UMYU Scientifica, Vol. 4 NO. 3, September 2025, Pp 009 – 018
https://doi.org/10.56919/2543.002
ISSN: 2955 – 1145 (print); 2955 – 1153 (online)
A periodical of the Faculty of Natural and Applied Sciences, UMYU, Katsina

REVIEW ARTICLE

An Efficient Neural Architecture Search Algorithm for AutoEncoder Optimization - A Systematic Literature Review

Samuel Ogbe Michael[1] and Ahmed Abubakar Aliyu[2]
[1]Department of Informatics, Faculty of Computing, Kaduna State University, Nigeria
[2]Department of Secure Computing, Faculty of Computing, Kaduna State University, Nigeria

ABSTRACT

Autoencoders have developed into neural search networks in recent years, and the majority of machine learning (ML) methods rely on the input properties to produce high-quality models. Increased dimensionality is a challenge that arises with larger datasets, which significantly reduces machine learning efficiency. In order to reduce the significant amount of high-dimensional data, researchers have developed feature reduction and selection techniques like Principal Component Analysis (PCA) and autoencoders with their different variations, including Convolutional autoencoders, Sparse autoencoders, Denoising autoencoders, Contractive autoencoders, Variational autoencoders, and Deep autoencoders. This paper discusses neural search structures, evolutionary methods, and various autoencoder types. A number of optimization techniques to increase training accuracy and decrease training time are also covered in the review. By thoroughly reviewing the literature from five scholarly databases on the subject, extracting pertinent information using a PRISMA flow diagram, and applying the proper eligibility criteria to synthesize and evaluate the best papers found, the research project adopted a thorough, detailed review. According to the research's findings, several neural search architectures and autoencoders used for data reconstruction have improved, including Venkataraman's Convolutional AEs, Baier et al.'s Self-Supervised Siamese Autoencoders, Lazebnik & Simon-Keren's Knowledge-integrated Autoencoder Model, Chen et al.'s Dirichlet Neural Architecture Search, and Zhang et al.'s Graph Hypernetworks for Neural Architecture Search.

ARTICLE HISTORY

Received 26 April, 2025
Accepted September 21, 2025
Published September 24, 2025

KEYWORDS

Algorithm optimization, Autoencoder, Machine learning, Neural Architecture Search.

INTRODUCTION

The study of how machines can change and adapt their behavior, improving their performance based on input information, is known as machine learning. The two most well-known subfields within this topic are supervised learning (such as regression and classification) and unsupervised learning (such as association rules and clustering) (Charte et al., 2019).

Most machine learning (ML) algorithms depend on the input characteristics to generate an excellent model (Shrestha & Mahmood, 2019). To address data issues, machine learning employs a range of algorithms (Mahesh, 2020). As the input data increase in dimensionality, the performance of the ML tends to decrease significantly (Singh et al., 2021. To deal with the considerable amount of high-dimensional data, researchers have developed feature reduction and selection techniques. One important method for reducing the dimensionality of data and locating data points with the highest variance is Principal Component Analysis (PCA) (Singh et al., 2021; Greenacre et al., 2023). However, PCA is a linear technique, so it cannot handle complex nonlinear data (Shrestha & Mahmood, 2019).

Autoencoders are generally used to perform dimensionality reduction of data (Popov et al., 2022). Autoencoders are unsupervised Artificial Neural Networks (ANN) that learn how to compress and encode data efficiently (Charte et al., 2018). They tend to learn how to reconstruct from the reduced encoded representation back to a close representation of the original input data representation as possible (Charte et al., 2020).

In the ML field, autoencoders have been of great importance because of their rapid ability to train a network to remove noise from input data and reconstruct it. The basic architecture of an autoencoder consists of three layers: encoder, code, and decoder (Bank et al., 2020). The autoencoders can be classified into the following types: Variational autoencoder, Convolutional autoencoder, Stack autoencoder, Sparse autoencoder, Denoising

Correspondence: Samuel Ogbe Michael. Department of Informatics, Faculty of Computing, Kaduna State University, Nigeria. ✉ sammygreat12345@gmail.com.

How to cite: Ogbe, S. M. & Abubakar, A. A. (2025). An Efficient Neural Architecture Search Algorithm for AutoEncoder Optimization - A Systematic Literature Review. *UMYU Scientifica*, 4(3), 9 – 18. https://doi.org/10.56919/2543.002

autoencoder, and Deep autoencoder, among others (Charte et al., 2018).

Methodology

The research work adopted a comprehensive, detailed review of an Efficient Neural Architecture Search Algorithm for Autoencoder Optimization. The method follows a precise step which includes:

Literature survey

I searched five (5) different academic databases to collect journal papers and conference proceedings on autoencoder neural networks, neural architecture search for autoencoders and optimized algorithm and their variations, and the evolutionary algorithm architecture. Google Scholar, IEEE Xplore, Science Direct, Springer, and Elsevier are the academic databases that were used in this study.

The lookup string utilized to search for the research papers from the academic databases was "autoencoders neural network", "neural architecture search for autoencoders", "optimized algorithm", "Principal component analysis", "Reinforcement Learning", and "Deep Neural Network".

Table 1: Summary of terminology and acronyms

Term/Acronym	Description
AE	Autoencoder
ANN	Artificial Neural Network
BAE	Bayesian auto-encoder
CAE	Contractive auto encoder
CDA	Coupled deep auto-encoder
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
DAE	Denoising auto-encoder
DE	Differential Evolution
DL	Deep Learning
EM	Evolutionary Method
ES	Evolution Strategy
EvoAAA	Evolutionary Methods for Automated Autoencoder Architecture search
GA	Genetic Algorithm
HSAE	Hessian regularized sparse auto-encoders
LAE	Laplacian regularized auto-encoder
LDA	Linear Discriminant Analysis
LLE	Locally Linear Embedding
MDA	multimodal deep auto-encoder
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
NCAE	Nonnegativity constraints auto-encoders
PCA	Principal Component Analysis
REPL	Representation Learning
SAE	Sparse auto-encoders
SCAE	Sparse contractive auto-encoder
SGD	Stochastic Gradient Descent
VANO	Variational Autoencoding Neural Operators

Data Extraction

After the first search, we retrieved 63,399 research papers from the databases. This was achieved by searching the academic databases with the search keywords and also joining the search words using Boolean 'OR'. Table 2 shows the paper collection and process employed to screen the output from each database. We further streamlined the search to only computer science-related papers and those published between 2019 and 2024 (in one case, 2017 paper was used due to its relevance to the research topic). A total of 6070 papers remained after this step, as 57,329 papers were removed. At the success of the previous step, we skimmed through the remaining 6070 papers' abstracts to isolate and discard those papers that were not relevant to the research and those that did not meet the inclusion criteria of the research outlined.

