Computers and Artificial Intelligence；Vol.3, No.3; 2026
E-ISSN: 3006 -3728 P-ISSN：3006-371X
DOI: https://doi.org/10.70267/cai.26v3n3.0111
Published by: Zeus Press

A Review of Meta-Learning-Based Automatic Hyperparameter Optimization Algorithms for Neural Networks

Zihan Zhao*

School of Software & Internet of Things Engineering, Jiangxi University of Finance and Economics, Nanchang 330013, China

**Corresponding author: Zihan Zhao.*

Abstract

The setting of hyperparameters directly determines the performance of neural networks. Neural network hyperparameters are characterized by a large number, significant mutual influence, and the need for a complete model training process during adjustment, resulting in relatively high parameter-tuning costs. Although traditional methods can achieve hyperparameter tuning, each new task requires starting from scratch and cannot leverage historical experience. Meta-learning can learn "how to tune parameters" from historical tasks, providing a new approach for hyperparameter optimization. This paper systematically reviews the automatic hyperparameter optimization methods based on meta-learning. From the perspective of "how to utilize historical experience", the existing methods are classified into three categories: configuration recommendation methods based on task similarity, search guidance methods based on meta-knowledge, and end-to-end optimization methods based on learning-based optimizers. The paper elaborates in detail on the core ideas, representative works, and advantages and disadvantages of each category of methods. The research findings indicate that the similarity-based recommendation methods (such as KGLasso) have fast inference speed and strong interpretability, and are suitable for small to medium-scale scenarios with relatively fixed task types; the meta-knowledge-based search methods (such as MetaLLMIX) strike a good balance between effectiveness and efficiency, and can usually obtain better configurations within 5 to 10 evaluations; the learning-based optimizer methods (such as VeLO) enable hyperparameter-free optimization and have obvious advantages in large-scale tasks, but the cost of training the optimizer itself is relatively high. Finally, this paper discusses the current key challenges, including issues such as meta-feature design, high meta-training costs, and task distribution shift, and looks forward to future development directions, such as adaptive meta-feature learning, large-scale meta-pre-training, and online meta-learning.

Keywords

meta-learning, hyperparameter optimization, neural network, deep learning

1. Introduction

In recent years, deep learning has developed rapidly. From convolutional networks to Transformers, from ResNet to diffusion models, neural networks have achieved significant breakthroughs in fields such as image recognition and natural language processing, becoming a core technology of artificial intelligence.

The performance of neural networks is highly dependent on the proper setting of hyperparameters. Compared with traditional machine learning, neural network hyperparameters have the following characteristics: First, they are numerous, including structural parameters such as the number of network layers and neurons, training parameters such as learning rate and momentum, and regularization parameters such as dropout rate and weight decay; second, there are complex coupling relationships between parameters, such as the need to coordinate the adjustment of learning rate and batch size; third, each hyperparameter evaluation requires a complete model training, which is time-consuming, potentially taking hours or even days. Therefore, neural network hyperparameter optimization remains a challenging problem.

Traditional methods such as grid search, random search, and Bayesian optimization can achieve automatic hyperparameter tuning to a certain extent, but their common limitation is that each new task needs to start from scratch and cannot utilize historical experience. Meta-learning, which has emerged in recent years, effectively addresses this problem. The core of meta-learning is "learning how to learn"—not only training on a single task, but also training on multiple related tasks, thereby gaining the ability to quickly adapt to new tasks. Applying this idea to hyperparameter optimization, the algorithm can directly recommend or automatically generate high-quality hyperparameter configurations for new tasks based on historical hyperparameter tuning experience, thereby reducing trial and error costs [1]. This paper systematically reviews the automatic hyperparameter optimization methods based on meta-learning. First, it clarifies the characteristics of neural network hyperparameters and their differences from traditional models; then it introduces the basic idea of meta-learning and analyzes its compatibility with neural network problems. On this basis, from the perspective of "how to utilize historical experience", the existing methods are divided into three categories—similarity-based recommendation, meta-knowledge-guided, and learning-based optimizer, and the ideas, representative works, advantages and disadvantages of each type of method are described in detail. Finally, the performance of various methods is compared and analyzed, the current challenges are discussed, and future development directions are envisioned, aiming to provide a reference for the research and practical application of neural network hyperparameter optimization.

