Evolving Neural Network Designs with Genetic Algorithms: Applications in Image Classification, NLP, and Reinforcement Learning

Rajesh Kumar Malviya , Enterprise Architect, rajeshkumarmalviya15@outlook.com
Sathiri machi, Quality systems Engineer, sathirimachi@yahoo.com
Ravi Kumar Vankayalapti, IT Architect, Equinix, ravikumar.vankayalapti.research@gmail.com
Lakshminarayana Reddy Kothapalli Sondinti, Senior Software Engineer, US bank lakshminarayana.k.s.usbank@gmail.com

Abstract

A method of evolving deep learning architectures using genetic algorithms is presented. The method is a first step towards a low-cost evolutionary search for task-specific neural networks. We evolve task-specific model architectures optimized for fast execution and low error on several standard machine learning tasks: image classification, character-level language modeling, and solving the cart pole problem. We also introduce a simple variation of the method that is capable of evolving neural networks with recurrent connections of varying depth and length and show performance on a word-level language modeling task. The method is implemented in an open-source library. We hope that the ability to run an evolutionary search at this scale will make it possible for a wide audience to develop deep learning architectures that are specialized for a variety of tasks and to develop many interesting novel architectural features.
A new method that uses evolutionary search to directly modify existing neural network architectures to perform a specific task is presented. We demonstrate that task-specific specialization of deep learning models can be useful in practice. We modify convolutional neural networks, residual networks, and an LSTM variant to perform various tasks, and show that specialized networks often perform better than models trained from scratch that have many more parameters and much larger training time. For example, on the object recognition task, a specialized model is built by training a base network to predict object position and then applying a series of genetic search operations to squeeze the network and fit new final layer weights to the output. The specialized model is 8 times faster and has 13% lower error, despite being 17 times smaller than a fully trained larger and slower network.

Keywords: Deep Learning, Genetic Algorithms, Neural Network Architectures, Task-Specific Models, Image Classification, Language Modeling, Evolutionary Search, Convolutional Neural Networks (CNNs), Reinforcement Learning, Performance Optimization.

1. Introduction

Artificial neural networks have strong potential to apply to a wide variety of complex and high-dimensional problems. However, neural networks are very difficult to design by human expertise or hand. Currently, neural network designs can achieve a high degree of performance for specific tasks as a result of a large increase in the volume and quality of data available for training, and tremendous improvements in performance due to being deeper networks, offering and requiring a higher number of trainable parameters. This performance, though, comes at the cost of a longer training time and a higher computational demand at runtime. Moreover, the process of designing deep neural networks remains manual at its core, generally guided by high computational trial-and-error portfolios combined with experience and knowledge from previous successful models. As such, the design process is costly in human labor, time, and computational resources.

1.1. Background and Motivation

Traditionally, designing neural networks to solve a particular problem has been a manual, trial-and-error process. Neural networks consist of layers of neurons, which contain a set of weights that determine the output of the neuron for given inputs. A neural network is created by specifying the number of layers, the number of neurons in each layer, and how the neurons are connected. Adjusting the weights is the core of the process, and this is typically done using an algorithm that minimizes the error over a training set. However, designing the network architecture itself is typically a manual process. Utilizing a method to automate this approach can lead to discovering superior neural network architectures.
In this study, we explore the use of genetic algorithms to evolve neural network architectures. The goal is to demonstrate that superior neural network designs can be found by evolving over a population of networks and selecting the fittest ones according to a fitness function. Additionally, we would like to avoid setting network hyperparameters in the form of trial and error. Since the 1980s, researchers have attempted to utilize genetic algorithms or similar optimization techniques to optimize important design components like network architecture, specific building blocks, and hyperparameters. Despite some success in smaller networks, a method that can effectively evolve larger networks has proven elusive. This contrasts with the advances in applying deep learning to various problems, resulting in little consideration for alternative approaches in designs. In recent years, however, there has been a rising interest in revisiting this

approach due to promising results in discovering novel neural network architectures.

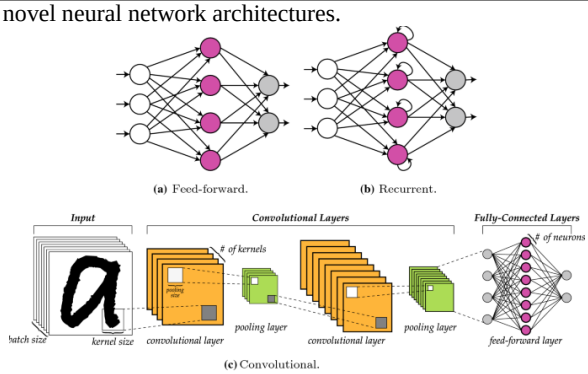

Fig 1 : Evolutionary Design of Neural Networks: Past, Present, and Future

1.2. Research Objectives

The main objective of this study is to use genetic algorithms to evolve novel architectures for neural networks that can improve performance across various domains, including areas such as image classification, natural language processing, and reinforcement learning. This study will explore the process of evolving multiple neural networks in parallel along with their connections, weights, and hyperparameters using backpropagation. The effect of neural network designs on model accuracy, convergence rate, and statistical efficiency in optimization performance will be evaluated.

Most studies have only aimed to address a single problem or have only used this approach for evolving a limited subset of features within the network, such as architectural features or hyperparameters. Another significant limitation of existing studies is their use of only a few benchmark datasets or simulation environments. These approaches thus cannot show when their proposed designs might fail on datasets with peculiar characteristics. For this study, a broad exploration of the effectiveness of evolved neural network designs will be performed on dozens of different kinds of datasets and have the most comprehensive benchmarking experiments to date. Hopefully, it proves to be a very big leap forward in the evolution of neural network design for the artificial intelligence community.

