LLM Guided Evolution - The Automation of Models Advancing Models

Anonymous Author(s)

ABSTRACT

In the realm of machine learning, traditional model development and automated approaches like AutoML typically rely on layers of abstraction, such as tree-based or Cartesian genetic programming. Our study introduces "Guided Evolution" (GE), a novel framework that diverges from these methods by utilizing Large Language Models (LLMs) to directly modify code. GE leverages LLMs for a more intelligent, supervised evolutionary process, guiding mutations and crossovers. Our unique "Evolution of Thought" (EoT) technique further enhances GE by enabling LLMs to reflect on and learn from the outcomes of previous mutations. This results in a self-sustaining feedback loop that augments decision-making in model evolution. GE maintains genetic diversity, crucial for evolutionary algorithms, by leveraging LLMs' capability to generate diverse responses from expertly crafted prompts and modulate model temperature. This not only accelerates the evolution process but also injects expert like creativity and insight into the process. Our application of GE in evolving the ExquisiteNetV2 model demonstrates its efficacy: the LLM-driven GE autonomously produced variants with improved accuracy, increasing from 92.52% to 93.34%, without compromising model compactness. This underscores the potential of LLMs to accelerate the traditional model design pipeline, enabling models to autonomously evolve and enhance their own designs.

CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; **Learning from critiques**; *Natural language generation.*

KEYWORDS

Large Language Models, Automated Machine Learning, Evolutionary Algorithms

ACM Reference Format:

Anonymous Author(s). 2024. LLM Guided Evolution - The Automation of Models Advancing Models. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (GECCO '24).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the ever-evolving domain of machine learning, the convergence of human cognitive skills and automated algorithms is entering a pivotal junction. This paper introduces "Guided Evolution" (GE), a novel framework that combines the human-like expertise of Large Language Models (LLMs) with the robust capabilities of Neural Architecture Search (NAS) through genetic algorithms. This innovative fusion advances automated machine learning, elevating traditional NAS by integrating a more insightful, intelligently guided evolutionary process.

Central to this framework is our "Evolution of Thought" (EoT) technique, which extends and refines concepts like Zero-Shot Chain-of-Thought, Automated Chain-of-Thought, and Tree-of-Thought [7, 17, 18]. These methodologies aim to improve the reasoning capabilities of LLMs. EoT takes a unique step forward by enabling LLMs to receive result-driven feedback, empowering them to make informed improvements based on the performance of their prior code augmentations, a significant advancement in intelligent automated machine learning.

EoT catalyzes LLMs to introspect and fine-tune suggestions based on past iterations, creating a self-enhancing feedback loop that fine-tunes architectural evolution. At the same, GE maintains essential genetic diversity for evolutionary algorithms while injecting human-like expertise and creativity into the evolutionary framework. Building from the insights of Ma et al. [11], our Guided Evolutionary framework is further enhanced by a Character Role Play (CRP) technique, to markedly increase the feasibility, usefulness and creativity of ideas engendered by the LLM.

The effectiveness of the Guided Evolution (GE) framework is showcased in the evolution of the ExquisiteNetV2 model. This evolution, initiated with a State-Of-The-Art (SOTA) seed model, not only demonstrates the capacity of LLMs to build upon and enhance SOTA models in collaboration with human expertise but also underscores their autonomous model design. This case study illustrates the framework's self-sufficient ability to generate improved model variants, emphasizing the burgeoning impact of LLMs in redefining traditional model design pipelines, a step towards models that independently evolve and refine their architectures.

2 NEURAL ARCHITECTURE SEARCH

Neural Architecture Search (NAS) stands at the forefront of machine learning innovation, focusing on the automated discovery of optimal neural network architectures. Within this dynamic field, a variety of methodologies have emerged, each contributing unique approaches and benefits. These include Reinforcement Learning (RL), Evolutionary Algorithms (EAs), Surrogate Model-Based Optimization, and One-Shot Architecture Search.

Focusing first on Reinforcement Learning, important contributions by Zoph and Le [20] and Baker et al. [1] have showcased how an RL agent can be used to iteratively optimize network architectures, with an emphasis on maximizing key performance metrics like validation accuracy. Building on this foundation, Ramachandran, Zoph, and Le expanded the RL [14] application to include the search for activation functions, furthering the versatility of this approach.