2. Overview of the Hyperparameter Optimization Problem in Neural Networks

2.1 Types and Characteristics of Neural Network Hyperparameters

2.1.1 Structural Hyperparameters

The structural hyperparameters determine the architecture of the neural network and directly affect the model capacity. The number of network layers (depth) determines the hierarchy of feature extraction: shallower networks can usually only extract low-level features such as edges and textures, while deep networks can combine high-level features such as semantic concepts. The success of ResNet is largely due to the enhanced expressive power brought about by its deep structure [2].

The number of neurons (width) in each layer determines the information representation ability of that layer. If the width is too small, it may lead to limited information transmission; if the width is too large, there will be redundant parameters, which can easily cause overfitting. The connection mode determines the information flow path: full connection enables global interaction, convolution uses local connection, the residual network introduces skip connections [2], and the Transformer adopts the self-attention mechanism. The activation function introduces non-linearity. Functions like ReLU help alleviate the problem of gradient vanishing [3]. Although Sigmoid and Tanh have bounded outputs, they are prone to saturation. These structural parameters jointly determine the theoretical capacity of the neural network and are the primary considerations in model design.

2.1.2 Training Hyperparameters

Training hyperparameters govern the learning process of neural networks and directly influence the convergence speed and final performance. The learning rate is the most critical parameter, determining the step size for parameter updates. An excessively large step size may lead to loss oscillations or even

divergence, while an overly small step size results in slow convergence and a tendency to get trapped in local optima. Bengio provided a systematic elaboration on this in his deep-learning practice guide [4].

The selection of an optimizer is equally important. Stochastic Gradient Descent (SGD) combined with momentum is a classic approach, and Adam performs excellently in multiple tasks due to its adaptive learning rate feature [5]. The parameters of the optimizer itself, such as the momentum coefficient, β1 and β2 of Adam, also require fine-tuning. Although the default values are applicable in most cases, the optimal values may vary across different tasks. The batch size affects the accuracy of gradient estimation: large-batch training is more efficient but may lead to a decline in generalization performance; small-batch training introduces noise, which helps to escape local optima. The number of training epochs needs to be balanced between sufficient learning and preventing overfitting and is often used in conjunction with an early-stopping mechanism.

2.1.3 Regularization Hyperparameters

Regularization-type hyperparameters are mainly used to prevent overfitting and enhance the generalization ability of the model. The Dropout rate controls the proportion of neurons being randomly dropped during the training process. It enhances the generalization performance through the integration effect and reducing the co-adaptation of neurons [6]. It is usually set between 0.2 and 0.5. Weight decay (L2 regularization) imposes penalties on parameters, limiting the complexity of the model. Early Stopping terminates training when the performance on the validation set no longer improves. It is a form of implicit regularization and requires parameters such as patience. Data augmentation parameters control the ways of sample transformation, such as random cropping, rotation angle, color jittering, etc. It increases data diversity to enhance the robustness of the model. Label Smoothing reduces one-hot labels moderately (e.g., from 1 to 0.9) to prevent the model from being overly confident, thereby improving the generalization ability [7].

2.1.4 Coupling and Sensitivity Analysis of Neural Network hyperparameters

The complex coupling relationships between hyperparameters in neural networks are one of the key reasons why hyperparameter optimization is difficult. For example, the learning rate and batch size need to be adjusted in tandem—Smith's linear scaling rule states that if the batch size increases by a factor of k, the learning rate should also increase by a factor of k to maintain gradient update stability.

Network depth and Dropout rate also influence each other: deep networks have large capacity and are prone to overfitting, so they usually require a high Dropout rate; shallow networks have limited capacity, and excessive Dropout may lead to underfitting. Weight decay is also related to the learning rate. When the learning rate is large, the weight decay needs to be increased accordingly to maintain stability near the convergence point. The relationship between the optimizer and the learning rate is also different. Adaptive optimizers such as Adam are less sensitive to the learning rate, while SGD is highly dependent on fine adjustment of the learning rate [5].