After the second screening, the number of useful research papers relevant to the selected SLR topic dropped to 33 papers. We further thoroughly read through the papers again via quality evaluation to achieve the aim of this research.

From the selected studies and papers, key information was systematically extracted, including the authors, publication year, methodologies, research outcomes, research contributions, research gaps, and datasets used. The data were organized to portray an elaborate review as shown in Figure 6.

Table 2: First Search Result

	Google Scholar	IEEE Xplore	Science Direct	Springer	Elsevier	Total
First Search	34,060	12749	8826	7644	120	63399
First Screening	1999	373	3230	457	11	6070
Second Screening	23	2	5	2	1	33

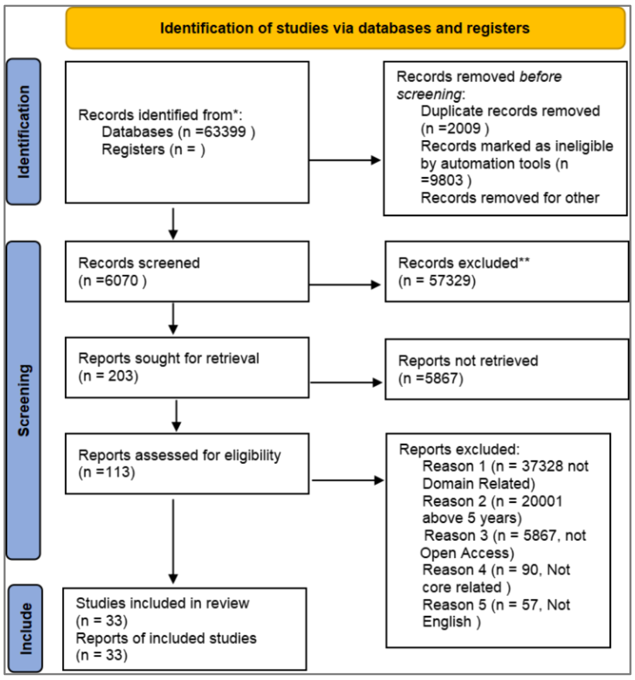
Identification of studies via databases and registers
Identification
Records identified from*: Databases (n =63399) Registers (n =)
Records removed *before screening*: Duplicate records removed (n =2009) Records marked as ineligible by automation tools (n =9803) Records removed for other
Screening
Records screened (n =6070)
Records excluded** (n = 57329)
Reports sought for retrieval (n = 203)
Reports not retrieved (n =5867)
Reports assessed for eligibility (n =113)
Reports excluded: Reason 1 (n = 37328 not Domain Related) Reason 2 (n = 20001 above 5 years) Reason 3 (n = 5867, not Open Access) Reason 4 (n = 90, Not core related) Reason 5 (n = 57, Not English)
Include
Studies included in review (n = 33)
Reports of included studies (n = 33)

Figure 6: PRISMA Flow Diagram for SLR Of an Efficient Neural Architecture Search Algorithm for Autoencoder Optimization (PRISMA 2020).

Eligibility criteria

Inclusion criteria: The inclusion criteria used for selecting the research papers include journal papers and conference proceedings, which were published from 2019 to 2024. Papers written in the English language and papers related to autoencoder neural networks, neural search architectures, and optimized algorithms were all included in the search, additionally, in instances where papers with identical studies and outcomes were identified, the most recent paper was selected. For some journal databases, the open access papers search option was used.

Exclusion criteria: Research Papers that were excluded in this study included papers that were written in languages other excluding English. We likewise excluded papers that were not related to neural search architecture, algorithm optimization, and papers whose contributions to the work are not explicitly stated in the abstract.

Synthesis and Analysis

The retrieved data were scrutinized to detect recurring patterns, trends, and themes related to the use of Neural Search Algorithms to optimize Autoencoders. A comparative analysis was employed to examine the findings across studies, providing insights into current trends and future research directions.

Feature examination

We Identified how each algorithm used in the autoencoder variation utilizes the bottleneck, weights (ReLu Function, Tan Sigmoid Function) and biases to achieve an improved model for accepting inputs from the encoder in low dimension, then passing it through the latent representation and finally decoding the input back to the output which is very close with little reconstruction error when compared with the original input data.

Algorithm evaluation

We collected experimental results from optimized algorithms in existing studies, compared their performances based on accuracy, efficiency, and computational complexity using a known dataset.

Identification of Gaps

The literature analysis revealed several gaps and areas that needed further investigation, particularly in the optimization of autoencoders using neural search algorithms. These gaps highlight potential opportunities for future research.

RESULTS AND DISCUSSION

Autoencoder Architecture

According to (Bank et al., 2020), an autoencoder is a specific type of neural network designed primarily to encode data into a meaningful and compact representation and then decode it back so that the reconstructed input is as similar to the original as feasible.

The backpropagation implementation serves as the foundation for the learning algorithm (Shrestha & Mahmood, 2019). The most basic AE, when there is only one hidden layer, is composed of two weight matrices, W and W', and two bias vectors, b and b' (Pulgar et al., 2020).

A neural network that has been trained to replicate its input is called an autoencoder, according to Bank et al. (2020) Their primary goal is to learn an "informative" representation of the data in an unsupervised manner that can be used for clustering and other purposes.

