional heuristic search approaches work well in achieving the same result.

4.2. Experimental Analysis – Meta Learning

As we transition from traditional classification tasks to the complex domain of meta learning, the second section of this chapter explains a detailed exploration of the Dynamic Population Optimization (DPO) algorithm's ability in addressing challenges that extend beyond conventional classification boundaries. Meta learning, distinguished by its ability to enable model adaptation across a spectrum of diverse tasks, introduces a layer of complexity that demands innovative approaches.

In this section, we start through the application of the DPO algorithm to optimize meta learning problems. Meta learning address the limitations of standard classification scenarios, requiring algorithms to show a increased level of adaptability and flexibility. The challenges posed by meta learning involve not only understanding individual tasks but also efficiently leveraging acquired knowledge to excel in new and unseen tasks.

In the coming experiments, our goal is to show how the DPO algorithm can handle the difficulties of meta learning, offering insights into its performance, adaptability, and effectiveness. By evaluating its behavior under these more complicated scenarios, we aim to contribute valuable observations that extend beyond the conventional classification, focusing on the algorithm's potential to address the evolving landscape of machine learning challenges.

4.2.1. Datasets description

In our performance evaluation, we have applied our optimizer on two popular datasets within the field of meta-learning. The first dataset, Omniglot, is thoughtfully summarized in Table 4.5, encapsulating key metrics and outcomes. The second dataset, MiniImageNet, is thoroughly documented in Table 4.6, providing an in-depth analysis of the algorithm's performance across various parameters. This detailed examination ensures a complete understanding of our algorithm's adaptability and effectiveness in handling diverse datasets.

Omniglot and MiniImagenet are highly regarded datasets in meta-learning research. They offer diverse sets of classes and tasks, crucial for evaluating the adaptability of meta-learning algorithms [6]. With a small number of instances per class, these datasets mirror real-world scenarios where learning from limited data is essential. The challenges they pose in generalization make them ideal for assessing the performance of meta-learning models [8]. Moreover, their popularity has made them standard benchmarks in the field [7]. Popular meta learning models such as MAML, Reptile and Meta-SGD, have been extensively tested on these datasets. By evaluating on Omniglot and MiniImagenet, researchers can gauge the algorithms' capacity to rapidly adapt and generalize effectively. These datasets serve as foundational tools for advancing the understanding and development of meta-learning techniques.

Omniglot: This Omniglot dataset contains 1,623 different handwritten characters from 30 different scripts, Figure 4.4, covering a wide range of languages and writing systems. This diversity challenges meta-learning models to adapt quickly to new and unfamiliar characters, making it an excellent choice for few-shot and one-shot learning scenarios, which are common in meta-learning.

Table 4.5. Specification of Omniglot dataset.

Specification	Value
Dataset Type	Character Recognition
Total Characters	1,000
Characters per Writing System	20
Writing Systems	50
Total Examples	20,000
Image Size	105x105 pixels

Moreover, The Omniglot dataset is relatively small with only 50 examples per class, making it a challenging benchmark for meta-learning. Meta-learning algorithms are designed to learn from a limited amount of data, and Omniglot's small size reflects real-world scenarios where adapting to new tasks with limited examples is important.

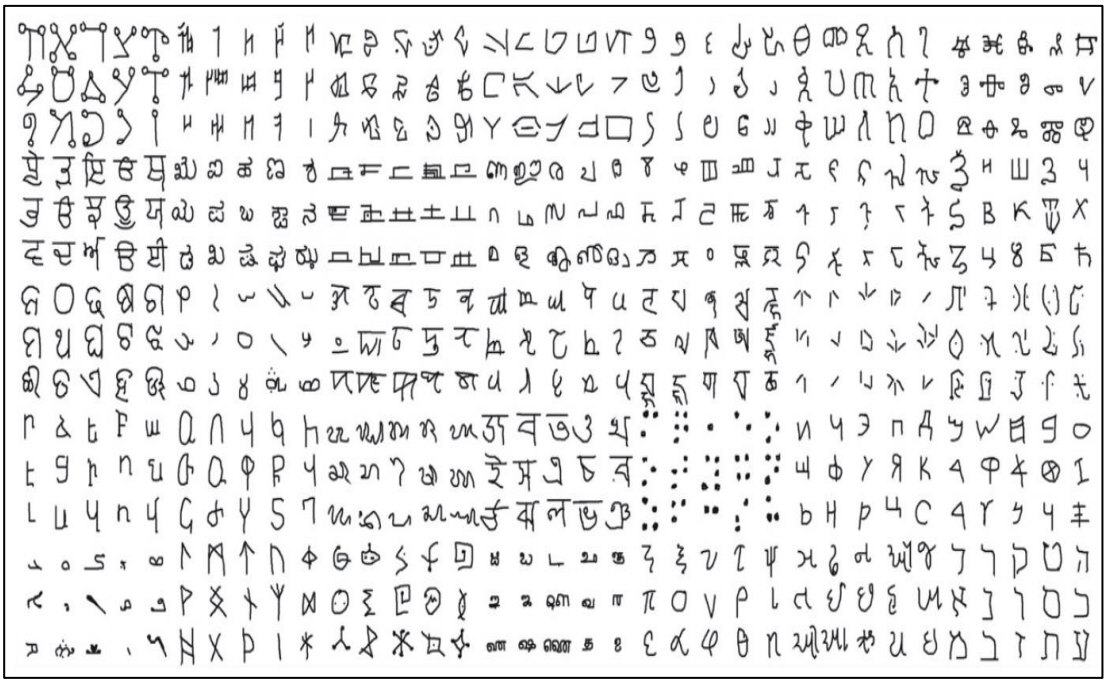


Figure 4.4. Omniglot dataset sample characters.

MiniImageNet: Serving as a subset representation of the widely-used ImageNet dataset, MiniImageNet is oriented specifically for the evaluation of meta-learning algorithms. Comprising 60,000 high-resolution color images, each measuring 84x84 pixels, Figure 4.5, the dataset spans across 100 distinct classes. These classes include a rich variety of objects, animals, and scenes, contributing to the dataset's diversity.