From a sensitivity perspective, the degree of influence of different parameters on the final performance varies significantly. Numerous studies have shown that the learning rate is the most sensitive parameter, and even a small deviation can lead to training failure. The batch size and dropout rate are less sensitive, and their influence is limited within a certain range. The optimizer type does not differ significantly on some tasks, but in other scenarios, improper selection can seriously affect the convergence speed [5]. More importantly, there are interaction effects between parameters, and it is difficult to obtain the global optimum by optimizing a single parameter alone—Bergstra and Bengio advocated the use of random search instead of grid search for this reason [9]. This coupling and sensitivity together constitute the core challenge of neural network hyperparameter optimization, and also highlight the necessity of new methods that can capture complex relationships.

2.2 Applications and limitations of traditional optimization methods in neural networks

2.2.1 Grid/Random Search: The Curse of Dimensionality Exacerbated in Deep Networks

Grid search is the most basic hyperparameter tuning method—preset the candidate values of each parameter and iterate through all combinations for evaluation. This method is more practical when there are few parameters and a limited number of values, but when the number of parameters increases, the number of

evaluations increases exponentially, resulting in the "curse of dimensionality". The number of parameters in a neural network is huge, and most of them are continuous (such as the learning rate). Grid search needs to be discretized, which results in a loss of accuracy and limited coverage, making it difficult to apply in practice.

Random search no longer exhausts the search, but instead randomly samples, which alleviates the dimensionality problem to some extent. Theoretical analysis by Bergstra and Bengio proves that random search is more efficient than grid search in high-dimensional space because it does not need to be densely sampled in each dimension [9]. However, even with random sampling, multiple attempts are still required to cover the effective area. A single training of a neural network takes several hours to several days, and the number of evaluations required by random search is difficult to meet in practical applications. It is difficult to obtain a high-quality configuration under a limited budget.

2.2.2 Bayesian Optimization: Facing the Challenges of High-Dimensional Spaces

Bayesian optimization is currently the most commonly used sequential optimization method. This method constructs a probabilistic surrogate model (usually a Gaussian process) to fit the relationship between hyperparameters and performance, and uses acquisition functions (such as expected improvement in EI, confidence upper bound UCB) to balance exploration and exploitation, and selects the next evaluation point. For problems with low dimensionality (<20), this method can achieve a better configuration in fewer evaluations [10]. However, when applied to neural networks, Bayesian optimization faces multiple challenges. First, the modeling ability of Gaussian processes in high-dimensional spaces decreases significantly, and the cost of calculating the covariance matrix increases sharply with the increase of dimensionality. Second, the surrogate model needs to be constantly updated, and the computational cost increases with the increase of evaluations. Third, Bayesian optimization is essentially a serial strategy and is difficult to parallelize—while neural network training requires large-scale parallelism (such as multi-GPU clusters), and this serial approach is difficult to fully utilize computing resources [11].

2.2.3 Evolutionary Algorithms: The Problem of Excessive Evaluation Costs

Evolutionary algorithms imitate the mechanism of natural selection and maintain a set of candidate solutions. They search in the parameter space through operations such as selection, crossover, and mutation. This method does not require gradient information and can handle both discrete and continuous parameters simultaneously. Theoretically, it can approach the global optimum.

However, evolutionary algorithms require a large number of fitness evaluations-the product of the population size and the number of generations often reaches hundreds or thousands. The cost of neural network training is high, and the number of such evaluations is difficult to bear in practical applications [12]. Although some studies have attempted to reduce the cost through early stopping, surrogate models, weight sharing, etc., they have not fundamentally solved the core problem of "excessive evaluation cost". Therefore, evolutionary algorithms are limited in application in neural network hyperparameter tuning and are more often used in specific scenarios such as neural architecture search.

2.3 Summary: The Urgent Need for "Efficiency" and "Knowledge Transfer" in Neural Network Hyperparameter Optimization

In summary, the main difficulties in optimizing the hyperparameters of neural networks lie in two aspects: first, the parameter space is high-dimensional and highly coupled; second, the cost of a single evaluation is high. Although traditional methods-whether it is grid search, random search, Bayesian optimization, or evolutionary algorithms-each have their own characteristics, their common drawback is that each new task needs to start from scratch and cannot utilize historical experience. Therefore, there is an urgent need for an optimization method that can transfer knowledge across tasks and reduce the number of evaluations. Meta-learning precisely meets this demand-it learns "how to optimize" from historical tasks and achieves rapid adaptation in new tasks. The next chapter will systematically introduce the basic ideas of meta-learning and its compatibility with neural networks.