Equation 1 : Image Classification

$$F(N) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{I}(y_i = \hat{y}_i)$$

Equation for Fitness Evaluation:

Where : *F(N)* is the fitness score of the neural network *N*.

Ntest is the number of test images.

y_i is the true label of the *i* - th image.

y^i is the predicted label from the neural network.

$\mathbb{I}$ is the indicator function, which is 1 if the prediction is correct and 0 otherwise.

2. Neural Networks and Genetic Algorithms

Neural network models are so named because of their vague resemblance to the structure of the human brain. These machine-learning mechanisms have had a great impact since their creation. They can be used for solving problems related to data inversion, optimization, and pattern recognition. The network has at least an input layer and an output layer. It can have multiple layers, hence the name multilayer network. The name deep neural network is used to describe a multilayer neural network that has many layers (hence the depth). A layer is a set of n neurons that do not interfere with each other: they receive data from the previous layer, perform computation over that data while applying the same mathematical operation to all members, send the resulting data to the next layer, and wait for the next data inputs.

The operation performed by the neurons is fed linearly into a function known as the activation function. The function to be applied to the neurons in the same layer can be pattern-dependent. Hence, the expression w1 f1 + w2 f2 + ... + wn fn will represent a neuron j in layer i. w is a vector with n components, called weights, and f is the input. The idea is to adjust the values of w for the neuron to generate an output according to the output from the biological neuron in the human brain. Neural network training consists of obtaining the best value for the w vector so that the output generated by the neural function can solve a given problem. Training often occurs using a data set created specifically for this purpose and through the presentation of that set to the network. Neural networks need to be designed for a given task before they are trained. The design can be based on different mathematical and/or biological concepts.

2.1. Neural Network Architectures

As part of evolving neural network designs with genetic algorithms, the architectures of the networks themselves are to be designed. There are many aspects to consider when designing a neural network, such as output units, inputs, number of layers, activation functions, and so on. However, the particular task defines what the best architecture is. Hence, a process of finding optimum architectures for neural networks for a particular task that involves more than just trial and error on the part of the designer is a flexible, programmatic approach. This implies finding specific tasks for which AI segmentation can be useful, such that several varied architectures can be trained on the task across multiple GPUs. The parts of the task that each architecture was specifically set up to handle are such that no one model could handle the entire task. It has been observed that an intra-network genetic algorithm can also be quite effective at determining the

architecture of a trainable segment that sits atop a larger fixed segment.

Given that it is observed that a genetic algorithm that gives considerable neural connections assigned to be zero performs more poorly, even if more time is given for search, it is clear that overall network sizes should be reduced. However, despite the considerably reduced size, there is a similar improvement in test loss, test accuracy, and the time to reach the target loss level. There are general limitations that are mostly related to the way training is done. While the segments can receive different data augmentations, overall, the neural networks need to be pre-trained together before excelling on the real task. However, training on synthesized data can be helpful to get a good start; it is ultimately domain-specific networks that are best. The best way to get a good start is to have several very capable reusable segments waiting in the wings.

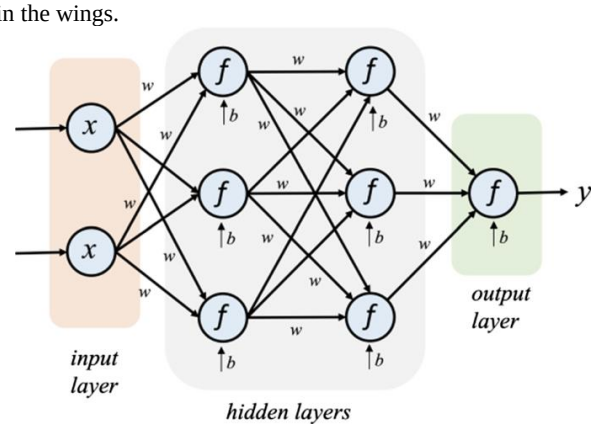

Fig 2 : Evolutionary Design of Neural Network Architectures

2.2. Genetic Algorithms

Genetic algorithms operate by generating a population of candidate designs (or genotypes), each of which is rated by a fitness function computing the quality of its resulting phenotype. By evaluating the ratings, high-quality individuals are selected to reproduce, transmitting portions of their design to produce offspring that will form the next generation's population. Genetic algorithms are very well-suited for parallel computation, as mutation and crossover operations can be implemented independently on many individuals simultaneously. They possess no insight into these designs beyond the ability to provide a fitness grade, which in this context is assigned by the neural network performing a specific task. Consequently, the search for the fittest individuals is conducted according to a 'blind' parallel strategy that operates without heuristic restriction on possible paths and which allows for non-intuitive design features to emerge. Genetic algorithms are not guaranteed to converge to a global maximum, as path-restricting heuristics are commonly utilized in evolutionary signal processing and backpropagation. The possibility of converging to local maxima is unlikely because individuals embodying large numbers of nodes, robust internal structures, and classifiers of high performance are capable of generating offspring of even greater performance. Differentiating between high- and low-performers depends only on the current machine learning model's ability to carry out the task.

3. Evolving Neural Network Designs

In this section, an overview detailing how designs of deep networks undergo evolution is provided. Image classification problems frequently involve a sequence of convolutional, pooling, dropout, and fully connected layers. While neural networks, fragmented in this form, have been analyzed sparingly until recently, they are different aspects of an entire network trained according to backpropagation. We take inspiration from nature and consider it wise to remix such dismembered networks via crossover. Crossover, in this context, may denote a merger that combines components of two distinct networks to result in a novel design solution. The act of producing approximations to such designs from sequences dictated by mutation and crossover is modeled as a genetic algorithm, backed by a neural architecture evaluation process. The approach is modular enough to cover such problem variants as optimization of network design for reinforcement learning, regression, or NLP tasks.