GECCO '24, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

In a distinct but equally innovative vein, Evolutionary Algorithms draw inspiration from the principles of biological evolution. They utilize mutation and crossover processes to evolve network architectures. The potential of neuroevolution in NAS was aptly demonstrated by Real et al. [15], creating the first evolved model to surpass hand-designs on ImageNet. Complementing this, CoDeepNEAT, developed by Miikkulainen et al. [12], extends the NEAT algorithm to deep learning, co-evolving modules and blueprints for constructing deep neural networks, thus highlighting the adaptability of evolutionary approaches in NAS. Furthermore, Lu et al. [10] introduced NSGA-NET, a multi-objective genetic algorithm, signifying a substantial leap forward in this domain.

Surrogate Model-Based Optimization diverges from RL's trial-and-error approach by employing predictive models to estimate the performance of various architectures to better navigate the architecture space Cai et al. [2] .

Lastly, the realm of One-Shot Architecture Search represents a paradigm shift in NAS. It involves training an expansive super network that encompasses all potential architectures within the search space. Notable contributions in this area include the Efficient Neural Architecture Search (ENAS) by Pham et al. [13], and the Differentiable Architecture Search (DARTS) by Liu, Simonyan, and Yang [9]. By leveraging shared weights across architectures, these methods significantly reduce computational requirements.

In the rapidly evolving field of machine learning, our research introduces novel methodologies that synergize evolutionary algorithms (EA) with the reasoning and generative capabilities of LLMs. We present "Guided Evolution" and "Evolution of Thought" (EoT), innovative approaches designed to address the inherent inefficiencies in traditional EA, a challenge highlighted in the works by Guariso et al. [4] and Yang et al. [16]. These inefficiencies become particularly pronounced with the increasing size and training costs of modern machine learning models, necessitating a novel approach to enhance efficiency and adaptability.

Our methodologies integrate LLMs within the genetic algorithm framework, thereby introducing a robust intrinsic feedback mechanism (EoT) and an intelligently supervised evolution, where mutations and genetic crossovers are chosen by the LLMs framework. This integration not only alleviates the inefficiency issues but also significantly reduces the manual intervention often required in EA, such as parameter tuning and range setting. By leveraging the domain expertise of LLMs, we achieve a dynamic, efficient, and adaptable framework for exploring complex solution spaces in machine learning. Moreover, genetic diversity is preserved through the LLM's capability to produce an unlimited number of responses from a single prompt. This is achieved by adjusting the temperature parameter and utilizing a range of hand crafted prompts, thereby facilitating a thorough exploration of alternative model designs.

Traditional EA methods are often limited by their rigid mutation and crossover structures, a constraint partially addressed by developments such as those by Zutty et al. [21], who introduced human-derived primitives. However, the utilization of predefined building blocks and tree structures in traditional genetic programming necessitates a prolonged set-up time. Rather, by generating individuals directly as interpretable code, LLMs save time by automatically enriching the evolutionary process with contextual understanding and nuanced insights which normally would come from human-derived primitives. Moreover, they are not constrained by a rigid structure; their development is limited only by the LLM's creativity.

The prevalent issue with LLMs stems from their design to emulate, rather than authentically generate, human-like responses. This often results in the production of erroneous or fabricated information, a phenomenon known as "hallucination" [5]. Addressing this, prompt engineering emerges as a method to navigate LLMs towards structured, logical reasoning via carefully formulated prompts. Our research significantly extends this nascent field. Additionally, an advantage of evolution is in the natural experimentation that arises during the process: a "hallucinated" piece of code will be evaluated against supervised data, its objectives and fitness scores accurately reflecting its success. Not every individual need be a winner as long as the evolution continues to make steady progress.

Building on the foundational work of Kojima et al. [7] in Chain-of-Thought (CoT) reasoning and the advancements by Zhang et al. [18] in Auto-CoT, our study introduces the EoT approach. This method incorporates genetic algorithms to enable LLMs to autonomously curate and refine prompts. Detailed in Section 3.2.2, EoT leverages selective evolutionary strategies to not only augment the quality of LLM outputs but also to introduce a mechanism for self-optimization. This approach marks a step forward in developing self-improving, adaptive language models, showcasing the potential for more autonomous and efficient evolution in the field of machine learning.