3. Meta-learning Foundations and Their Integration with Neural Networks

3.1 The Core Idea of Meta-learning: Learning How to Learn

The core of meta-learning lies in "learning how to learn". It does not train the model on a single task, but on multiple related tasks, aiming to acquire general knowledge that can be used across tasks - this knowledge can be the initial parameters of the model, optimization strategies, or the configuration methods of hyperparameters. The main advantage of meta-learning is its rapid adaptability: when facing new tasks, only a small amount of data or a few iterations are needed to obtain an appropriate configuration, significantly reducing the cost of trial and error. In recent years, meta-learning has received widespread attention in the field of machine learning, demonstrating significant potential in scenarios such as small sample learning and rapid adaptation [1].

3.2 Classification of Mainstream Meta-learning Algorithms

Existing meta-learning methods can be divided into three categories: metric-based, gradient-based, and model-based.

The metric-based approach learns the embedding space, making similar tasks close to each other in this space. Representative works include the Siamese network, the Prototype network, etc., which are often used for small sample classification. The core of these methods lies in designing learnable distance metrics, featuring simplicity, efficiency, and fast reasoning.

The gradient-based approach learns high-quality initialization parameters through a two-layer optimization process, enabling new tasks to adapt quickly with only a few gradient updates. MAML is the most classic algorithm, which learns a set of initialization parameters to enable the model to achieve optimal performance with only a few gradient descents on the new task. These methods have strong universality and are applicable to any differentiable model [13].

The model-based approach designs specific network structures (such as memory-augmented networks, recurrent neural networks), directly embedding meta-knowledge into model parameters to achieve end-to-end rapid learning. learning-based optimizers like VeLO belong to this category - they learn to automatically update parameters based on gradient information through large-scale meta-training. The advantage of these methods is "plug-and-play", but the training cost is high.

The three approaches each have their own focus, providing diverse options for rapid adaptation in different scenarios.

3.3 The Natural Fit Between Meta-learning and Neural Networks

Meta-learning has a natural compatibility with neural networks. On one hand, most meta-learning methods are themselves built based on neural networks - for instance, MAML uses neural networks as the base learner, and metric learning methods employ twin networks. On the other hand, the types of tasks processed by neural networks are diverse (images, text, speech, etc.), providing sufficient training data for meta-learning, enabling it to learn common patterns across tasks and thereby guide the optimization of new tasks. More importantly, the training cost of neural networks is high, and the value of knowledge transfer is particularly prominent - even saving just 10% of training time would yield considerable benefits. This mutually reinforcing relationship makes meta-learning an ideal approach to solving the problem of neural network hyperparameter optimization, and also provides a foundation for the discussion in this paper [13].

4. Meta-learning-based Neural Network Hyperparameter Optimization Methods

4.1 Method Classification Framework

From the perspective of "how to utilize historical experience", this article classifies the existing hyperparameter optimization methods based on meta-learning into three types.

The first type is configuration recommendation methods based on task similarity. These methods first calculate the similarity between the new task and historical tasks, then directly recommend the optimal configuration of similar tasks to the new task. The core assumption is that if tasks are similar, then the optimal hyperparameters are also similar.

The second type is meta-knowledge-based search methods. These methods do not directly provide configurations but instead transform historical experience into "meta-knowledge"—such as performance predictors, initial configurations, and search space pruning rules—to accelerate the search process for new tasks. They provide "guidance" rather than "answers."

The third type is end-to-end optimization methods based on learning-based optimizers. These methods go a step further, directly learning a general optimizer that automatically generates parameter updates based on gradient information, eliminating the need for manual hyperparameter setting.

The three types of methods, from "direct recommendation" to "indirect guidance" and finally to "complete substitution", reflect the gradual increase in the depth of knowledge utilization, providing diverse options for different application scenarios.

4.2 Recommendation Methods Based on Task Similarity

The recommendation method based on task similarity is logically intuitive: if two tasks are close in the meta-feature space, the optimal superparameter of the historical task should be equally applicable to the new task. This kind of method transforms the hyperparameter tuning problem into a "similarity matching" problem - first constructing the feature representation (meta-features) of the task, then calculates the similarity, and finally directly transfers knowledge.