Table 4.6. Specification of MiniImageNet dataset.

Specification	Value
Dataset Type	Object Recognition
Total Classes	100
Images per Class	600
Total Examples	60,000
Image Size	84x84 pixels

The dataset simulates few-shot learning scenarios by providing a limited number of examples per class (600 images). This mirrors real-world situations where adapting to novel tasks with only a small set of examples is crucial. Meta-learning algorithms, designed to give insights from restricted data, are put to the test in the challenging landscape presented by MiniImageNet.

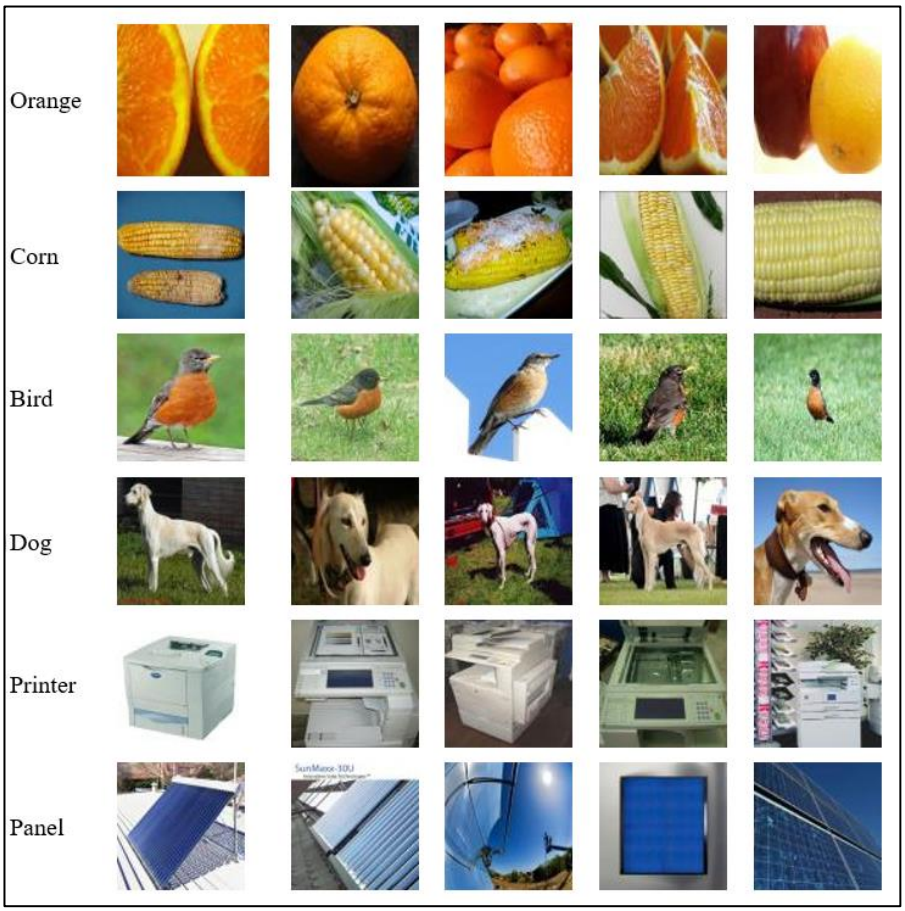


Figure 4.5. MiniImageNet dataset sample classes.

Researchers treat MiniImageNet as a benchmark to assess the robustness, adaptability, and generalization capabilities of meta-learning models. The varied and comprehensive nature of the dataset's classes ensures that models trained on

MiniImageNet are well-equipped to handle a wide array of visual recognition tasks, making it an invaluable resource in the field of meta-learning research.

4.2.2. Experimental details

As meta learning algorithms, we have selected three popular meta learning algorithms, MAML [39], Reptile [62] and Meta-SGD [57] to measure the convergence of our optimizer compared to training the same algorithms using traditional gradient based optimizers. All three mentioned algorithms provide a powerful framework for meta-learning without requiring complex architecture modifications or specialized network designs. MAML, Reptile and Meta-SGD are relatively simple algorithms to implement and conceptually intuitive.

Table 4.7. Obtained accuracy for MAML at different iterations.

Dataset	Omniglot		MiniImageNet	
Epoch	ADAM	DPO	ADAM	DPO
10	**0.1526**	0.1136	**0.0965**	0.0765
20	**0.1832**	0.1668	**0.1257**	0.1035
50	**0.3044**	0.2954	**0.1824**	0.1737
100	0.3732	**0.3865**	0.2435	**0.2565**
200	0.5532	**0.5865**	0.3317	**0.3565**
500	0.7115	**0.7421**	0.4625	**0.5045**
1000	0.8803	**0.8994**	0.5621	**0.5781**

We begin by examining the performance of the models on the Omniglot dataset, a widely used benchmark for few-shot classification tasks. The comparison is made between the traditional ADAM optimizer and our proposed DPO across different meta-learning approaches and obtained accuracy is captured at different epochs.