3 METHODOLOGY

In our methodology, we introduce the Guided Evolution (GE) framework, initiating with ExquisiteNetV2 as the seed model. This model is first dissected into discrete code blocks, each corresponding to a distinct Python class. These blocks function analogously to genetic segments in a genome, providing the foundational elements for our LLM driven evolutionary process.

For this study, we utilized Mixtral [6], Mistral AI's 8x7B Mixture of Experts Open Source Model, noted for its balance of efficiency and high performance in code generation. Mixtral employs a Sparse Mixture of Experts (SMoE) architecture, selectively utilizing 13B out of 47B parameters across eight feedforward blocks for each token, optimizing for inference efficiency. This model demonstrates superior performance in code generation compared to competitors like Llama 2 70B, making it particularly effective for the precise and complex code modifications required in our Guided Evolutionary framework.

In a departure from traditional evolutionary algorithms (EA), our approach replaces conventional mutation and mating operations with a series of prompts directed at a LLM. This introduces a more dynamic and exploratory dimension to the EA framework. To further encourage innovative architectural evolution, we utilize "Character Role Play" (CRP), as outlined in Section 3.2.1. This strategy enhances the creativity and unconventionality of guided mutations.

Another key feature of our methodology is the EoT approach (detailed in Section 3.2.2). This induces an intrinsic feedback mechanism within the EA framework, allowing it to adapt and respond

Gene Segment	Description
get_optimizer	Optimization Process
SE	Squeeze-and-Excitation
SE_LN	Squeeze-and-Excitation Layer-Normalization
DFSEBV2	Feature-Processor
FCT	Feature-Concentrator
EVE	Extreme-Value-Expansion
ME	Max-Min Expansion
DW	Depthwise Convolution
ExquisiteNetV2	Aggregator

Table 1: Block Descriptions

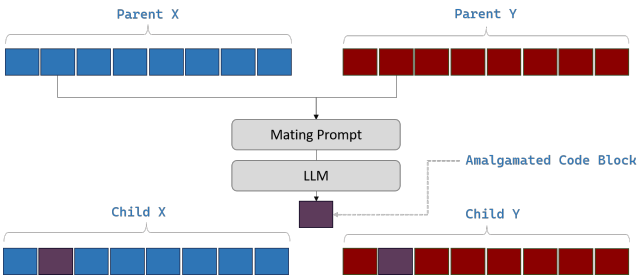

Figure 1: LLM Driven Code Block Mating

to effective changes in the model architecture. Through this mechanism, the LLM becomes attuned to successful adaptations, guiding the evolution of the model architecture in a more informed and targeted manner.

This combination of LLM repeated prompting and the EoT methodology creates a dynamic environment for the guided evolution of the seed model. It allows for both exploratory diversification and the exploitation of effective architectural changes, providing a novel approach to evolving neural network architectures.

By breaking down the ExquisiteNetV2 model into modular segments, such as the feature concentrator, we can direct the LLM's capabilities more precisely. This focused approach ensures that each segment is optimized in isolation, allowing for a detailed and specific enhancement of the model's overall architecture. Table 1 delineates the decomposition of the state-of-the-art model into distinct code segments, providing a structured overview of its components.

3.1 Mating

To enhance genome mating efficiency, the Guided Evolution (GE) framework employs a strategic selection process. Figure 1 shows the process of randomly selecting two distinct code segments from the available genomes, ensuring they are not identical to avoid ineffectual mating attempts. These selected segments are then processed through a LLM. The LLM's task is to intelligently amalgamate these segments, aiming to either heighten accuracy or boost efficiency in the resultant genome.

Category	Summary
Hyperparam.	Modify existing parameter values to alter the model's behavior.
Hyperparam. Uncommon	Change parameter values to less conventional ones to explore diverse outcomes.
Complex	Introduce advanced functionality that could potentially enhance the model's accuracy.
Reduce Model Size	Streamline the code by reducing parameters with minimal impact on the model's performance.
Uncommon	Implement a distinctive or rarely used enhancement to the model.
Significant	Execute substantial modifications to the code, incorporating auxiliary functions for improved structure and capability.

Table 2: Prompt Type Categories

3.2 Mutation

Incorporating a LLM into our mutation process has introduced a higher degree of flexibility within the Guided Evolutionary framework. Mutation here involves the random selection of code segments, which are then augmented through LLM prompts. To enhance the diversity of these mutations, we varied the model temperature for each prompt between 0.05 and 0.4, and set the maximum token length to a random value within the range of 600 to 1400 tokens.