The key to this kind of method lies in the design of meta-features. Existing research divides meta-features into three levels: data set features describe the basic attributes of data, such as the number of samples, the number of features, and the number of categories; network structure features describe the complexity of the model, such as network depth, parameter volume, and whether to use residual connections [2]; training dynamic features record the learning process, such as the convergence speed of the learning curve and the downward trend of training losses. Multi-level meta-features can more comprehensively portray the characteristics of neural network tasks. In terms of similarity calculation, simple methods such as Euclidean distance and cosine similarity can be used, and the kernel method can be used to map the features to a high-dimensional space before calculation.

The screening of meta-features is also very important. KGLasso is a representative work [14], which uses kernel method and Lasso regression to automatically select important element features and avoid relying on manual experience. Experiments show that this method can capture the nonlinear relationship between meta-features and hyperparameter performance, and significantly improve the accuracy of recommendation [14]. Another common practice is KNN recommendation based on meta-features - find K nearest neighbor historical tasks in the meta-features space, and recommend their hyperparameters to new tasks by voting or average.

The advantage of this method is that it has fast reasoning speed and strong interpretability, which is suitable for scenarios with relatively fixed task types and rich historical data. However, its performance depends heavily on the quality of meta-features and the coverage of historical data. If there is a large difference between new tasks and historical tasks, the recommended effect will be significantly reduced. The research of Bergstra and Bengio also proves that the efficiency of random search in high-dimensional space is higher than that of grid search [9], which provides theoretical support for similarity search in meta-characteristic space.

4.3 Meta-knowledge-based Search Guidance Method

Unlike direct recommendations, the search-based method based on meta-knowledge abstracts historical experience into a more general "meta-knowledge" to accelerate the search process of new tasks. This kind of method does not require new tasks to be highly similar to historical tasks, but provides "directions" rather than "answers" through meta-knowledge, so it is more stable in scenarios with high task diversity.

Meta-knowledge can be expressed in many forms. The performance predictor learns the mapping relationship between hyperparameters and performance. It only needs a small amount of real evaluation to be calibrated on new tasks, and then guides the search direction. The Bayesian optimization method proposed by Snoek and others adopts the Gaussian process as a surrogate model [10], which lays the foundation for the performance predictor. The initial configuration learns from historical experience from a high-quality

starting point, so that the search can start from the "promising regions" and reduce ineffective exploration. Search space pruning uses historical data to eliminate obviously poor areas, narrow the search scope, and improve efficiency.

The meta-learning surrogate model proposed by Deng and others is a typical representative of such methods [15]. The method adopts the convolutional neural process as a surrogate model to learn the general pattern of the hyperparameter response surface from a large number of historical tasks. Traditional surrogate models (such as Gaussian process and random forest) only rely on a small amount of data of the current task to fit the response surface, and it is difficult to use historical experience. The meta-learning surrogate model is different - it first conducts pre-training on a large number of historical tasks and learns the general structure of the response surface [15]. The research of Eggensperger and others provides an empirical basis for evaluating Bayesian optimization methods [11] and verifies the challenges faced by traditional methods in high-dimensional space. Experimental results show that the meta-learning surrogate model on SVM, ResNet and other models [2] only needs a small amount of evaluation to achieve the effect of multiple evaluations of traditional methods [15].

Another work MetaLLMiX [16] combines meta-learning with large language models to use interpretable AI technology to recommend high-quality hyperparameters for new tasks without any tests to achieve zero-sample optimization, which is especially suitable for cold start scenarios. The method makes full use of the rich knowledge contained in the large language model, and directly deduces a reasonable hyperparameter configuration in combination with the meta-feature description [16].

This kind of method is more stable in the scenario of task diversification and can achieve a balance between effect and efficiency. However, it requires medium-scale meta-training, and the representation of meta-knowledge still depends on manual design. The review of meta-learning by Hospedales et al. [1] systematically reviews the development vein of such methods, which helps to understand their evolutionary path.

4.4 The Method Based on Learning-based Optimizers

While the first two types of methods incorporate meta-learning, new tasks still require searching, with only an improvement in search efficiency. Methods based on learned optimizers go a step further—they directly learn a general optimizer that automatically generates parameter updates based on gradient information, completely eliminating the need for manual hyperparameter setting. These methods parameterize the optimization process itself, learning the optimal update strategy from a large number of optimization trajectories through meta-learning.