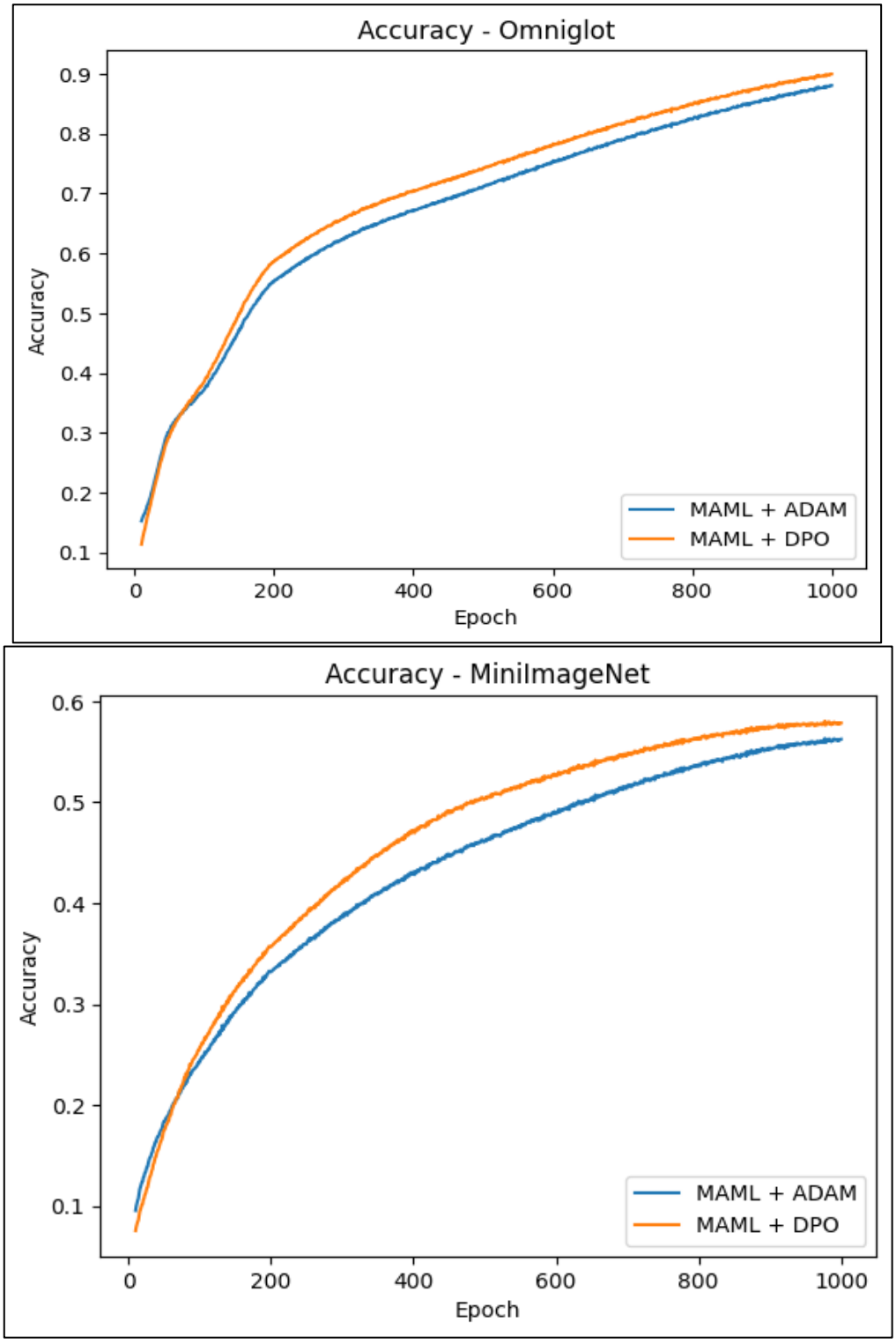


Figure 4.6. Accuracy comparison for MAML.

The training accuracy, as computed using Formula 3.4, was recorded at various training epochs. Table 4.8 presents a summary of the recorded results, first utilizing the traditional gradient-based optimizer (Adam), followed by running the same algorithm with the DPO optimizer.

Table 4.8. Obtained accuracy for Reptile at different iterations.

Dataset	Omniglot		MiniImageNet	
Epoch	ADAM	DPO	ADAM	DPO
10	**0.1354**	0.1265	**0.0832**	0.0658
20	**0.1654**	0.1458	**0.1187**	0.1178
50	**0.3114**	0.2899	**0.1792**	0.1698
100	0.3827	**0.3987**	0.2387	**0.2435**
200	0.5708	**0.5961**	0.3267	**0.3468**
500	0.7439	**0.7632**	0.4798	**0.5212**
1000	0.8852	**0.8993**	0.5705	**0.5791**

When examining the results presented in Figure 4.6, we can draw a clear and significant conclusion regarding the performance of our algorithm in our experiments. It's apparent that our algorithm is demonstrating markedly improved performance when compared to our baseline or other existing methods. This improvement is notably reflected in the accuracy metric, which quantifies the correctness of predictions made by our model.

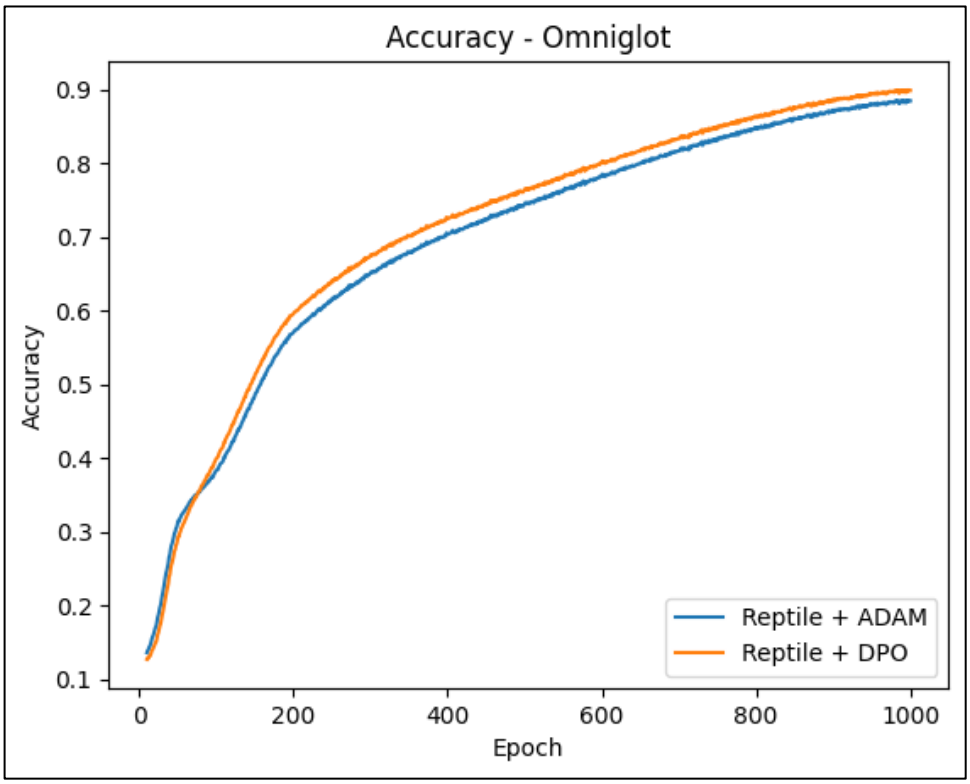


Figure 4.7. Accuracy comparison for Reptile.