As various adaptation techniques have yielded more suited deep network designs, method advancement has turned somewhat toward fast-paced empirical exploration. Many recent techniques entail making incremental adjustments to the existing, trained network. The genetic algorithm (GA) differs in its approach to optimization. It starts with a mostly uninformed population from which there are successive culls followed by the birth of and repopulation with the next generation selected through the use of fitness scores pertinent to a task at hand. GAs are scalable, fully parallelizable, and exhibit uses in many problem areas. Not widely seen, the GA has been hitherto used to evolve deep network architectures but is now gaining in popularity.

3.1. Encoding Neural Network Structures

To create valid natural networks, there are a few constraints to consider. Firstly, the structure must be of fixed depth, which is an important practical consideration. On the one hand, some computational tractability is needed, and thus infinite or large depth cannot be allowed. On the other hand, if one increases the depth sufficiently, one may approximate many missing or hidden constants arbitrarily closely. By the universal approximation theorem, a feedforward artificial neural network with a nonconstant univariate continuous sigmoidal or piecewise linear function as an activation function can approximate any continuous function on a compact subset of the real line and also derive more efficient deep neural networks. For some learning problems, the structure of a deep feedforward artificial neural network should encode the most important empirical or learned characteristics that will be used to solve the problem. Since evolving the neural network on large training data takes a relatively long time, it is much more efficient to design the standard neural network and then transfer the learned model parameters. For a given structure, backpropagation can perform efficiently to transfer learned model parameters. In this research, we use backpropagation to transfer learned model parameters.

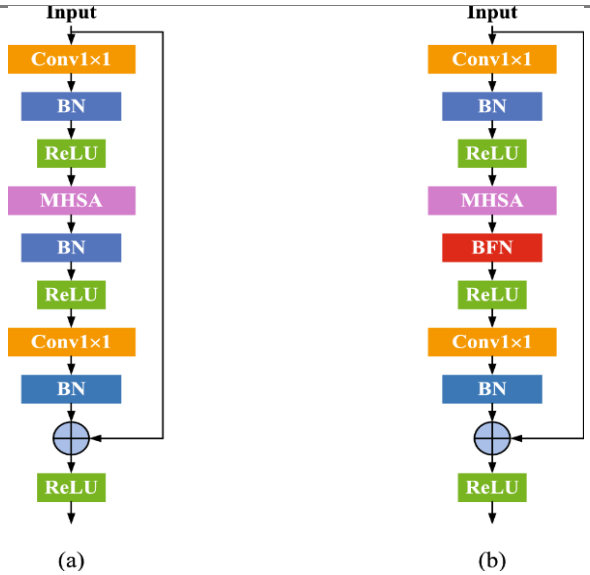

Fig 3 : Evolutionary Neural Architecture Search Combining multi-branch ConvNet and Improved Transformer

Then we classify a structure as a valid neural network design if: 1. The structure is of depth N, where N may be different for the three enclaves. 2. Where a given layer is an SFC, for any descendant SFC, the encoding should denote the actual ACF outputs. In particular, the predecessor action vector should include in its encoding the dimensions of the arrays encoding the models and feature vectors for this layer of the SFC. 3. All SFCs must be connected to some ACF. 4. For any given ACF, the encodings should only correspond to ACF operations.
For the neural networks, 1D Convolutional Neural Network could be considered as ACF, and Fully Connected Neural Network is a more typical example of ACF. Exceptionally, DenseNet should be used for the ACF of the second largest caterpillar that is composed of several trainable layers. The structure of a Concatenation of Deeply Supervised Residual Networks should be used almost identically for any encoding in the first enclave, but for all other enclaves, there will be no encoded ACF. In those cases, the encoding should just include Operation 1 on its last 4 parameters. The interconnection will perform connection operations with no ACF involved. Similarly, for output A, there will be no activation function to perform.

3.2. Fitness Evaluation and Selection

Each induced neural network is evaluated individually, and the selection is made through the genetic algorithm. Some parents may produce good offspring, which means that they might die to guarantee genetic diversity, while an individual with a low fitness may produce one with a high fitness. To address this issue, the selection method should give a low (but non-zero) probability for a poor-fitness individual to be chosen. One possible selection method is tournament selection. Given two different individuals chosen at random, the one with the higher fitness value is chosen to enter the next generation. This process is repeated until the next generation is fully completed.
By simply choosing the fitter individuals, we take advantage of their good attributes and minimize the impact of poor attributes. All individuals have a chance, but the fittest have a better one. This randomness introduces good properties from random sampling methods. Also, it is a simple algorithm that is easy to implement. These features make tournament selection an interesting choice to be used in this project. Another possible selection method is roulette selection. In this case, the probability for an individual to be chosen depends on the fitness of the individual. If the best individual has a non-null discrepancy value, the probability for that individual is given by: fitness(i) / sum(fitness), where fitness(i) is the fitness of the individual and sum(fitness) is the sum of all individuals' fitness.