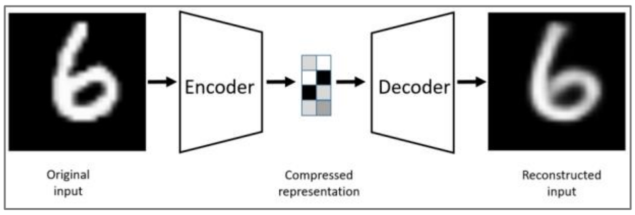

Figure. 1: Example of an autoencoder example (Bank et al., 2020).

Aamir et al. (2021) defined autoencoders (AEs) as an unsupervised neural network that applies back propagation behaviour, setting up the high-dimensional input feature set into a low-dimensional output feature set and then recovering the original feature set from the output. Wu et al. (2024) described autoencoders as an unsupervised learning technique which have been frequently used for feature extraction and representation learning in the vector-space problem. Mai & Henderson (2021) opined that text autoencoders are often used for unsupervised conditional text generation by applying mappings in the latent space to change attributes to the desired values. However, (Charte et al., 2023)described autoencoders as a specific type of unsupervised symmetrical neural network aimed to build a coding that maximizes the reconstruction of data patterns. Lazebnik & Simon-Keren (2023) defined AEs as a type of neural network that learn to encode and decode data in an unsupervised manner.

Autoencoder can be employed in computer vision, climate modeling, and physical systems (Seidman et al., 2023), visualization, anomaly detection, hashing, and noise removal (Charte et al., 2023; Aamir et al., 2021; Baier et al., 2023), electricity theft detection (Lazebnik & Simon-Keren, 2023), recommender systems (Xia et al., 2022; Ye et al., 2023), and natural language processing (NLP) (Felhi, 2023).

There exist different types and models of autoencoders in used today, such as sparse autoencoders (SAEs), denoising autoencoders (DAEs), laplacian regularized autoencoder (LAE), coupled deep autoencoder (CDA), hessian regularized sparse autoencoders (HSAE), nonnegativity constraints autoencoders (NCAE), multimodal deep auto-encoder (MDA), and Bayesian auto-encoder (BAE), contractive auto-encoder (CAEs), (Aamir et al., 2021). Also, we have the convolutional autoencoders, denoising autoencoders, variational autoencoders, polytopic autoencoders(Heiland & Kim, 2024), and variational autoencoders (Felhi, 2023; Seidman et al., 2023; Aliyu et al., 2025).

Different researchers have designed and deployed various methodologies in designing and implementing autoencoder models in the past. Aamir et al. (2021) used feature reduction by employing three parallel contractive auto encoders (CAEs) with all being arranged in layers (inside each layer, the process of encoding and decoding takes place) and trained in a feedforward nature for minimizing the objective function in order to minimize the reconstruction error while (Bunker et al., 2024) employed a *Visual Geometry Group* (VGG)-inspired methodology by gradually contracting/expanding the feature map while increasing/decreasing the channel dimensions with the models using Gaussian random positional encodings with k = 16. Wu et al. (2024) proposed a nonlinear representation of functional data using neural network autoencoders designed to process data in the form that it is usually collected without the need for preprocessing, whereas Mai & Henderson, (2021) in their model, encoded the text into a variable-size bag of vectors that grows with the size of the text, as in attention-based models. It allows it to encode and reconstruct much longer texts than standard autoencoders using vector mapping and a single-vector loss. Heiland & Kim (2024) proposed a polytopic autoencoder architecture that includes a lightweight nonlinear encoder, a convex combination decoder, and a smooth clustering network in their method while (Charte et al., 2023) applied an automated autoencoder architecture search procedure, based on evolutionary (strategy) methods with focus on different amounts of layers and units, individual activation functions per layer and several loss functions in their model and (Lazebnik & Simon-Keren, 2023) applied a method for Knowledge-integrated AutoEncoder (KiAE), which is constructed from two components: a partial distance regressor and an Long Short-Term Memory (LSTM)-based AE with seven fully connected, size adaptive, layers (three as part of the encoder, three as part of the decoder, and one as a representation layer) with a training phase and an inference phase. Baier et al. (2023) presented a new self-supervised method that combines the benefits of Siamese architectures and denoising autoencoders, which could compensate for the shortcomings of the individual components with a denoising autoencoder that is trained on a single corrupted view of the data, while Siamese networks use two views. Ye et al. (2023) proposed a lightweight and principled graph masked autoencoder model (named MAERec) to automate the self-supervised augmentation process. It adaptively and dynamically distills informative signals for reconstruction in response to changing sequential recommendation environments. Felhi (2023)developed a method that enhanced the interpretability of recent representation learning techniques, while accounting for the unavailability of annotated data, while Seidman et al. (2023) provided a novel, rigorous mathematical formulation of the variational objective in function spaces for training. Variational Autoencoding Neural Operators (VANO) first maps an input function to a distribution over a latent space using a parametric encoder and then decodes a sample from the latent distribution to reconstruct the input, as in classic variational autoencoders. Xia et al. (2022), on the other hand, proposed a network architecture of CRANet, which is formed as an integrative structure with a reflective receptor network and an information fusion autoencoder module, which endows its recommendation framework with the ability of encoding implicit user's pairwise preferences on both interacted and non-interacted items. Additionally, a parametric regularization-based tied-weight scheme is designed to perform robust joint training of the two-stage CRANet model.