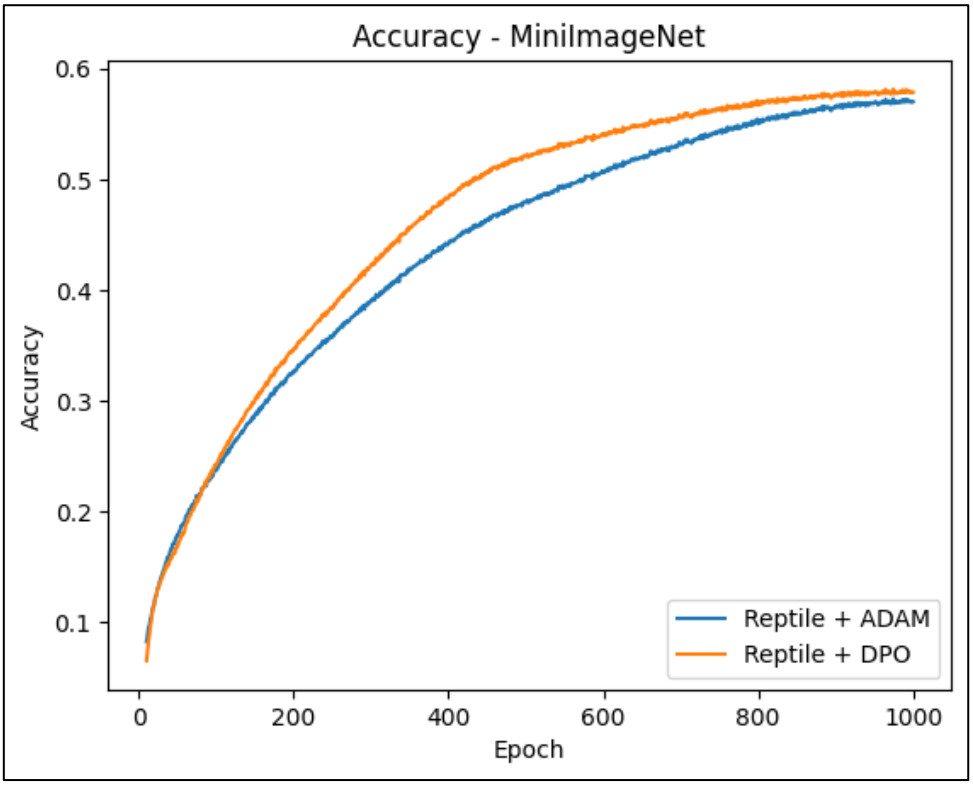


Figure 4.7. (Continued) Accuracy comparison for Reptile.

What's particularly noteworthy is that this enhanced performance is achieved without the need for additional training cycles. In both Figure 4.6, 4.7 and 4.8, we've maintained an equal number of training cycles for our algorithm and the baseline. This controlled experimental setup ensures a fair comparison between the two approaches.

Table 4.9. Obtained accuracy for Meta-SGD at different iterations.

Dataset	Omniglot		MiniImageNet	
Epoch	ADAM	DPO	ADAM	DPO
10	**0.1723**	0.1468	**0.0624**	0.0527
20	**0.1967**	0.1765	**0.0947**	0.0911
50	**0.3354**	0.3054	0.1684	**0.1724**
100	0.3967	**0.4154**	0.2209	**0.2255**
200	0.5806	**0.6032**	0.2965	**0.321**
500	0.7536	**0.7824**	0.4601	**0.4967**
1000	0.8612	**0.8844**	0.5587	**0.5751**

The fact that our algorithm consistently produces higher accuracy within the same number of training cycles highlights its efficiency and effectiveness in learning and making predictions.