Distinguishing features of our approach include the integration of "Character Role Play" and "Evolution of Thought" (EoT), the latter being akin to the AutoCoT methodology and introduced in Section 3.2.2. "Character Role Play" broadens the exploration within the potent areas of the LLM's latent space, fostering innovative suggestions. EoT, on the other hand, introduces a feedback loop, refining the LLM's output over successive iterations for progressive improvement.

This methodology positions the GE framework not merely as a tool for random mutation but as a strategic guide for the LLM. It directs the exploration and enhancement of solutions in a more focused and effective manner, as detailed in our discussion of EoT in Section 3.2.2.

3.2.1 Character Role Play: In the Guided Evolution framework, we approach the mutation process distinctively, diverging from conventional methods. Our initial experiments utilizing a LLM for code mutation identified a tendency towards producing standard solutions, exemplified by the frequent selection of a 0.2 dropout rate. This trend likely mirrors the commonality of such values in the model's training dataset, limiting the exploration of a more diverse solution space.

To address this, we implemented a method that incorporates some unconventional prompt templates, specifically designed to prompt the LLM away from generating these typical solutions. This method employs a range of randomly selected prompt templates, some of which guide the LLM to generate more atypical or novel code modifications. Descriptions of these prompt categories can be found in Table 2

Name	Summary Description
Expert	A top expert in machine learning with a deep understanding of advanced AI methods.
Dr. MaGoo	An AI innovator with a serendipitous approach, surprising peers with unorthodox model enhancements.
Innovative Scientist	A globally famous AI researcher known for creative and unconventional techniques.

Table 3: Character Role Play

This approach increased the diversity and scope of exploration in our preliminary tests, boosting the likelihood of developing solutions that surpass current state-of-the-art models. To further encourage diversity and enhance the quality of generated code, we introduced an element of "expert" roleplay in our prompts. This feature was intended to encourage the LLM to access its latent knowledge of "expert-level code" by casting it as a skilled AI researcher when queried. Namely, we incorporated a range of character personas for the LLM to adopt, each designed to induce different types and qualities of mutations.

Specifically, we choose three unique character roles which act as compatible extensions to the previously mentioned 6 foundational prompts. These character types were chosen after manual analysis monitoring the quality of the produced models in response to both the standard prompts and the character-augmented prompts. Details regarding the three character roles, including their descriptions, are delineated in Table 3.

Our evolutionary algorithm utilizes six fundamental prompt templates, compatible with three distinct "expert" character roles. By combining these baseline prompts with the character role variations that modify the base template, we achieve a total of twenty-four diverse prompts. This variety in prompts fosters a broader and more enriched exploration of potential solutions by the LLM.

In future studies, prompt templates and character implementation are something that could be co-evolved as opposed to arbitrarily chosen.

3.2.2 Evolution of Thought: In 2022, Zhang et al [18]. pioneered the Auto Chain of Thought prompting (Auto-CoT), an innovative approach that automates the construction of demonstrations for LLMs. This method employs question clustering, where a dataset is divided into several clusters of similar questions. For each cluster, a representative question is selected, and its reasoning chain is generated using Zero-Shot-CoT with simple heuristics. Thus Auto-CoT addresses the limitations of manual and zero-shot CoT (Chain of Thought) prompting by leveraging the diversity of questions to mitigate the impact of errors in reasoning chains generated by LLMs. The key innovation lies in utilizing clustering algorithms to group similar questions, thus ensuring a variety of reasoning types and improving the robustness of the model's problem-solving capabilities.

Our study introduces the "Evolution of Thought" (EoT) methodology, which extends the reasoning principles of both Auto-CoT and Manual-CoT. What sets EoT apart is its incorporation of feedback from evolutionary processes within the Guided Evolution (GE)

```
Q1: Let's explain step by step why this code segment
modification enhanced performance.

Example of initial code:
```Python code and explanation text```        <-- Seed Segment X

Example of modified code:
```Python code```                            <-- Elite Gene Segment X

Q2: With the insights from Q1 in mind, lets apply the
same principles in a concise manner to update the code
snippet below?

Code snippet to be augmented:
```Python code```                            <-- New Gene Segment Y