Aamir et al. (2021), Charte et al. (2023), and Baier et al. (2023) used CIFAR and MNIST datasets to evaluate the performance, efficiency, and accuracy of their proposed autoencoder models, while Bunker et al. (2024) and Heiland & Kim (2024) used differential equations (Navier–Stokes equations) datasets with different viscosities to test the validity of their designed methods. Mai & Henderson (2021), Xia et al. (2022), and Felhi (2023) evaluated their methods using the Yelp dataset (for sentence sentiment transfer). Lazebnik & Simon-Keren (2023)used the economic dataset (which contains pricing and other properties), physics dataset (which contains different scales and ratios used to describe the mechanism of spherical particles settling in the air while experiencing aerodynamic drag), and biological dataset (containing genome sequences) to evaluate the performance of their proposed method whereas (Ye et al., 2023) employed the Amazon Books, Amazon Toys, and Retailrocket datasets (i.e users interactions with items in the categories of books and toys) to evaluate the performance and accuracy of their proposed autoencoder. Xia et al. (2022) evaluated their method using the MovieLens dataset (ML-1M, ML-10M), Netflix, and the Foursquare Checkin dataset.

In their research work, (Aamir et al., 2021) results revealed that the proposed model outperform DAE, RBM, SCAE, ScatNet and PCANet models in terms of training error, testing error and execution time and they proposed future research work on the evaluation of more complex datasets, such as: Toronto Face Detection (TFD) and Canadian Institute For Advanced Research (CIFAR) while (Bank et al., 2020) made significant contribution by complementing their training objective with architectures that permit discretization on any mesh (even irregular, non-grid meshes) and proposed a self-supervised training scheme exploiting the ability to evaluate the encoder and decoder on distinct meshes. They anticipate particular benefit in the use of more sophisticated generative models on the latent space, for example, diffusion models, analogous to stable diffusion in future methods. Mai & Henderson (2021) extended conditional text generation methods from single-vector bottleneck AEs to Bag-of-Vector Autoencoders (BoV-AEs), which encode text into a variable-size representation where the number of vectors grows with the length of the text. Heiland & Kim (2024) results of the single cylinder case indicated that the model well outperforms Proper Orthogonal Decomposition (POD) in terms of reconstruction and they anticipate improved methods in handling more challenging dynamics of the double cylinder due to the reconstruction deterioration in the extrapolation regime of their method while (Charte et al., 2023)results show the ability of their approach to find better architectures and able to concentrate most of the useful information in a minimized coding and in a reduced time. In their research work, Lazebnik & Simon-Keren (2023) results demonstrated that the KiAE model effectively captures the underlying structures and relationships between the input data and external knowledge, implying it generates a more useful representation. They identified areas for improvement in finding the optimal representation dimension for a given dataset, leveraging previous experience with the meta-learning approach, and further enhancing the investigation of how KiAE can be extended to incorporate unlabeled data. Baier et al. (2023) SidAE model shows results that are more stable regarding initialization seeds and the amount of time used for pre-training epochs, while Ye et al. (2023) results show that their model, Collaborative Reflection-Augmented Autoencoder Network (CRANet), was capable of exploring transferable knowledge from observed and unobserved user-item interactions, thereby handling missing data. Their research work identified the need for stability in SSL training by generalizing the approach to out-of-distribution sequences, which will help address the data distribution shift between training and test data in sequential recommendation, and generalize the model to newly arrived item sequences in the future. Felhi (2023) results identify two unnecessary components in the functioning scheme of Semi-Supervised VAEs, namely the Kullback-Leibler Divergence and the unobserved latent variable, proving indeed they were of no use to the semi-supervised VAE framework, and removing them speeds up computations, whereas Seidman et al. (2023) results show that their method takes a fraction of the training time and model size compared to competing approaches, and improved performance in terms of reconstruction error and sample generation quality.

Popov et al. (2022) in their research study aimed to improve the performance and efficiency of autoencoders for non-linear dimensionality reduction tasks by leveraging a meta-learning framework. The study employs a meta-learning approach where the autoencoder architecture and its hyperparameters are optimized through a meta-learner. This involves training the autoencoder on multiple tasks or datasets, allowing it to learn transferable knowledge that can be applied to new unseen tasks. The methodology includes sampling various tasks to create a diverse training environment. Each task is designed to simulate different types of non-linear relationships and data distributions, ensuring the model learns to generalize effectively across different scenarios.

Some identified limitations in the study include Limited Adaptability of Existing Models, Generalization Issues, Insufficient Exploration of Architectures, and Data Efficiency.

The key contributions include presenting a Meta-learning Framework for Autoencoders. This contribution addresses a critical limitation of conventional autoencoders, making the model more robust in real-world applications where data variability is high.

Venkataraman (2022) in his research paper, aimed to explore and demonstrate the effectiveness of Convolutional Autoencoders (CAEs) for removing noise from images. The primary goal is to investigate the application of CAEs for image denoising tasks. This involves adapting the CAE architecture and training process to effectively eliminate noise from images while conserving important image features. The study addresses the problem of reducing noise in digital images using deep learning techniques, specifically, convolutional autoencoders. This was achieved by modeling the architecture design using convolutional layers, pooling, and upsampling, as well as the implementation of ReLU activation functions.

This study's primary contributions are as follows: 1. Convolutional Autoencoder Architecture Development for Denoising 2. Improved Image Quality with Detail Preservation 3. Demonstration of Generalizability Across Noise Types and Levels.

The key research gaps are: The model struggles to adapt to the complex and diverse noise patterns found in real-world images, particularly in fields where noise varies significantly (e.g., medical imaging or remote sensing), and difficulty in Preserving Fine Details and Image Structures.

Neural Search Architecture Overview

Neural networks are difficult and slow to train, as shown by Bello et al., (2017) According to (Shrestha & Mahmood, 2019), the human nervous system and brain structure serve as models and inspirations for the machine learning (ML) technology known as a neural network. There are input, hidden, and output layers for the processing units. Neural network implementation involves the following steps: 1. Obtain a set of training and testing data. 2. Get the network trained 3. Use test results to make predictions.

Multiple layers of nodes make up a deep neural network (Shrestha & Mahmood, 2019). Various architectures, such as the Convolution Neural Network, Autoencoder, Restricted Boltzmann Machine (RBM), and Long Short-Term Memory (LSTM), have been designed to address issues in various fields or use cases.