4. Applications in Image Classification

Convolutional neural networks (CNNs) are designed to process data that have grid-like topology, such as speech or image data. Speech data, often represented with spectrograms, are frequency and time-dependent, while image data are two-dimensional. Normally, a CNN processes input data by taking the signal, convolving a set of filters across it, and producing an output signal called feature maps. The filters are determinant factors in building a CNN. Being manually designed, these filters have some hand-picked features and lack common sense in solving complex problems. In effect, the applications of CNNs are limited because finding a set of proper filters calls for great expertise. We strive to automatically design CNNs with genetic algorithms and verify their effectiveness. We tested these automatically found CNN designs on ten image classification tasks. Experimental results reveal a significant outperformance of our algorithm compared to traditional CNN designs. Furthermore, our designs demonstrate fast and stable convergence.
Another application of modern CNNs is the stacked generalization technique. We utilize one level of inductive transfer for successfully utilized feature extractors to enhance the diversity. Instead of boosting transformations or pooling, we design and verify generic variations with convolution layers combined with empirically validated rectified linear activation functions. The objective of this work is to demonstrate that parameter-less feature extractors can be designed in an automated and efficient architecture even for sparse and rescaled images with interesting applications in biologically inspired deep learning, face analysis, and video surveillance. In our study, we demonstrate the performance of evolved feature extractors tested with ten widely utilized image classification benchmarks. Our outcome is that designed parameter-less feature extractors using unsupervised inductive transfer outperform related state-of-the-art methods. Our approach can help practitioners save time and effort while designing feature extractors for customized image classification tasks.

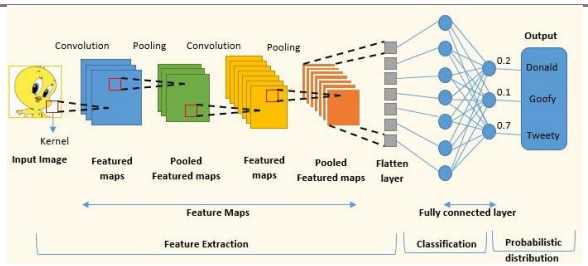

Fig 4 : Convolutional Neural Networks (CNN)

4.1. Current Challenges

The current issues regarding evolving neural networks with genetic algorithms mainly revolve around genetic encoding, decoding, and the ability to reach the desired output. There is much debate in the community about which encoding schemas are best and how to approach this process. Specifically, when understanding genetic algorithms and utilizing them to evolve neural networks, opening the parameter space can be quite challenging. With linear encoding, adding a few more connections or nodes and/or layers to the network quickly results in probabilistic ambiguity or distortion. It is difficult to explore across many varied alterations, and parameters accrue. Excessive complexity can easily destroy good solutions, even when parts of it are redundant and the complexity is unwanted.

Also, depending on the size of the genome, many parameters are not possible to adjust due to the number of connections and weights that are potentially needed. Thus, evolution in such networks becomes a complex task, as noise will be introduced into the system and all aspects of the setting suddenly play a fundamental role. Those seemingly minimal changes in the inherited code could generate considerable smart changes in the way networks react to the inputs, establish the hidden states, and produce the desired outputs because of the pattern setup that was learned through the evolutionary process. Programs with intricate genomes – at high or low population sizes – retain their sensitivity to reward variations and struggle to perform consistently in the task set.

Equation 2 : Natural Language Processing (NLP)

Equation for Genetic Algorithm Selection:

$$P(N_i) = \frac{F(N_i)}{\sum_{j=1}^{M} F(N_j)}$$ Where:

P(Ni) is the probability of selecting the network *Ni* from the population.

F(Ni) is the fitness of the network *Ni* based on performance metrics like accuracy or F1 score.

M is the total number of networks in the population.

4.2. Case Studies

In this section, we apply the Evolino framework to four classic problems in machine learning: pattern recognition via image classification, both in high-dimensional and semantic settings, text classification via natural language processing, and reinforcement learning for control problems. Each method in question performs network evolution while imposing minimal domain knowledge. Although our results are modest compared to some of the state-of-the-art methods for these problems, neural evolution remains incredibly slow. Nonetheless, the design of new methods or the combination of neural evolution with other techniques possibly outperforms existing solutions for otherwise hard problems. We hope our results will motivate such future work. Any loss measurements for predictions or errors are directly calculated using our optimized networks without any need for additional modeling or training.

First, we use Evolino to evolve neural designs that achieve optimal pattern recognition in a high-dimensional image classification task. Our use case begins with a series of training images augmented into 270-dimensional rows for respective objects. We then design a feature energy cost function to measure the amount of white-on-black pixels in each object. We localize information by calculating object maximum and minimum boundary pixel values at the coordinates that constitute the formation of objects. Data contains 28x28 pixel images. Another Euclidean space value is specified for the top section of the page. The root mean square (RMS) value of the feature energy in formation indicators then becomes the loss function. As a result, optimal 1, 2, and 3-layer perceptron and convolution network designs are found for image causality identification for the digit classes.

5. Applications in NLP

Natural Language Processing (NLP) tasks can be computationally intensive, as they often require processing large amounts of text. Many state-of-the-art NLP tools leverage recurrent neural networks and long short-term memory units, both of which can have networks orders of magnitude larger than feedforward networks. This means that, even for training data of constant size, NLP tasks are often challenging to optimize. In industrial settings, it is not unusual for NLP deployment times to increase linearly with the amount of text processing required. Solutions usually involve running in parallel, sharding, or invoking cloud resources, all of which are expensive in memory and operating costs, especially for smaller organizations or startups. In this chapter, neural networks that can perform more efficiently than their RNN and LSTM counterparts on NLP tasks will be sought out, once both traditional genetic algorithm-based learning and Q-learning are sufficiently effective at increasing the complexity of network design to allow non-memory-efficient agents to encode useful knowledge.