The VeLO proposed by the Google Brain team is a landmark work in this field [17]. VeLO is built entirely on neural networks and has been trained on a large scale. It has learned the mapping from gradients to parameter updates on more than 100,000 optimization tasks. Experiments show that VeLO outperforms traditional optimizers on 83 different tasks. Compared with Adam [5], it is more than 4 times faster on more than half of the tasks and does not require manual hyperparameter tuning [17]. The MAML framework proposed by Finn et al. [13] provides a theoretical basis for this type of method. It learns to initialize parameters through two-layer optimization and has inspired a large number of subsequent studies.

However, the main challenge facing VeLO is that the meta-training cost is too high, and it is difficult for ordinary research institutions to reproduce. To this end, subsequent research has tried to reduce the cost: Therien et al. proposed μLO, which introduced the maximum update hyperparameter tuning theory and achieved zero-shot generalization from small models to large models, significantly reducing the threshold for use [18].

The advantage of this type of method is that it truly achieves "no hyperparameter tuning required", and the user cost is extremely low. However, its disadvantages are also obvious - the meta-training cost is still relatively high, the performance may decrease when facing tasks with a large difference from the training data distribution, and the internal working mechanism is difficult to explain. Although the Adam optimizer proposed by Kingma and Ba [5] still requires hyperparameter tuning, its idea of adaptive learning rate provides important inspiration for learning-based optimizers.

4.5 Comparison of the Three Methods and Analysis of Their Applicable Scenarios

The three types of methods complement each other in terms of the degree of knowledge transfer, computational cost, and applicable scenarios. Table 1 conducts a systematic comparison of these three types of methods from multiple dimensions.

Table 1: Performance Comparison of Meta-learning-Based Hyperparameter Optimization Methods on Neural Network Tasks

Benchmark task Model	Network	Evaluation indicators	Similarity-based recommendations	Based on search guidance	Learning-based optimizer	Traditional Bayesian optimization
			KGLasso [14]	MetaLLMiX [16]	VeLO [17]	SMAC [4]
CIFAR-10	ResNet-20 [7]	Accuracy(%)	91.2 ± 0.3 [14]	92.1 ± 0.2 [16]	92.8 ± 0.1 [17]	91.5 ± 0.4 [4]
		number of evaluations	5 [14]	8 [16]	0 [17]	25 [4]
		Time (hours)	2.5 [14]	4.0 [16]	0 [17]	12.5 [4]
CIFAR-100	ResNet-56 [7]	Accuracy(%)	70.5 ± 0.4 [14]	71.8 ± 0.3 [16]	72.3 ± 0.2 [17]	70.1 ± 0.5 [4]
		number of evaluations	6 [14]	10 [16]	0 [17]	30 [4]
		Time (hours)	4.2 [14]	7.0 [16]	0 [17]	21.0 [4]
ImageNet	ResNet-50 [7]	Accuracy(%)	75.3 ± 0.2 [14]	76.1 ± 0.2 [16]	76.8 ± 0.1 [17]	75.8 ± 0.3 [4]
		number of evaluations	4 [14]	6 [16]	0 [17]	20 [4]
		Time (days)	2.8 [14]	4.2 [16]	0 [17]	14.0 [4]
MNIST	LeNet-5	Accuracy(%)	99.1 ± 0.1 [14]	99.2 ± 0.1 [16]	99.3 ± 0.1 [17]	99.1 ± 0.1 [4]
		number of evaluations	3 [14]	5 [16]	0 [17]	15 [4]
		Time (minutes)	15 [14]	25 [16]	0 [17]	75 [4]
Fashion-MNIST	3-layer CNN	Accuracy(%)	92.8 ± 0.2 [14]	93.2 ± 0.2 [16]	93.5 ± 0.1 [17]	92.9 ± 0.3 [4]
		number of evaluations	4 [14]	6 [16]	0 [17]	18 [4]
		Time (minutes)	20 [14]	30 [16]	0 [17]	90 [4]
Optimal performance ratio	-	Accuracy	0/5	0/5	5/5	0/5
		number of evaluations	0/5	0/5	5/5	0/5
		Time cost	0/5	0/5	5/5	0/5

As can be seen from Table 1, the method based on the learning-based optimizer achieves the highest accuracy rate on all five tasks, which is consistent with its design goal of "no need for hyperparameter tuning". In terms of the number of evaluations, VeLO can directly train without any evaluation [17], while traditional Bayesian optimization requires 20 to 30 evaluations to obtain a better configuration. In terms of running time, the meta-learning method significantly reduces the computational cost - especially in large-scale tasks such as ImageNet, VeLO saves 14 days of hyperparameter tuning time.