Neural networks can be used to solve a variety of problems, including pattern recognition, classification, clustering, dimensionality reduction, computer vision, natural language processing (NLP), regression, and predictive analysis (Shrestha & Mahmood, 2019).

The primary objective of any NAS is to automate finding the best architecture that will produce an appreciable high predictive performance on given or unseen data (Heuillet et al., 2023). The NAS methods can be categorized into three components: 1. Search strategy 2. Search space 3. Performance estimation strategy. Figure 2 below shows the visual summary.

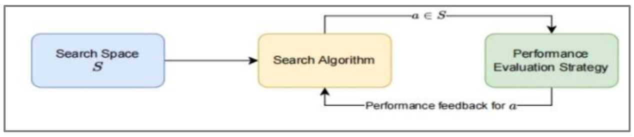

Figure. 2. A Neural Architecture Search framework's typical layout (Heuillet et al., 2023).

The different methods used for search strategies in NAS are Evolutionary Algorithms, Bayesian optimization, Gradient-based methods, and Reinforcement Learning (RL). However, most of these methods rely on huge computational resources and are unable to generalize across tasks (data sets).

Categories of Neural Architecture Search

Tabular Methods

The earliest design of neural network automation techniques was heuristic-based strategies that handled tabular input. Random search, tree-based sampling, regular grid sampling of the search space, and numerous additional techniques influenced by optimization are examples of such techniques. Even though ML-based techniques (see the ensuing subsections) and Differentiable NAS in particular have swiftly overtaken them, these techniques are still used in the NAS space because several have been proposed in recent years(Shrestha & Mahmood, 2019).

Learning via Reinforcement

By interacting with its environment in a way that maximizes cumulative rewards, an agent develops decision-making abilities under the machine learning technique known as reinforcement learning (RL). Unlike supervised learning, where labeled data guides learning, RL relies on trial and error, using a system of rewards and punishments to refine its strategy over time as depicted in Figure 3 (Wistuba et al., 2019).

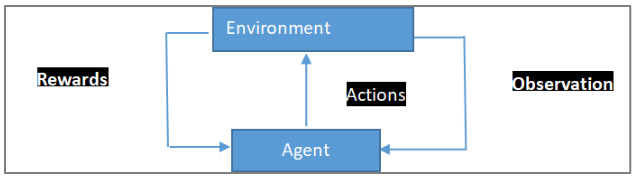

Figure 3: A general framework for reinforcement learning algorithms (Wistuba et al., 2019).

Zhang et al. (2018) in their research work aimed to revolutionize Neural Architecture Search (NAS) by introducing Graph HyperNetworks (GHNs) to significantly speed up the search process. The goal of this research is to enhance the efficacy and efficiency of Neural Architecture Search (NAS). The research was achieved by modeling neural architectures as graphs and predicting their performance without exhaustive training, thereby reducing computational costs while achieving competitive or superior model performance.

The key steps applied, from data preparation to model evaluation, taken by the researchers include: 1. Data Collection and Representation 2. Graph HyperNetwork Design 3. Training the Graph HyperNetwork: 4. Search Strategy and Evaluation of NAS. 5. Evaluation Metrics by Prediction Accuracy and Search Efficiency. 6. Baseline Comparisons:

Some research gaps identified in this approach include: 1. Limited Exploration of Scalability to Larger Search Spaces. 2. Generalization Across Diverse Architecture Types.

The Study provides several valuable contributions to the field of NAS and neural network optimization, which include: 1. Introduction of Graph HyperNetworks for Efficient NAS, an innovative approach to modeling neural architectures as graphs, enabling performance predictions without fully training each candidate model. 2. Graph-Based Representation of Neural Architectures. 3. Development of a Performance Prediction Framework to forecast neural networks' performance based on their graph representations. 4. Reduction in Computational Costs of NAS by eliminating the need for exhaustive training.

Self-Observed Representation Search for Evolutionary Neural Architectures via Learning

“Self-supervised Representation Learning for Evolutionary Neural Architecture Search" by Chen Wei demonstrates how optimizing the evolutionary neural architecture (NAS) search by integrating self-supervised representation learning process can significantly improve the efficiency, generalization, and scalability of NAS methods (Wei et al., 2020) By leveraging self-supervised learning, the model can better capture essential patterns and structures in the architecture space without needing extensive labeled data. Self-supervised learning improves the exploration-exploitation tradeoff in evolutionary NAS.

One of the primary contributions of this work is the introduction of self-supervised learning (SSL) into the process of evolutionary NAS, which can effectively learn useful representations of architectures without requiring labeled data, and these representations are then used to guide the evolutionary NAS.

The limitations identified revolve around computational inefficiency, generalization limitations, representation inefficiency, and scalability issues in traditional NAS methods.

Masked Autoencoders

The robustness of masked autoencoder Learners, which use Neural Architecture Search, can be effectively utilized as a powerful and robust approach for neural architecture search (NAS) (Hu et al., 2023). The technique of neural architecture search (NAS) is computationally costly, which involves exploring a vast search space of potential network architectures to find the best one for a given task. Traditional NAS methods often suffer from high computational costs and can be sensitive to hyperparameter choices. The researchers aimed to develop a more efficient and robust NAS method that can effectively explore the architecture search space while minimizing computational overhead (Hu et al., 2023).

Some identified gaps in this research are: Search Space Exploration, Evaluation Efficiency, Computational Cost, Uncertainty Quantification and Interpretability, Understanding Learned Representations (explaining how the MAE makes decisions can improve transparency and trust in the generated architectures), and Scalability, such as Handling large and complex search spaces, can be challenging.

Multiscale Autoencoders for Graph Neural Networks for Interpretable Scientific Artificial Intelligence (AI)

Barwey et al. (2023) aimed to develop a novel approach for constructing interpretable and efficient reduced-order models (ROMs) for complex scientific simulations. This approach aims to address the specific challenges posed by complex, structured, and multiscale data commonly encountered in scientific domains. The model is built on a Graph Neural Network (GNN) layer backbone to encode graph-structured data effectively. The MGNN-AE architecture incorporates multiscale learning mechanisms by creating hierarchical representations that reflect different spatial or temporal scales present in the scientific data. The model uses an autoencoder framework. This helps in reducing dimensionality while preserving important structural information.