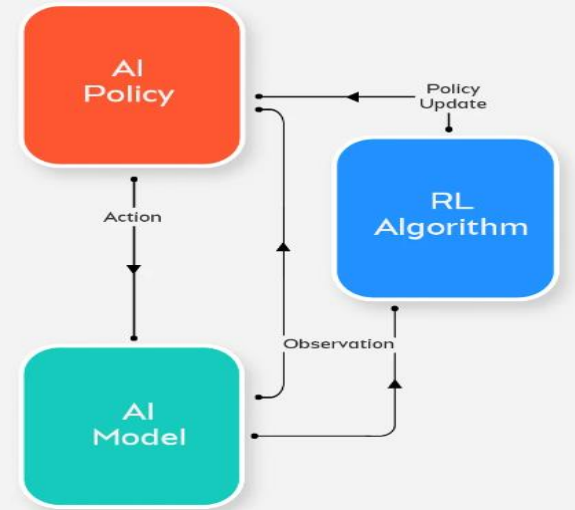

Fig 5 : NLP Applications of Reinforcement Learning

The class of NLP tasks considered are tasks that seek to score or classify a document, such as the determination of emotional sentiment, ratings of complexity, or classification of writing style. Language preprocessing for most modern neural network solutions to these tasks is often extensive, including complex transform layers, text batching, and word embedding. In many cases, complex translation layers, digesters, or connecting layers such as RNN encoders or traditional neural translation operations must also be trained on top of the resulting word embedding. As such, feature extraction for input is sometimes as time-intensive as training. I hypothesize that if the extracted features could be compressed or reduced significantly, traditional neural networks could recreate the same complex manifold to the compressed manifold's dimensionality, resulting in an operation that is feasible within some linear time as a function of reduced dimensionality.

5.1. Current Trends

Several recent works are trying to scale up NE methods even with CO and RI to train large and complex NN architectures with more complex search spaces. There are also works on using multimodal RBF kernels as the activation function. There are also works proposing algorithms that speed up the evaluation of the architectures; however, there is still an increasing demand for computationally efficient evaluators of the architectures, and this could potentially act as a bottleneck for the usage of the NE-based approaches. In the sense of CO and RI, some works also employ prior knowledge to regularize the learned architectures; however, it is quite common for NE to exemplify CO and RI to try and focus on HS. Hand tuning is still common and rarely focuses on making the optimization process interpretable.

Because of these limitations, it is also interesting to analyze and define principles that are required for automated NN architecture design and search from other perspectives. In this paper, we provide a large-scale, broad, and systematic empirical benchmarking of some of the most recent and commonly used Neural Evolution (NE) algorithms. With over 40 different Neural Evolution algorithms that appear in the literature, benchmarking them through all designs becomes prohibitively difficult. Considering the constraints in available computation and the difficulty of applying some of the designs in practice, it is often the case that these designs are evaluated only in a handful of tasks. It is also unclear if a design's performance will consistently translate to tasks in different domains and with different properties, such as image classification, reinforcement learning, and language modeling; the domains that are of interest to most practitioners.

5.2. Case Studies

In the sections below, we describe three case studies where we utilized the proposed genetic algorithm approach to evolve neural networks. The applications have different data dimensions and data formats, such as computer vision and NLP. We also used this genetic algorithm to generate a convolutional neural network for image classification, an LSTM design for NLP, and feedforward networks for Markov decision processes in reinforcement learning.

5.2.1. Evolving CNN

Image classification is a challenging task that requires significant engineering and design effort. Evolving neural networks have been used to automatically design appropriate building blocks instead of relying on expert techniques. Our work enhances this and proposes an approach that uses a genetic algorithm to evolve the entire CNN. We remove heuristic processes such as choosing filters, reducing image inputs, and placing activation functions. We propose a genetic algorithm to optimize these image-processing elements and develop appropriate CNN structures, including layer numbers, activation functions, and batch normalization layers. The effectiveness of the proposed approach is validated on benchmark datasets. Our experimental results show that the proposed approach is efficient and improves the performance of benchmark CNNs.

5.2.2. Evolving LSTM

In addition to evolving the CNN structures that dominate these empirical attempts in image classification, we also study the application of evolutionary deep learning to more general problems. As a natural extension of existing neural network architectures, we design a genetic algorithm to evolve long short-term memory networks to handle NLP problems in various dimensions, such as 1D NLP and 2D NLP problems, and define an architecture that can flexibly adjust to process the input. We design a loss function evaluation scheme that provides a more realistic assessment of the performance of the proposed designs in each generation. Our longitudinal study is the first to combine evolving 2D LSTM, decision trees, and image clustering to solve NLP problems, and our experimental results also emphasize its effectiveness in practical settings.

6. Applications in Reinforcement Learning

In this chapter, we will examine a variety of applications for the tools underlying our experiments in computational convergence. Starting with this chapter, we will talk about ways that state-of-the-art research in deep learning can benefit from the advances in the hybrid AI area that we have been addressing. We start our demonstration in the simplest reinforcement learning applications just to help illustrate how data can be used in RL, but the ideas described are generally applicable over a wider range of control applications, and indeed artificially determined target vectors are a common part of state-of-the-art training for many

types of neural network architectures. We motivate our investigation of training CTL systems with a somewhat simplified example that we think can help to illustrate how even a little bit of useful prior knowledge can aid in the training and performance of neural network controllers, particularly for the inherently fuzzy multiple-goal systems studied in hybrid AI platforms like CTL's connectionist techniques. The simulation task chosen is directed at a basic concept of controlled dynamic systems performance.

Neural networks have proven to be highly effective trainable function modelers. However, the use of existing neural architectures for higher-level system control tasks is often severely cramped by a lack of appropriate guiding control training and/or limitations in neural network response and behavior. We examine motivations for evoking desired response behavior from a neural network with the use of preconditioned target vectors. In cases of control system learning, the complete a priori knowledge of the network behavior exists only during the learning process. Consequently, it may be very difficult to control a planning experiment because we would have to solve the reinforcement learning problem.