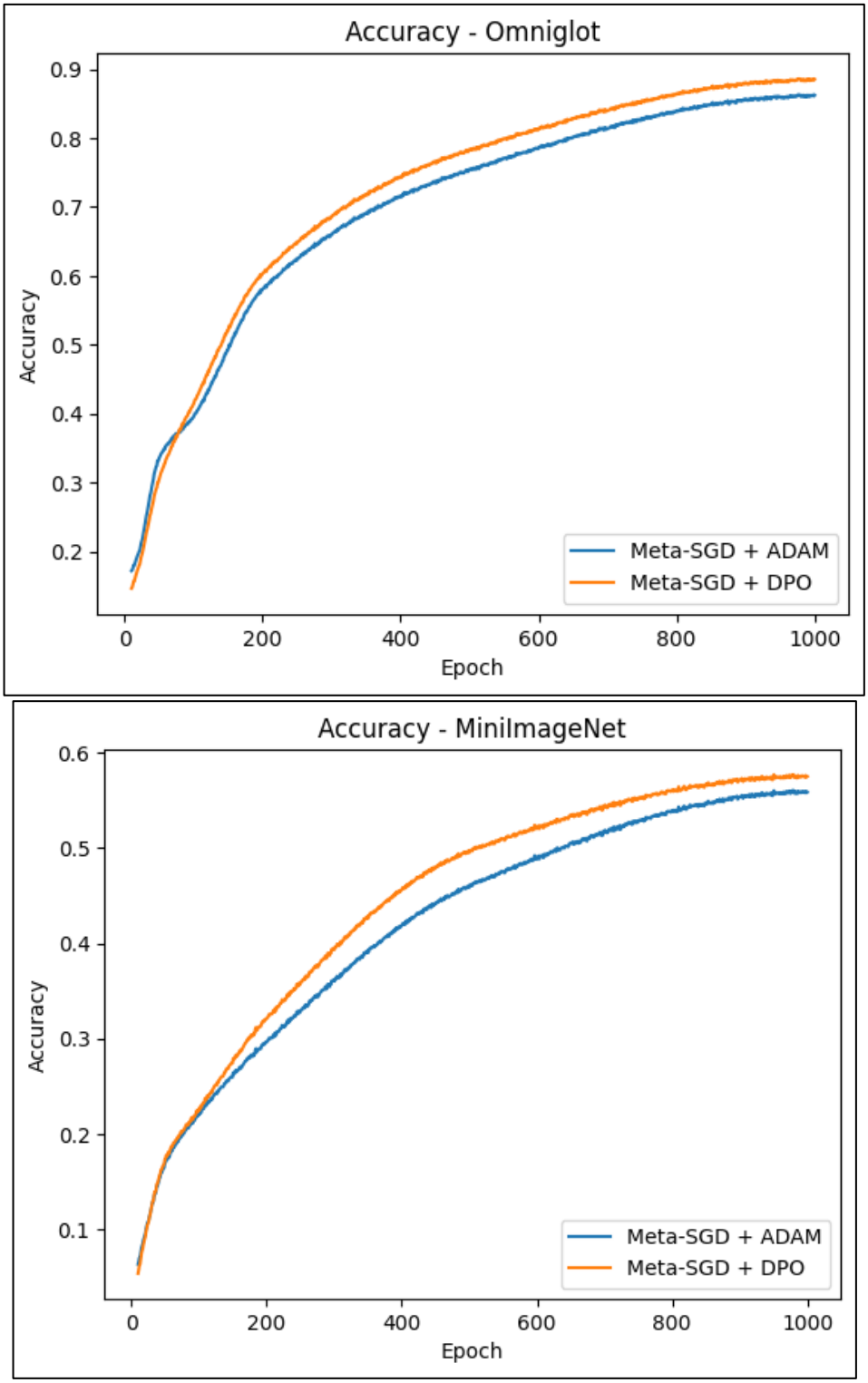


Figure 4.8. Accuracy comparison for Meta-SGD.

In simpler terms, our algorithm proves to be more adept at the given task, consistently delivering superior results compared to the baseline while expending the same training resources. This outcome underscores the value and promise of our approach in achieving better results in a resource-efficient manner.

4.3. Discussion

After reviewing the experimental results presented in Tables 4.7 to 4.9 and Figures 4.6 to 4.8, we can conclude our observations as follows:

- For MAML, at earlier epochs (e.g., 10 and 20), DPO shows a lower accuracy than ADAM. However, as the training progresses, DPO catches up and even surpasses ADAM at epoch 200 and onward, where DPO achieves an accuracy of 0.8994 compared to ADAM's 0.8803 at epoch 1000 for Omniglot and 0.5621, 0.5781 respectively for MiniImageNet.
- This suggests that while ADAM may initially lead in performance, DPO gradually converges to higher accuracies, showcasing its potential for improved long-term learning which resulted in approximately 2% improvement when applied on MAML on both datasets.
- Similar trends were observed in the Meta-SGD meta-learning approach. While ADAM starts with a higher accuracy in the initial stages, DPO consistently improves its performance and surpasses ADAM at the final iteration. At the concluding step, DPO achieves an accuracy of 0.8844, outperforming ADAM, which attains 0.8612 for Omniglot and 0.5587, 0.5751 respectively for MiniImageNet also here talking about 2% accuracy improvement by running 1000 iterations. This reversal in accuracy trends highlights the effectiveness of DPO in adapting and learning over the course of the meta-training process.
- However, for Reptile, the improvement of performance is lower than what we observed for MAML and Meta-SGD, when running on Omniglot we got 0.8852 for ADAM and 0.8993 for DPO, and 0.5705, 0.5791 for MiniImageNet, both experiments showed almost 1% accuracy improvement on Reptile compared to 2-2.5% for MAML and Meta-SGD.

To summarize, after conducting 1000 training cycles, the accuracy of both MAML and Meta-SGD saw improvements of 2-2.5% across Omniglot and MiniImageNet datasets. However, Reptile showed a 1% accuracy enhancement on both datasets and same

number of epochs. To comprehend the rationale behind these findings, it is essential to revisit Chapter 2 and delve into the intricacies of how each algorithm operates. This exploration will shed light on why employing the heuristic approach proved more effective for MAML and Meta-SGD compared to Reptile.

Both MAML (Model-Agnostic Meta-Learning) and Meta-SGD incorporate a two-step optimization process during their respective meta-training phases. This shared characteristic involves an inner loop and an outer loop. In the inner loop, the model undergoes task-specific adaptation with a limited dataset, allowing it to quickly adjust its parameters to the specifics of a given task. The outer loop then updates the model's parameters based on the aggregated experience across multiple tasks, aiming to find a set of parameters that generalize well and facilitate rapid adaptation. This dual-step optimization framework is a fundamental aspect of both methodologies, reflecting their commonality in addressing the challenges of meta-learning and few-shot learning scenarios.

However, in contrast to the two-step optimization process employed by MAML and Meta-SGD, Reptile adopts a simpler approach that emphasizes faster convergence. During training, Reptile performs only a few gradient steps on a given task within an inner loop. These steps allow the model to quickly adjust its parameters to better suit the characteristics of the specific task at hand. Importantly, instead of fine-tuning the model extensively for each individual task, Reptile accumulates the updates obtained from these short optimization steps across multiple tasks in the outer loop. This strategy promotes a more generalized adaptation, as the accumulated updates influence the model's parameters to be more broadly applicable across a range of tasks. By prioritizing simplicity and efficiency, Reptile aims to strike a balance between adaptation to specific tasks and achieving faster convergence during meta-training.