The primary contributions are as follows: Development of the Multiscale Graph Neural Network Autoencoder (MGNN-AE), Improvement in Interpretability for Scientific Machine Learning, Integration of Multiscale Learning Mechanisms for Hierarchical Data.

Some of the identified limitations include: Lack of Interpretability in Scientific Machine Learning Models, Challenges in Capturing Multiscale Structures in Scientific Data, Insufficient Integration of Domain-Specific Knowledge with Machine Learning Models, and Limited Evaluation Metrics for Interpretable Representations in Scientific Contexts.

Chen et al. (2020) in their research work "DRNAS: Dirichlet Neural Architecture Search" aimed to create a new, effective Neural Architecture Search (NAS) technique based on the Dirichlet distribution. They aimed to leverage the properties of the Dirichlet distribution to effectively explore the space of possible neural architectures. The Dirichlet distribution is a probability distribution over a simplex, making it suitable for representing the probabilities of different architectural choices.

The architecture space is treated as a probabilistic space, where each candidate architecture is drawn from a Dirichlet distribution, allowing the search process to explore new architectures while refining the distribution based on prior search results using an an optimization Search Strategy.

The key contribution of the paper is the introduction of Dirichlet Neural Architecture Search (DRNAS) as a novel framework and Probabilistic Search Framework.

While DrNAS demonstrates promising results, it is essential to consider potential limitations, such as: 1. Computational Complexity. 2. Hyperparameter Sensitivity. 3. Exploration vs. Exploitation Balance. 4. It is still necessary to fully examine the applicability to a variety of tasks, including reinforcement learning and natural language processing.

Algorithm Architecture

According to (Dehghani et al., 2020), an algorithm is deemed intelligent if it can efficiently solve a problem with optimization as its primary feature using the least amount of information and in the quickest period of time.

According to a more thorough description, the heuristic technique is a tactic that forgoes some knowledge in order to arrive at a solution as quickly and accurately as possible. Natural processes, such as biological processes or rules that explain physical phenomena, are usually the basis for heuristic algorithms. These algorithms have been divided into a number of groups, including physics-based, evolution-based, and swarm-based algorithms (Dehghani et al., 2020). The figure 4 below summarizes the meta-heuristic algorithms.

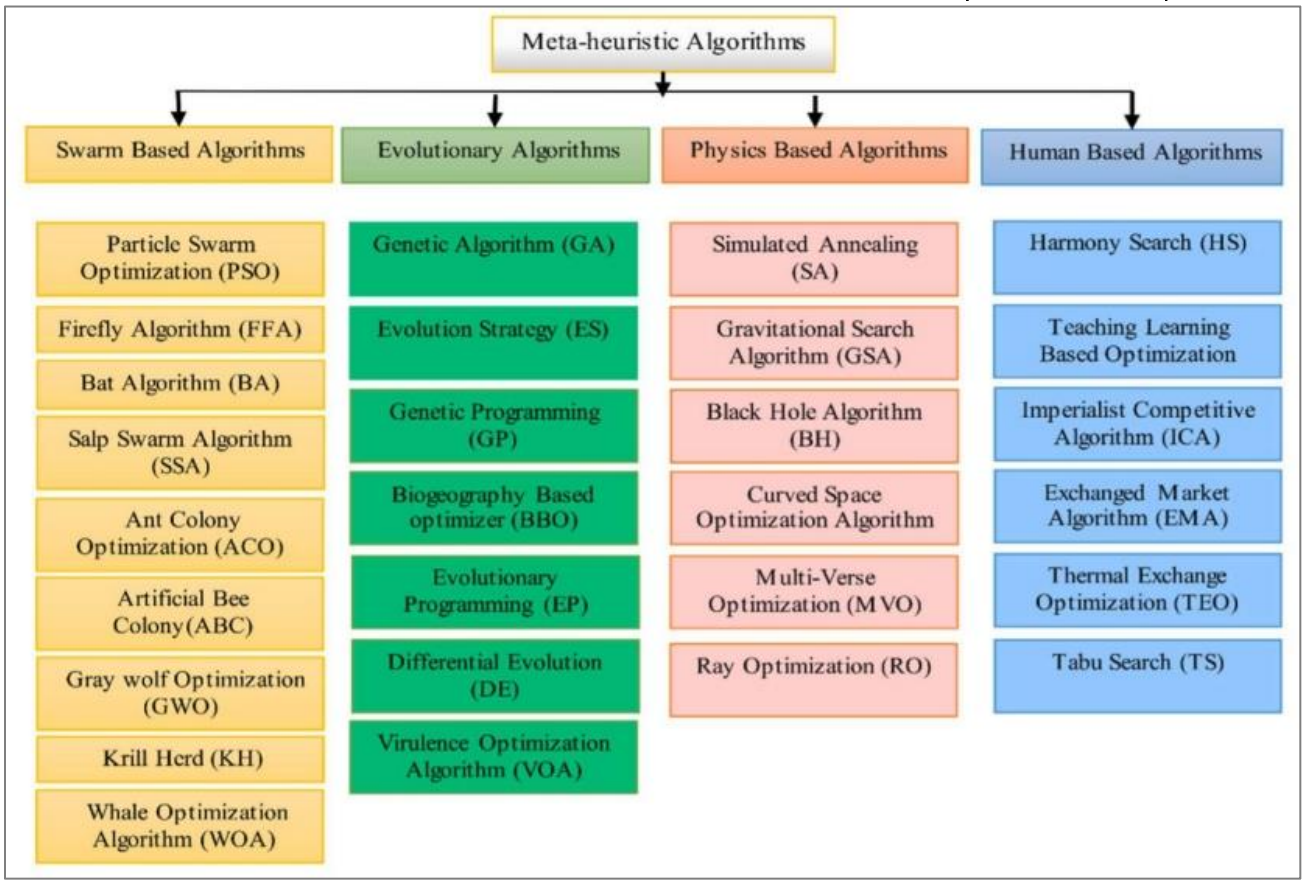
Meta-heuristic Algorithms
Swarm Based Algorithms
Evolutionary Algorithms
Physics Based Algorithms
Human Based Algorithms
Particle Swarm Optimization (PSO)
Firefly Algorithm (FFA)
Bat Algorithm (BA)
Salp Swarm Algorithm (SSA)
Ant Colony Optimization (ACO)
Artificial Bee Colony(ABC)
Gray wolf Optimization (GWO)
Krill Herd (KH)
Whale Optimization Algorithm (WOA)
Genetic Algorithm (GA)
Evolution Strategy (ES)
Genetic Programming (GP)
Biogeography Based optimizer (BBO)
Evolutionary Programming (EP)
Differential Evolution (DE)
Virulence Optimization Algorithm (VOA)
Simulated Annealing (SA)
Gravitational Search Algorithm (GSA)
Black Hole Algorithm (BH)
Curved Space Optimization Algorithm
Multi-Verse Optimization (MVO)
Ray Optimization (RO)
Harmony Search (HS)
Teaching Learning Based Optimization
Imperialist Competitive Algorithm (ICA)
Exchanged Market Algorithm (EMA)
Thermal Exchange Optimization (TEO)
Tabu Search (TS)