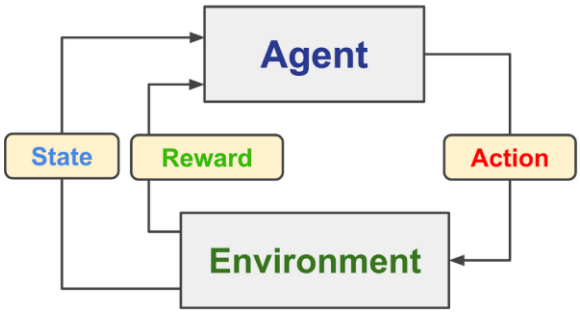

Fig 6 : Reinforcement Learning vs Genetic Algorithm

6.1. Overview of RL and NNs

To understand the integration of genetic algorithms and neural networks into reinforcement learning, a few basic concepts need to be established. It is an AI problem where an agent plays a game in an environment with several states and then modifies its actions accordingly to get a reward. After many games, it follows a policy that maps each state of a game to an action. Neural networks can be modified in such a way that they can also act as reinforcement learning agents. They can follow a policy well if some input can be provided. Many of the optimization strategies in neural networks are either optimization of the weights in the output layer using gradient descent methods or optimization through either backpropagation or supervised learning. These methods can be considered as modifications in the probability of the action with the maximum current reward, i.e., in the Q-value.

However, in cases of large spaces of state and action, it is difficult to set actions with the maximum value. Large Q-tables can also contain many values as the space grows, and experience replay can be used to reduce the correlation between close state-action pairs, so it acts as a buffer, storing experience and randomizing over it when a batch of mini-batches is generated. These approaches have their downsides as well: they tend to be computationally expensive to train. A genetic algorithm can be trained to perform very well in search and autocorrelated problems using only binary strings as the phenotype, and at the same time, very little mathematical knowledge is required. In many environments, reinforcement learning using backpropagation can deliver results significantly faster than genetic algorithms in the optimized mapping of Q-values.

6.2. Case Studies

Next, we present three demonstrations of using EFLONNs to solve different types of machine-learning tasks. The three demonstrations cover image classification with various types of data augmentation, text classification with stop-word removal, and reinforcement learning in the general Atari 2600 game environment. Firstly, in the image classification task, the original training images are applied with various augmentation techniques to expand the training set. The expanded set is then used as the training data for EFLs. This can be realized by regarding the original training images as some reconstructed versions of the generated ones, which are then used in Evans learning as amplified or distorted training data. Tests show that EFLONN trained using EFLs and the expanded set outperforms those trained by CNNs and other approaches that use the original training data only. Secondly, the text classification task sometimes benefits from removing some words that appear frequently in the corpus. This suggests that context-aware word representation is helpful for text classification. One way to achieve this is to use EFLONN for text classification in which relationships between high-frequency and low-frequency words are captured. The other way is to preprocess text data by removing high-frequency words using EFLN. Both approaches outperformed CNN and achieved state-of-the-art performance in the SST task. Finally, we played with reinforcement learning using EFLONN training in the Atari 2600 game environment. The experimental results revealed that EFLONN is a promising approach for developing stronger general game models, in particular, when combined with augmentation techniques.

7. Conclusion and Future Directions

We found that neuro-evolution can be accomplished efficiently using genetic algorithms. After evolving three significantly different designs of neural networks for three different tasks, we can concur with the common sentiment that 'smaller is better.' For NLP and reinforcement learning applications, smaller model sizes did not impair accuracy or effectiveness. We found that evolving ensemble designs could yield more robust classifiers by allowing for a soft voting approach to classification. We can also conclude that evolving neural network designs can yield unique and interpretable architectures. Future work could involve any number of evolutionary algorithms to find the best neural network. For example, we could try to mimic the algorithms that nature tried through a tree search with reinforcement learning guiding the search away from what has been empirically proven to be less frequently successful. Another simple future project just involves using the general methods written to tune the neural network's size and seeing if that will improve cross-validation F1 score. We could then add one-at-a-time searches where we eliminate connections by either the lowest weight or by individual node or layer elimination. Other improvements would include adding the ability to evolve a single-layer, one-node dense model instead of the single node, which reduces the process of feature engineering. Additionally, we would want to move the ensemble architecture from just a weighting scheme to allow up to three layers.

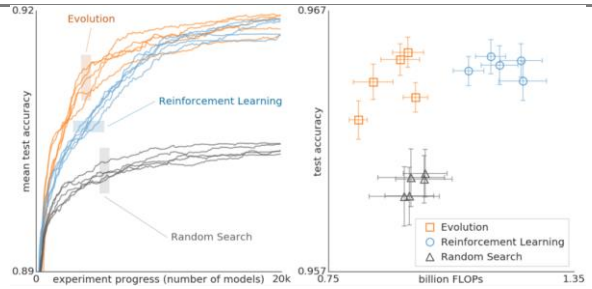

Fig 7 : Using Evolutionary AutoML to Discover Neural Network Architectures

7.1. Summary of Findings

In this work, genetic algorithms have been used to evolve neural architectures. This work has investigated the types of neural architectures that can be evolved in connection with image classification, sentiment analysis, and reinforcement learning. The image classification architectures evolved on a dataset, with fully connected architectures and CNNs emerging as the best-performing architectures. For reinforcement learning, the neural architectures were evolved to be used as a Q-function approximator in the Q-learning method and were evolved and tested on classical games. For this domain, a variety of different recurrent architectures were evolved, with LSTMs, GRUs, memory cells, and feed forwards all evolving to be effective architectures. Finally, for natural language processing, the architectures were evolved in connection with a sentiment analysis dataset. For this domain, fully connected and CNN architectures offered the best performance. This work has found that when evolving architectures, even those closely related architectures can have significantly different performances. More sophisticated methods for evolving or searching neural network architectures have been developed or proposed with increasing frequency. Such methods fall broadly into two categories, with the first being methods that attempt to search over a pre-decided network structure with varying weights and the second being those that attempt to search over both network structure and weights. Despite the recent advances in developing more sophisticated search routines, evolutionary computation approaches still make for intuitive and simple-to-implement tools in these tasks, while also providing a form of finite-sample bound through population-based optimization methods. This work has explored a basic evolutionary computation-based strategy for constructing neural architectures and has exhibited the success of genetic algorithms on different neural network tasks: those traditionally associated with computer vision, those associated with natural language processing, and those associated with reinforcement learning.