The superior performance of a heuristic-based optimizer on MAML and Meta-SGD compared to Reptile could be attributed to the distinct optimization strategies employed by each meta-learning algorithm. MAML and Meta-SGD involve a two-step optimization process with inner and outer loops, which allows for more detailed adjustments to model parameters. The heuristic approach, tailored to capture these challenges, might complement the iterative adaptation process of MAML and Meta-SGD, thereby enhancing their overall efficiency. In contrast, Reptile's simpler approach, relying on only a few gradient steps, might be less responsive to the nuanced

adjustments that a heuristic optimizer provides, resulting in relatively smaller performance gains. The effectiveness of the heuristic approach may align more closely with the optimization needs of MAML and Meta-SGD, leading to improved performance on these specific meta-learning algorithms.

5. CONCLUSION

In this study, we introduced a novel approach that combines a heuristic algorithm with dynamic population-based feature followed by backpropagation derivative-based algorithm to train a neural network. The hybrid model demonstrated strong initial performance across five benchmark datasets (Iris, MNIST, CIFAR-10, CIFAR-100, and Fashion) including a substantial number of trainable parameters. Notably, our approach yielded favorable outcomes when the heuristic ratio was set to 20%, emphasizing its effectiveness in initializing the training process. Moreover, experimental analysis confirmed that the reason behind this accuracy improvement is the dynamic nature of our heuristic algorithm. We conducted the same experiments with the dynamic population flag is On then Off, observing that enabling the dynamic population resulted in higher accuracy across all the used datasets.

While our application primarily targeted fixed neural network models, the encouraging results prompt us to consider extending this approach beyond fine-tuning weight vectors. There is potential to apply this methodology to generate complete neural network architectures. Additionally, the parallelization of the initialization process, facilitated by employing multiple agents, offers avenues for exploring efficient parallelization strategies.

In a parallel effort, our research tackled meta-learning challenges by employing the same hybrid optimizer that utilizes the heuristic algorithm -dynamic population- in addition to backpropagation optimization. The effectiveness of this approach was evaluated using two widely recognized datasets in the field of meta-learning: Omniglot and MiniImageNet. As a meta-learning platform, we seamlessly integrated our optimization function into leading meta-learning frameworks, namely MAML, Reptile, and Meta-SGD. Remarkably, even with fewer training cycles, our approach demonstrated superior accuracy, achieving 2-2.5% when applied to both MAML and Meta-SGD. In contrast, we observed an enhanced accuracy of approximately 1% for Reptile compared to relying solely on gradient-based methods.

It's worth noting that the variance in results is attributed to the implementation details of each meta-learning platform used.

Nevertheless, it is crucial to highlight that our investigation was limited to the Omniglot and MiniImageNet datasets and applied specifically to the mentioned meta-learning algorithms. To ensure the broader applicability of our proposed algorithm, further exploration across a diverse range of datasets and meta-learning algorithms is necessary. This expanded examination will serve to validate the algorithm's robustness and confirm its independence from specific datasets or neural network models. Furthermore, extending our testing to real-world problems is imperative to assess the algorithm's performance in practical scenarios and enhance its relevance beyond controlled experimental conditions.

REFERENCES

[1] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[2] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," in IEEE Access, vol. 7, pp. 53040-53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[3] Emmert-Streib F, Yang Z, Feng H, Tripathi S and Dehmer M (2020) An Introductory Review of Deep Learning for Prediction Models With Big Data. Front. Artif. Intell. 3:4. doi: 10.3389/frai.2020.00004

[4] Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN COMPUT. SCI. 2, 420 (2021). https://doi.org/10.1007/s42979-021-00815-1

[5] Chong, H.Y., Yap, H.J., Tan, S.C. et al. Advances of metaheuristic algorithms in training neural networks for industrial applications. Soft Comput 25, 11209–11233 (2021). https://doi.org/10.1007/s00500-021-05886-z

[6] Hospedales, Timothy M. et al. “Meta-Learning in Neural Networks: A Survey.” IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2020): 5149-5169.

[7] Huisman, M., van Rijn, J.N. & Plaat, A. A survey of deep meta-learning. Artif Intell Rev 54, 4483–4541 (2021). https://doi.org/10.1007/s10462-021-10004-4

[8] Vettoruzzo, A., Bouguelia, M. R., Vanschoren, J., Rögnvaldsson, T., & Santosh, K. C. (2023). Advances and Challenges in Meta-Learning: A Technical Review. arXiv preprint arXiv:2307.04722.

[9] Beheshti, Zahra & Shamsuddin, Siti Mariyam. (2013). A review of population-based meta-heuristic algorithm. International Journal of Advances in Soft Computing and Its Applications. 5. 1-35.

[10] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE.

[11] Rajwar, K., Deep, K. & Das, S. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artif Intell Rev 56, 13187–13257 (2023). https://doi.org/10.1007/s10462-023-10470-y.

[12] Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Hasan, M.; Van Essen, B.C.; Awwal, A.A.S.; Asari, V.K. A State-of-the-Art Survey on Deep Learning Theory and Architectures. Electronics 2019, 8, 292. https://doi.org/10.3390/electronics8030292.

[13] Yao, Guangle & Lei, Tao & Zhong, Jiandan. (2018). A Review of Convolutional-Neural-Network-Based Action Recognition. Pattern Recognition Letters. 118. 10.1016/j.patrec.2018.05.018.

[14] Xiao, Y., Tian, Z., Yu, J. et al. A review of object detection based on deep learning. Multimed Tools Appl 79, 23729–23791 (2020). https://doi.org/10.1007/s11042-020-08976-6

[15] Hastie, T., Tibshirani, R., Friedman, J. (2009). Unsupervised Learning. In: The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York, NY. https://doi.org/10.1007/978-0-387-84858-7_14

[16] Radford, Alec & Metz, Luke & Chintala, Soumith. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

[17] J. Jia and W. Wang, "Review of reinforcement learning research," 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Zhanjiang, China, 2020, pp. 186-191, doi: 10.1109/YAC51587.2020.9337653.

[18] Rui Nian, Jinfeng Liu, Biao Huang, A review On reinforcement learning: Introduction and applications in industrial process control, Computers & Chemical Engineering, Volume 139, 2020, 106886, ISSN 0098-1354, https://doi.org/10.1016/j.compchemeng.2020.106886.