Figure 4. Categorization of meta-heuristic algorithms (Noroozi et al., 2022).

Finding the ideal weight vector values to address a class of problems in an area is the aim of the learning algorithm. Several well-known training algorithms include the Levenberg-Marquardt algorithm, Momentum, Gradient Descent, Stochastic Gradient Descent, and Backpropagation through time.

Evolutionary Algorithms

An evolutionary algorithm is one of the training aspects of Algorithms. Evolutionary algorithms (EAs) offer a viable, efficient, and flexible framework for Neural Architecture Search (NAS) that can outperform or complement traditional NAS approaches in terms of adaptability, diversity, and scalability (Liu et al., 2020). Evolutionary algorithms offer significant advantages for NAS, and systematic evaluation and synthesis of existing approaches can drive innovation in this domain. Evolutionary algorithms, inspired by natural selection, offer a robust and flexible search mechanism that can effectively navigate the vast and complex search space of neural architectures, striking a better balance between exploration and exploitation than traditional methods. The following are crucial elements of these population-based global optimizers for the following black-box functions: survivor selection, recombination and mutation, parent selection, and initialization (Wistuba et al., 2019).

The initialization specifies how the population's first generation is produced. Following initialization, the optimizer iterates through the subsequent phases until it terminates. (See below, Figure 5):

1. Choose parents for reproduction from the population.
2. To produce new individuals, use recombination and mutation processes.
3. Assess the new hires' level of fitness.
4. Select the survivors of the population.

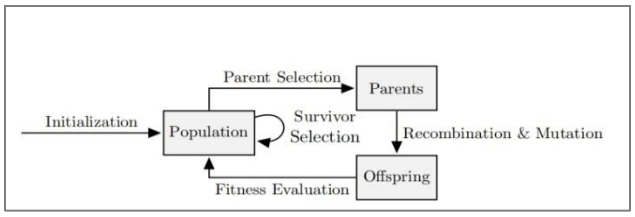

Figure 5: A general framework for evolutionary algorithms (Wistuba et al., 2019).

Some advantages of Evolutionary Algorithms include:

1. Evolutionary Algorithms (EAs) are Well-Suited for NAS as EAs naturally support operations like mutation and crossover, which allow them to discover innovative architectures that may not be found by gradient-based or reinforcement learning-based NAS methods.

2. EAs Enable Diverse and High-Performing Architectures through the stochastic and population-based nature of evolutionary algorithms.

3. EAs Can Be Computationally Competitive with Efficient Strategies by integrating techniques such as surrogate modeling, parallelism, and hybrid strategies.

Some gaps identified with EAs include the lack of standard benchmarks and evaluation protocols, which provide actionable insights to guide future development in the field.

CONCLUSION

In this study, we reviewed various autoencoders, their variations, and architectures, and also explored the neural architecture search and its categories, including tabular-based approaches, Reinforcement Learning, Evolutionary Algorithms, and Gradient Descent and Differentiability, along with their limitations and contributions. Algorithms and their structure were also reviewed, with some of the training models explored, such as the Levenberg-Marquardt algorithm, momentum, gradient descent, stochastic gradient descent, and backpropagation through time, with a variety of datasets utilized to verify the outcomes of various proposed models and methods employed by various researchers and highlighting key findings and future work of the various research papers reviewed.

RECOMMENDATIONS

With machine learning constantly evolving and a growing emphasis on deep learning, future work should focus on developing a practical model of an autoencoder with an improved neural architecture search using an evolutionary algorithm to solve real-life complex problems without supervision or human interference.

REFERENCES

Aamir, M., Mohd Nawi, N., Wahid, F., & Mahdin, H. (2021). A deep contractive autoencoder for solving multiclass classification problems. *Evolutionary Intelligence, 14*(4), 1619–1633.[crossref]

Aliyu, A. A., Ibrahim, M., & Abdulkadir, S. (2025). A Blockchain Enhanced Deep Learning Approach for Intrusion Detection in Trusted Execution Environments. *Digital Technologies Research and Applications, 4*(1), 135–157. [crossref]

Baier, F., Mair, S., & Fadel, S. G. (2023). *Self-Supervised Siamese Autoencoders.* [crossref]

Bank, D., Koenigstein, N., & Giryes, R. (2020). *Autoencoders.*[crossref]

Barwey, S., Shankar, V., Viswanathan, V., & Maulik, R. (2023). *Multiscale Graph Neural Network Autoencoders for Interpretable Scientific Machine Learning.*[crossref]