Equation 3 : Elitism in Selection

Equation for Elitism Strategy:

$$S = \{N_i \in P | i \leq k\}$$

Where : S is the set of selected elite networks.

P is the population of networks.

k is the number of top-performing networks retained based on their fitness scores.

7.2. Potential Areas for Further Research

In the proposed implementation, a standard genetic algorithm is used with a preprocessing approach outlined. Many possible principles may be exploited in the optimization process. Further investigation may be performed to determine if the derived benefits are complementary and perform well with alternative genetic-inspired optimization methods. Size weight centroids require reset or correction upon loss of diversity events, e.g., maintaining diversity via a measure of the clusters' radius or some representation of sparsity. Potentially leveraging recent advances in deep learning to derive a meaningful numerical cluster radius may be beneficial. Furthermore, network topology optimization is performed, exceeding traditional hyperparameter searches. Future research may investigate the potential effect of training with a more generic dataset, mimicking the one-shot learning behavior of the evolved network.
Change the loss function between processes to enable a variety of neural network functions to be optimized, potentially removing the requirement for transfer learning. Evolving the loss function with the neuron topology further simplifies the problem and ensures that the function evaluation methods scale as similarly as possible. Investigate autoregression. Additionally, most efforts regarding network topology alterations investigate altering the connections to improve resilience, energy conservation, or alterations within a hardware implementation, allowing neural network function changes with further research. Periodic enhancements to the process enable cross-computations among seeds or alternative population variations, for example, tiny differences between the network approximations or optimizing more than one weight vector.
The population variety is essential primarily when using parallel hardware, which is not explored in the current work. Additionally, it is possible to mix methodologies to retain the memory cell with programmable logic devices or other approaches, reducing the derived network resources and therefore improving the potential for parallel training possibilities. Implementing the genetic approach with alternative hardware approaches will enable the native neural functions to be optimal and further investigate the use of non-traditional device functions in future work. Branching prediction may also be used further with other types of hardware, potentially offering energy conservation, reduced memory usage, or other benefits.

8. References

[1] Avacharmal, R., Gudala, L., & Venkataramanan, S. (2023). Navigating The Labyrinth: A Comprehensive Review Of Emerging Artificial Intelligence Technologies, Ethical Considerations, And Global Governance Models In The Pursuit Of Trustworthy AI. Australian Journal of Machine Learning Research & Applications, 3(2), 331-347.

[2] Mahida, A. Secure Data Outsourcing Techniques for Cloud Storage.

[3] Perumal, A. P., & Chintale, P. Improving operational efficiency and productivity through the fusion of DevOps and SRE practices in multi-cloud operations.

[4] Kommisetty, P. D. N. K. (2022). Leading the Future: Big Data Solutions, Cloud Migration, and AI-Driven Decision-Making in Modern Enterprises. Educational Administration: Theory and Practice, 28(03), 352-364.

[5] Bansal, A. (2022). Establishing a Framework for a Successful Center of Excellence in Advanced Analytics. ESP Journal of Engineering & Technology Advancements (ESP-JETA), 2(3), 76-84.

[6] Vaka, D. K. (2024). The SAP S/4HANA Migration Roadmap: From Planning to Execution. Journal of Scientific and Engineering Research, 11(6), 46-54.

[7] Vaka, D. K. SAP S/4HANA: Revolutionizing Supply Chains with Best Implementation Practices. JEC PUBLICATION.

[8] Avacharmal, R. (2024). Explainable AI: Bridging the Gap between Machine Learning Models and Human Understanding. Journal of Informatics Education and Research, 4(2).

[9] Mahida, A., Chintale, P., & Deshmukh, H. (2024). Enhancing Fraud Detection in Real Time using DataOps on Elastic Platforms.

[10] Mandala, V., & Kommisetty, P. D. N. K. (2022). Advancing Predictive Failure Analytics in Automotive Safety: AI-Driven Approaches for School Buses and Commercial Trucks.

[11] Kumar Vaka Rajesh, D. (2024). Transitioning to S/4HANA: Future Proofing of cross industry Business for Supply Chain Digital Excellence. In International Journal of Science and Research (IJSR) (Vol. 13, Issue 4, pp. 488–494). International Journal of Science and Research. https://doi.org/10.21275/sr24406024048

[12] Perumal, A. P., Chintale, P., Molleti, R., & Desaboyina, G. (2024). Risk Assessment of Artificial Intelligence Systems in Cybersecurity. American Journal of Science and Learning for Development, 3(7), 49-60.

[13] Kommisetty, P. D. N. K., & Abhireddy, N. (2024). Cloud Migration Strategies: Ensuring Seamless Integration and Scalability in Dynamic Business Environments. In International Journal of Engineering and Computer Science (Vol. 13, Issue 04, pp. 26146–26156). Valley International. https://doi.org/10.18535/ijecs/v13i04.4812

[14] Bansal, A. (2024). Enhancing Customer Acquisition Strategies Through Look-Alike Modelling with Machine Learning Using the Customer Segmentation Dataset. International Journal of Computer Science and Engineering Research and Development (IJCSERD), 14(1), 30-43.