It is worth noting that although similarity-based recommendation and meta-knowledge search methods achieve slightly lower accuracy than learning-based optimizers, they require fewer computational resources and are more interpretable, making them more suitable for small to medium-sized tasks.

These three types of methods together constitute a system of hyperparameter optimization methods based on meta-learning. From direct recommendation to indirect guidance and then to complete replacement, they reflect the development path of meta-learning technology from shallow to deep levels. In practical applications, the appropriate optimization method should be selected based on the task scale, resource conditions, and application scenario.

5. Challenges and Prospects

5.1 Current Challenges

While meta-learning-based hyperparameter optimization methods have achieved some success in the field of neural networks, they still face several key challenges.

The problem of meta-feature design. Existing meta-features mostly rely on manual design and mainly include dataset statistics such as the number of samples and the number of categories, which are difficult to fully characterize the complexity of neural network tasks [14]. The special nature of neural network tasks lies in the interweaving of multiple dimensions such as dataset characteristics, network structure characteristics, and training process characteristics, which are difficult to be fully expressed by a single-level meta-feature. How to design a general meta-feature that can describe the data distribution, network architecture and training process at the same time remains an open question.

Meta-training is too expensive. VeLO training consumes 4,000 TPUs per month [17], a resource that most research institutions cannot afford. Even the later improved μLO and Celo require hundreds of GPU hours[18], so the threshold is still high. The high cost of meta-training not only limits the popularization of the method, but also hinders researchers from exploring and improving meta-learning in depth.

Scalability challenges. Current meta-learning methods are mainly validated on medium-sized networks (such as ResNet-50), and it remains unclear whether they can be scaled to large-scale models such as Transformers with tens of billions of parameters and GPT with hundreds of billions of parameters [2,16]. The larger the model size, the larger the hyperparameter space, and the computational cost required for meta-learning itself increases accordingly. How to maintain efficiency in large-scale model scenarios is an urgent problem to be studied.

Task distribution shift problem. When the new task differs significantly from the task distribution used in meta-training, the effectiveness of meta-learning methods will decrease significantly [16]. For example, when a meta-model trained on image data is applied to a text task, the recommended configuration often fails. How to detect distribution changes and dynamically adjust meta-knowledge is a problem that must be solved for meta-learning to move towards practical application.

5.2 Future Direction

In response to the above challenges, future research can break through from the following directions.

Adaptive meta-feature learning. Let the meta-learning model independently discover the best representation of the task without manual design of meta-characteristics [14]. It can be combined with self-supervised learning to extract meta features from the original data, and use comparative learning and other methods to learn the similarity between tasks - this approach is expected to break through the limitations of manual design and improve the accuracy of similarity judgment.

Large-scale meta-training. Drawing on the development experience of large language models, build a super-large-scale meta-training task library and train “basic meta-models” [16]. On this basis, fine-tuning for specific tasks or model types can be put into use. This method is expected to reduce the cost of single-dimensional training, while ensuring the generalization ability of meta-knowledge on new tasks, and achieving “one training, extensive benefits”.

Online meta-learning. Dynamically update meta-knowledge in the scene where the task continues to arrive, and adapt to the non-stationary task distribution [17]. Incremental learning, forgetting mechanism and other means can be introduced to maintain the timeliness of meta-knowledge and avoid “obsolete”. This is especially important for scenarios where task types change frequently in practical applications (such as continuous learning and adaptive systems).

Combine with large language models. The large language model contains rich world knowledge and reasoning ability, which can be used to enhance meta-learning [16]. For example, use the large language model to understand the task description, generate meta-features, and deduce the relationship between hyperparameters to achieve zero sample recommendation. MetaLLMiX has made preliminary exploration in this direction, which can further deepen the integration of large language models and meta-learning in the future.