[19] Liu, Weibo et al. “A survey of deep neural network architectures and their applications.” Neurocomputing 234 (2017): 11-26.

[20] Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, Shyu ML, Chen SC, Iyengar S. A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surv (CSUR). 2018;51(5):1–36.

[21] Dhillon, A., Verma, G.K. Convolutional neural network: a review of models, methodologies and applications to object detection. Prog Artif Intell 9, 85–112 (2020). https://doi.org/10.1007/s13748-019-00203-0

[22] Feldmann, S., Schmiedt, M., Schlosser, J.M. et al. Recursive quality optimization of a smart forming tool under the use of perception based hybrid datasets for training of a Deep Neural Network. Discov Artif Intell 2, 17 (2022). https://doi.org/10.1007/s44163-022-00034-4

[23] Ciampiconi, L., Elwood, A., Leonardi, M., Mohamed, A., & Rozza, A. (2023). A survey and taxonomy of loss functions in machine learning. arXiv preprint arXiv:2301.05579.

[24] Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., & Chavez-Urbiola, E. A. (2023). Loss Functions and Metrics in Deep Learning. A Review. arXiv preprint arXiv:2307.02694.

[25] Sun, R. (2019). Optimization for deep learning: theory and algorithms. arXiv preprint arXiv:1912.08957.

[26] Amari, S. I. (1993). Backpropagation and stochastic gradient descent method. Neurocomputing, 5(4-5), 185-196. https://doi.org/10.1016/0925-2312(93)90006-O

[27] Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent Advances in Stochastic Gradient Descent in Deep Learning. Mathematics, 11(3), 682. https://doi.org/10.3390/math11030682

[28] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147). PMLR.

[29] Dauphin, Y., De Vries, H., & Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. Advances in neural information processing systems, 28.

[30] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[31] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7).

[32] Dozat, T. (2016) Incorporating Nesterov Momentum into Adam. Proceedings of the 4th International Conference on Learning Representations, Workshop Track, San Juan, Puerto Rico, 2-4 May 2016, 1-4.

[33] Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. Engineering Applications of Artificial Intelligence, 60, 97-116.

[34] Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.

[35] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

[36] Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013, May). Improving deep neural networks for LVCSR using rectified linear units and dropout. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 8609-8613). IEEE.

[37] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27 (pp. 270-279). Springer International Publishing.

[38] J. Schmidhuber, “Evolutionary Principles In Self-referential Learning,” On learning how to learn: The meta-meta-... hook, 1987.

[39] Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning (pp. 1126-1135). PMLR.

[40] Qian, K., & Yu, Z. (2019). Domain adaptive dialog generation via meta learning. arXiv preprint arXiv:1906.03520.

[41] Vanschoren, J. (2019). Meta-Learning. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham. https://doi.org/10.1007/978-3-030-05318-5_2

[42] Ma, P., Zhang, Z., Wang, J., Zhang, W., Liu, J., Lu, Q., & Wang, Z. (2021). Review on the application of metalearning in artificial intelligence. Computational intelligence and neuroscience, 2021.

[43] Koch, G., Zemel, R., & Salakhutdinov, R. (2015, July). Siamese neural networks for one-shot image recognition. In ICML deep learning workshop (Vol. 2, No. 1).

[44] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.

[45] Garcia, V., & Bruna, J. (2017). Few-shot learning with graph neural networks. arXiv preprint arXiv:1711.04043.

[46] Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for one shot learning. Advances in neural information processing systems,

[47] Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. Advances in neural information processing systems, 30.

[48] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1199-1208).

[49] Shyam, P., Gupta, S., & Dukkipati, A. (2017, July). Attentive recurrent comparators. In International conference on machine learning (pp. 3173-3181). PMLR.

[50] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016, June). Meta-learning with memory-augmented neural networks. In International conference on machine learning (pp. 1842-1850). PMLR.

[51] Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P. (2016). Rl $^ 2$: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779.

[52] Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., ... & Botvinick, M. (2016). Learning to reinforcement learn. arXiv preprint arXiv:1611.05763.

[53] Munkhdalai, T., & Yu, H. (2017, July). Meta networks. In International conference on machine learning (pp. 2554-2563). PMLR.

[54] Mishra, Nikhil et al. “A Simple Neural Attentive Meta-Learner.” International Conference on Learning Representations (2017).

[55] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., ... & De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. Advances in neural information processing systems, 29.

[56] Ravi, S., & Larochelle, H. (2016, November). Optimization as a model for few-shot learning. In International conference on learning representations.

[57] Li, Z., Zhou, F., Chen, F., & Li, H. (2017). Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835.

[58] Grant, E., Finn, C., Levine, S., Darrell, T., & Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical bayes. arXiv preprint arXiv:1801.08930.

[59] Finn, C., Rajeswaran, A., Kakade, S., & Levine, S. (2019, May). Online meta-learning. In International Conference on Machine Learning (pp. 1920-1930). PMLR.

[60] Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., & Ahn, S. (2018). Bayesian model-agnostic meta-learning. Advances in neural information processing systems, 31.

[61] Rajeswaran, A., Finn, C., Kakade, S. M., & Levine, S. (2019). Meta-learning with implicit gradients. Advances in neural information processing systems, 32.

[62] Nichol, A., & Schulman, J. (2018). Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999, 2(3), 4.

[63] Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., & Hadsell, R. (2018). Meta-learning with latent embedding optimization. arXiv preprint arXiv:1807.05960.

[64] Bertinetto, L., Henriques, J. F., Torr, P. H., & Vedaldi, A. (2018). Meta-learning with differentiable closed-form solvers. arXiv preprint arXiv:1805.08136.

[65] Lee, K., Maji, S., Ravichandran, A., & Soatto, S. (2019). Meta-learning with differentiable convex optimization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10657-10665).

[66] Finn, C., Xu, K., & Levine, S. (2018). Probabilistic model-agnostic meta-learning. Advances in neural information processing systems, 31.