[15] Vaka, Dilip Kumar. "Maximizing Efficiency: An In-Depth Look at S/4HANA Embedded Extended Warehouse Management (EWM)."

[16] Avacharmal, R., Pamulaparthyvenkata, S., & Gudala, L. (2023). Unveiling the Pandora's Box: A Multifaceted Exploration of Ethical Considerations in Generative AI for Financial Services and Healthcare. Hong Kong Journal of AI and Medicine, 3(1), 84-99.

[17] Mahida, A. Explainable Generative Models in FinCrime. J Artif Intell Mach Learn & Data Sci 2023, 1(2), 205-208.

[18] Vaka, D. K. (2024). Enhancing Supplier Relationships: Critical Factors in Procurement Supplier Selection. In Journal of Artificial Intelligence, Machine Learning and Data Science (Vol. 2, Issue 1, pp. 229–233). United Research Forum. https://doi.org/10.51219/jaimld/dilip-kumar-vaka/74

[19] Perumal, A. P., Deshmukh, H., Chintale, P., Molleti, R., Najana, M., & Desaboyina, G. Leveraging machine learning in the analytics of cyber security threat intelligence in Microsoft azure.

[20] Kommisetty, P. D. N. K., & dileep, V. (2024). Robust Cybersecurity Measures: Strategies for Safeguarding Organizational Assets and Sensitive Information. In IJARCCE (Vol. 13, Issue 8). Tejass Publishers. https://doi.org/10.17148/ijarcce.2024.13832

[21] Bansal, A. (2023). Power BI Semantic Models to enhance Data Analytics and Decision-Making. International Journal of Management (IJM), 14(5), 136-142.

[22] Muthu, J., & Vaka, D. K. (2024). Recent Trends In Supply Chain Management Using Artificial Intelligence And Machine Learning In Manufacturing. In Educational Administration Theory and Practices. Green Publication. https://doi.org/10.53555/kuey.v30i6.6499

[23] Pillai, S. E. V. S., Avacharmal, R., Reddy, R. A., Pareek, P. K., & Zanke, P. (2024, April). Transductive–Long Short-Term Memory

Network for the Fake News Detection. In 2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE) (pp. 1-4). IEEE.

[24] Mahida, A. (2024). Integrating Observability with DevOps Practices in Financial Services Technologies: A Study on Enhancing Software Development and Operational Resilience. International Journal of Advanced Computer Science & Applications, 15(7).

[25] Vaka, D. K. (2024). Procurement 4.0: Leveraging Technology for Transformative Processes. Journal of Scientific and Engineering Research, 11(3), 278-282.

[26] Mandala, V. (2022). Revolutionizing Asynchronous Shipments: Integrating AI Predictive Analytics in Automotive Supply Chains. Journal ID, 9339, 1263.

[27] Bhardwaj, A. K., Dutta, P. K., & Chintale, P. (2024). AI-Powered Anomaly Detection for Kubernetes Security: A Systematic Approach to Identifying Threats. In Babylonian Journal of Machine Learning (Vol. 2024, pp. 142–148). Mesopotamian Academic Press. https://doi.org/10.58496/bjml/2024/014

[28] Kommisetty, P. D. N. K., & Nishanth, A. (2024). AI-Driven Enhancements in Cloud Computing: Exploring the Synergies of Machine Learning and Generative AI. In IARJSET (Vol. 9, Issue 10). Tejass Publishers. https://doi.org/10.17148/iarjset.2022.91020

[29] Bansal, A. (2024). Enhancing Business User Experience: By Leveraging SQL Automation through Snowflake Tasks for BI Tools and Dashboards. ESP Journal of Engineering & Technology Advancements (ESP-JETA), 4(4), 1-6.

[30] Vaka, D. K., & Azmeera, R. Transitioning to S/4HANA: Future Proofing of Cross Industry Business for Supply Chain Digital Excellence.

[31] Avacharmal, R., Sadhu, A. K. R., & Bojja, S. G. R. (2023). Forging Interdisciplinary Pathways: A Comprehensive Exploration of Cross-Disciplinary Approaches to Bolstering Artificial Intelligence Robustness and Reliability. Journal of AI-Assisted Scientific Discovery, 3(2), 364-370.

[32] Mahida, A. (2023). Enhancing Observability in Distributed Systems-A Comprehensive Review. Journal of Mathematical & Computer Applications. SRC/JMCA-166. DOI: doi. org/10.47363/JMCA/2023 (2), 135, 2-4.

[33] Vaka, D. K. (2024). From Complexity to Simplicity: AI's Route Optimization in Supply Chain Management. In Journal of Artificial Intelligence, Machine Learning and Data Science (Vol. 2, Issue 1, pp. 386–389). United Research Forum. https://doi.org/10.51219/jaimld/dilip-kumar-vaka/100

[34] Perumal, A. P., Deshmukh, H., Chintale, P., Desaboyina, G., & Najana, M. Implementing zero trust architecture in financial services cloud environments in Microsoft azure security framework.

[35] Kommisetty, P. D. N. K., vijay, A., & bhasker rao, M. (2024). From Big Data to Actionable Insights: The Role of AI in Data Interpretation. In IARJSET (Vol. 11, Issue 8). Tejass Publishers. https://doi.org/10.17148/iarjset.2024.11831

[36] Bansal, A. Advanced Approaches to Estimating and Utilizing Customer Lifetime Value in Business Strategy.

[37] Vaka, D. K. (2024). Integrating Inventory Management and Distribution: A Holistic Supply Chain Strategy. In the International Journal of Managing Value and Supply Chains (Vol. 15, Issue 2, pp. 13–23). Academy and Industry Research Collaboration Center (AIRCC). https://doi.org/10.5121/ijmvsc.2024.15202