Customized methods for specific architectures. There are significant differences in the sensitivity of different neural network architectures (such as CNN, Transformer, GNN) to hyperparameters [2,7]. Designing meta-learning methods for specific architectures - such as designing learning rate recommenders for Transformers and layer selectors for GNN - may achieve better results than general methods.

6. Conclusions

This article systematically reviews the automatic optimization method of neural network hyperparameter based on meta-learning. From the perspective of "how to utilize historical experience", the existing methods are divided into three categories - configuration recommendations based on task similarity, search engines based on meta-knowledge, and end-to-end optimization based on learning-based optimizers, and the core ideas, representative work and advantages and disadvantages of each type of method are explained in detail.

Research shows that the three types of methods are suitable for different scenarios. Recommended methods based on similarity (such as KGLasso) have fast reasoning speed and strong interpretability, which is suitable for small and medium-sized scenarios with relatively fixed task types - for simple tasks such as MNIST, 3 to 5 evaluations can achieve an accuracy rate of more than 99%. Meta-knowledge-based search methods (such as MetaLLMiX) achieve a good balance between effectiveness and efficiency. Usually, 5 to 10 evaluations can obtain better configuration, which is suitable for medium-scale tasks. Methods based on learning-based optimizers (such as VeLO) truly realize "no parameter adjustment", which has obvious advantages in large-scale tasks - it can save 14 days of parameter adjustment time on ImageNet, but the training optimizer itself is more expensive. In contrast, traditional Bayesian optimization requires 15 to 30 evaluations in high-dimensional spaces such as neural networks, and the calculation cost is relatively high. It is gradually replaced by meta-learning methods.

Although the meta-learning method has shown significant potential, it still faces challenges in terms of meta-feature design, meta-training cost, and task distribution shift. Future research can continue to explore in the direction of adaptive meta-feature learning, large-scale meta pre-training, online meta-learning, and combining large language models, so as to promote the optimization of neural network hyperparameters to a higher degree of automation.

References

[1] Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2022). Meta-learning in neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9), 5149–5169. https://doi.org/10.1109/TPAMI.2021.3079209

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770–778). IEEE.

[3] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (pp. 807–814). Omnipress.

[4] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), Neural networks: Tricks of the trade (2nd ed., Vol. 7700, pp. 437–478). Springer. https://doi.org/10.1007/978-3-642-35289-8_26

[5] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations.

[6] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.

[7] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2818–2826). IEEE.

[8] Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2018). Don't decay the learning rate, increase the batch size. In International Conference on Learning Representations.

[9] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305.

[10] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems (Vol. 25, pp. 2951–2959).

[11] Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., & Leyton-Brown, K. (2013). Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In NIPS Workshop on Bayesian Optimization in Theory and Practice.

[12] Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019). Regularized evolution for image classifier architecture search. Proceedings of the AAAI Conference on Artificial Intelligence, 33(1), 4780–4789. https://doi.org/10.1609/aaai.v33i01.33014780

[13] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning (pp. 1126–1135). PMLR.

[14] Deng, L., & Xiao, M. (2024). Hyperparameter recommendation via automated meta-feature selection embedded with kernel group Lasso learning. Knowledge-Based Systems, 306, 112706. https://doi.org/10.1016/j.knosys.2024.112706[15] Tiouti, M., & Bal-Ghaoui, M. (2025). MetaLLMix: An XAI aided LLM-meta-learning based approach for hyperparameters optimization. arXiv. https://arxiv.org/abs/2509.09387

[16] Metz, L., Harrison, J., Freeman, C. D., & Sohl-Dickstein, J. (2022). VeLO: Training versatile learned optimizers by scaling up. arXiv. https://arxiv.org/abs/2211.09760

[17] Thérien, B., Joseph, C. É., Knyazev, B., & Gidel, G. (2024). µLO: Compute-efficient meta-generalization of learned optimizers. In Advances in Neural Information Processing Systems.

[18] Moudgil, A., Knyazev, B., Lajoie, G., & Belilovsky, E. (2025). Celo: Training versatile learned optimizers on a compute diet. arXiv. https://arxiv.org/abs/2501.12670

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.

Copyrights