Bello, I., Zoph, B., Vasudevan, V., & Le, Q. V. (2017). *Neural Optimizer Search with Reinforcement Learning.*

Bunker, J., Girolami, M., Lambley, H., Stuart, A. M., & Sullivan, T. J. (2024). *Autoencoders in Function Space.* [crossref]

Charte, D., Charte, F., del Jesus, M. J., & Herrera, F. (2020). A Showcase of the Use of Autoencoders in Feature Learning Applications. In *Advances in Intelligent Systems and Computing* (Vol. 1000, pp. 445–456). Springer. [crossref]

Charte, D., Charte, F., García, S., del Jesus, M. J., & Herrera, F. (2018). A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion, 41*, 37–52. [crossref]

Charte, D., Charte, F., García, S., & Herrera, F. (2019). A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence, 8*(1), 1–14. [crossref]

Charte, F., Rivera, A. J., Martínez, F., & del Jesus, M. J. (2023). EvoAAA: An evolutionary methodology for automated neural autoencoder architecture search. *Integrated Computer-Aided Engineering, 30*(1), 85–102. [crossref]

Chen, X., Wang, R., Cheng, M., Tang, X., & Hsieh, C.-J. (2020). *DrNAS: Dirichlet Neural Architecture Search.* [crossref]

Dehghani, M., Montazeri, Z., Dhiman, G., Malik, O. P., Morales-Menendez, R., Ramirez-Mendoza, R. A., Dehghani, A., Guerrero, J. M., & Parra-Arroyo, L. (2020). A spring search algorithm applied to engineering optimization problems. *Applied Sciences, 10*(18), 6173. [crossref]

Felhi, G. (2023). *Interpretable Sentence Representation with Variational Autoencoders and Attention.* [crossref]

Greenacre, M., Groenen, P. J. F., Hastie, T., Iodice D'Enza, A., Markos, A., & Tuzhilina, E. (2023). Principal component analysis. *Nature Reviews Methods Primers, 2*(1), 100. [crossref]

Heiland, J., & Kim, Y. (2024). *Polytopic Autoencoders with Smooth Clustering for Reduced-order Modelling of Flows.* [crossref]

Heuillet, A., Nasser, A., Arioui, H., & Tabia, H. (2023). *Efficient Automation of Neural Network Design: A Survey on Differentiable Neural Architecture Search.* [crossref]

Hu, Y., Chu, X., & Zhang, B. (2023a). *Masked Autoencoders Are Robust Neural Architecture Search Learners.* [crossref]

Hu, Y., Chu, X., & Zhang, B. (2023b). *Masked Autoencoders Are Robust Neural Architecture Search Learners.* [crossref]

Lazebnik, T., & Simon-Keren, L. (2023). Knowledge-integrated AutoEncoder Model. *Expert Systems with Applications, 249*, 124108. [crossref]

Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Tan, K. C. (2020). A Survey on Evolutionary Neural Architecture Search. *IEEE Transactions on Neural Networks and Learning Systems, 32*(5), 1949–1968. [crossref]

Mahesh, B. (2020). Machine Learning Algorithms - A Review. *International Journal of Science and Research, 9*(1), 381–386. [crossref]

Mai, F., & Henderson, J. (2021). *Bag-of-Vectors Autoencoders for Unsupervised Conditional Text Generation.* [crossref]

Noroozi, M., Mohammadi, H., Efatinasab, E., Lashgari, A., Eslami, M., & Khan, B. (2022). Golden Search Optimization Algorithm. *IEEE Access, 10*, 37515–37532. [crossref]

Popov, A. A., Sarshar, A., Chennault, A., & Sandu, A. (2022a). *A Meta-learning Formulation of the*

Autoencoder Problem for Non-linear Dimensionality Reduction. [crossref]

Popov, A. A., Sarshar, A., Chennault, A., & Sandu, A. (2022b). *A Meta-learning Formulation of the Autoencoder Problem for Non-linear Dimensionality Reduction*. [crossref]

PRISMA 2020 - Creating a PRISMA flow diagram - LibGuides at University of North Carolina at Chapel Hill. (n.d.). Retrieved March 7, 2025, from [crossref]

Pulgar, F. J., Charte, F., Rivera, A. J., & del Jesus, M. J. (2020). Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines. *Information Fusion, 54*, 44–60. [crossref]

Seidman, J. H., Kissas, G., Pappas, G. J., & Perdikaris, P. (2023). *Variational Autoencoding Neural Operators*. [crossref]

Shrestha, A., & Mahmood, A. (2019). Review of deep learning algorithms and architectures. *IEEE Access, 7*, 53040–53065. [crossref]

Singh, J., Azamfar, M., Li, F., & Lee, J. (2021). A systematic review of machine learning algorithms for prognostics and health management of rolling element bearings: fundamentals, concepts and applications. *Measurement Science and Technology, 32*(1), 012001. [crossref]

Venkataraman, P. (2022). *Image Denoising Using Convolutional Autoencoder*. [crossref]

Wei, C., Tang, Y., Niu, C., Hu, H., Wang, Y., & Liang, J. (2020). *Self-supervised Representation Learning for Evolutionary Neural Architecture Search*. [crossref]

Wistuba, M., Rawat, A., & Pedapati, T. (2019). *A Survey on Neural Architecture Search*. [crossref]

Wu, S., Beaulac, C., & Cao, J. (2024). *Functional Autoencoder for Smoothing and Representation Learning*. [crossref]

Xia, L., Huang, C., Xu, Y., Xu, H., Li, X., & Zhang, W. (2022). Collaborative Reflection-Augmented Autoencoder Network for Recommender Systems. *ACM Transactions on Information Systems, 40*(1), 1–28. [crossref]

Ye, Y., Xia, L., & Huang, C. (2023). Graph Masked Autoencoder for Sequential Recommendation. In *SIGIR 2023 - Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 321–330). [crossref]

Zhang, C., Ren, M., & Urtasun, R. (2018). *Graph HyperNetworks for Neural Architecture Search*. [crossref]