[67] Kaveh, M., Mesgari, M.S. Application of Meta-Heuristic Algorithms for Training Neural Networks and Deep Learning Architectures: A Comprehensive Review. Neural Process Lett 55, 4519–4622 (2023). https://doi.org/10.1007/s11063-022-11055-6

[68] Devikanniga, D., Vetrivel, K., & Badrinath, N. (2019, November). Review of meta-heuristic optimization based artificial neural networks and its applications. In Journal of Physics: Conference Series (Vol. 1362, No. 1, p. 012074). IOP Publishing.

[69] Wong, W. K., & Ming, C. I. (2019, June). A review on metaheuristic algorithms: recent trends, benchmarking and applications. In 2019 7th International Conference on Smart Computing & Communications (ICSCC) (pp. 1-5). IEEE.

[70] Tian, Z., & Fong, S. (2016). Survey of Meta-Heuristic Algorithms for Deep Learning Training. InTech. doi: 10.5772/63785

[71] Beni, G., & Wang, J. (1993). Swarm intelligence in cellular robotic systems. In Robots and biological systems: towards a new bionics? (pp. 703-712). Berlin, Heidelberg: Springer Berlin Heidelberg.

[72] Mucherino, A., & Seref, O. (2007, November). Monkey search: a novel metaheuristic search for global optimization. In AIP conference proceedings (Vol. 953, No. 1, pp. 162-173). American Institute of Physics.

[73] Mirjalili, S. (2015). The ant lion optimizer. Advances in engineering software, 83, 80-98.

[74] Yang, X. S., & Hossein Gandomi, A. (2012). Bat algorithm: a novel approach for global engineering optimization. Engineering computations, 29(5), 464-483.

[75] Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. International journal of swarm intelligence, 1(1), 36-50.

[76] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4), 28-39.

[77] Yang, XS., Deb, S. Cuckoo search: recent advances and applications. Neural Comput & Applic 24, 169–174 (2014). https://doi.org/10.1007/s00521-013-1367-1

[78] Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial intelligence review, 42, 21-57.

[79] Połap, D., & Woźniak, M. (2017). Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism. Symmetry, 9(10), 203. DOI:10.3390/sym9100203

[80] Yang, X. S. (2009). Harmony search as a metaheuristic algorithm. Music-inspired harmony search algorithm: theory and applications, 1-14.

[81] Rutenbar, R. A. (1989). Simulated annealing algorithms: An overview. IEEE Circuits and Devices magazine, 5(1), 19-26.

[82] Abdechiri, M., Meybodi, M. R., & Bahrami, H. (2013). Gases Brownian motion optimization: an algorithm for optimization (GBMO). Applied Soft Computing, 13(5), 2932-2946.

[83] Yan, G. W., Hao, Z., & Xie, J. (2013). A novel atmosphere clouds model optimization algorithm. Journal of Computers (Taiwan), 24(3), 26-39.

[84] Ghorbani, N., & Babaei, E. (2014). Exchange market algorithm. Applied soft computing, 19, 177-187. https://doi.org/10.1016/j.asoc.2014.02.006

[85] Gudise, V. G., & Venayagamoorthy, G. K. (2003, April). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706) (pp. 110-117). IEEE.

[86] Ilonen, J., Kamarainen, J. K., & Lampinen, J. (2003). Differential evolution training algorithm for feed-forward neural networks. Neural Processing Letters, 17, 93-105.

[87] Kulluk, S., Ozbakir, L., & Baykasoglu, A. (2012). Training neural networks with harmony search algorithms for classification problems. Engineering Applications of Artificial Intelligence, 25(1), 11-19.

[88] Karaboga, D., Akay, B., & Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In Modeling Decisions for Artificial Intelligence: 4th International Conference, MDAI 2007, Kitakyushu, Japan, August 16-18, 2007. Proceedings 4 (pp. 318-329). Springer Berlin Heidelberg.

[89] Sexton, R. S., Dorsey, R. E., & Johnson, J. D. (1999). Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. European Journal of Operational Research, 114(3), 589-601.

[90] R. Livni, S. Shalev-Shwartz, O. Shamir, "On the Computational Efficiency of Training Neural Networks" Advances in Neural Information Processing Systems, vol. 1 2014.

CURRICULUM VITAE

Name Surname: Ömer MİRHAN

EDUCATION:

- **Bachelor in Computer Engineering**: 2006, Damascus University, Faculty of Information Technology, Department of Software Engineering
- **Master in Management Information Systems**: 2010, Arab Academy for Banking & Financial Sciences, Management Information Systems
- **PhD in Computer and Information Engineering**: 2023 (Expected), Sakarya University, Computer and Information Engineering

PROFESSIONAL EXPERIENCE AND AWARDS:

- Twenty years of experience in the field of software development and information systems management. Currently working as ERP Project Manager in a software development and consultancy company
- Areas of expertise include: ERP systems implementation, business intelligence, data warehousing, software development, system analysis, software project management and database administration

LANGUAGES

- Arabic
- English
- Turkish

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Mirkhan, Amer & Çelebi, Numan. (2022). Binary Representation of Polar Bear Algorithm for Feature Selection. Computer Systems Science and Engineering. 43. 767-783. 10.32604/csse.2022.023249.

- Mirkhan, A., Çelebi, N. (2021). Finding the Optimal Features Reduct, a Hybrid Model of Rough Set and Polar Bear Optimization. In: INFUS 2020. Advances in Intelligent Systems and Computing, vol 1197. Springer, Cham. https://doi.org/10.1007/978-3-030-51156-2_186
- Mirkhan, Amer and Çelebi, Numan, "A hybrid model of heuristic algorithm and gradient descent to optimize neural networks", Bulletin of the Polish Academy of Sciences Technical Sciences vol. 71, no. 6, doi=10.24425/bpasts.2023.147924, BPASTS, 2023
- Mirkhan, A., Çelebi, N. (2023). Dynamic Heuristic Approach to Enhance the Performance of Few-Shot Meta-Learning. In: The 16th International Conference on the Developments in eSystems Engineering (DeSE2023)