A1: Let us think step by step
```

Figure 2: EoT prompt template

framework. In this framework, each mutation subsequent to the first generation can conduct a basic LLM mutation, "Character Role Play", or the EoT mutation. EoT uniquely adopts the SPEA-2 elite selection algorithm to identify high-performing individuals from the previous generation. These individuals are then used as exemplary models for the LLM in subsequent mutations. This approach resonates with Zero-Shot-CoT through its "Let us think step by step" reasoning, and with Manual-CoT in its use of demonstrative examples. Leveraging the comparison of a selected elite block against its un-evolved seed serves as a thought guide for the LLM and encourages it to reflect about why mutations caused performance gains. The LLM is then asked to apply these derived insights to new code blocks, thus forming a performance enhancing feedback mechanism across generations. The EoT template is detailed in Figure 2.

Through analyzing empirical feedback on preceding concepts, EoT is designed to mimic and accelerate human like software development and invention.

3.3 Evolutionary Loop

This study introduces a novel approach in machine learning by integrating LLM adaptations into the three key components of evolutionary algorithms: genome creation, mutation, and mating. Detailed discussions on the adaptations of mating and mutation processes are presented in Section 3.1 and 3.2. The following section section provides an overview of the entire evolutionary process, which also covers how our method addresses gene creation.

In our evolutionary framework, genes are generated through LLM-driven mutations applied to copies of the state-of-the-art seed code. This gene creation method involves the seed code undergoing our unique LLM mutation procedure, intentionally excluding EoT mutations due to their reliance on previous change evaluations.

The evaluation process within this study is multi-objective, simultaneously considering accuracy on a holdout set and the model's parameter count. This dual-objective strategy enables the evolutionary algorithm to effectively balance accuracy and model size, optimizing both in tandem.

Elitism plays a crucial role in our approach due to the low likelihood of recovering or reconverging to a lost individual. We maintain elitism in the evolutionary loop through the Strength Pareto Evolutionary Algorithm 2 (SPEA-2) developed by Zitzler et al [19]. SPEA-2 was selected for its ability to preserve top-performing solutions while ensuring diversity. It achieves this balance via "fine-grained
```

fitness assignment" and "k-th nearest neighbor density estimation" methods that enhance exploration and prevent premature convergence by evaluating solutions based on dominance relationships and density in the objective space.

In the context of selection for mating and mutation processes, the study leverages the NSGA-II algorithm, as delineated by Deb et al. [3]. This algorithm is characterized by its 'fast non-dominated sorting' technique coupled with 'crowding distance' assessments, pivotal for curating a diverse yet superior pool of solutions. The fast non-dominated sorting algorithm stratifies solutions by their dominance levels, while the crowding distance metric evaluates the population's spatial density, preferentially guiding the selection process towards sparser regions. Such a methodology ensures an optimal equilibrium between the caliber and the variety of solutions, a crucial factor for effective exploration within the multiobjective optimization domain.

The integration of prompt variations significantly broadens the effective model search space, limited only by the LLM's vast output capabilities. This expansion greatly benefits the discovery of unique and effective solutions, but it also necessitates the retention of efficient individuals. To address this, before mutation and mating, both SPEA-2 and NSGA-II selections are conducted on populations of the same size. Post-augmentation and replacement through mating and mutation, the new individuals are concatenated with the SPEA-2 selected genes. Elite genes are only replaced if the new generation surpasses their performance. This strategy is employed to mitigate issues arising from the considerably expansive search space , ensuring the capture and retention of optimal solutions.

Algorithm 1: LLM-Guided Evolution Algorithm

Function LLMGuidedEvolution(*SeedCode*):
  Population ← InitializePopulation(*SeedCode*)
  ElitePopulation ← ∅
  HallOfFame ← ∅
  while *not ConvergenceCondition()* **do**
    EvaluatedPopulation ← Evaluate(*Population*)
    EliteCandidates ← SPEA2(*EvaluatedPopulation*)
    SelectionForMating ← NSGA2(*EvaluatedPopulation*)
    MatedPopulation ← LLMMate(*SelectionForMating*)
    MutatedPopulation ← LLMMutate(*MatedPopulation*)
    HallOfFame ← UpateHoF(*EvaluatedOffspring*)
    CombinedPopulation ← Concatenate(*EliteCandidates*, *EvaluatedOffspring*)
    Population ← *CombinedPopulation*
  return *HallOfFame*

In our mutation process, a probabilistic method, directed by "prob_eot" is used to alternate randomly between fixed prompt mutation and "Evolution of Thought" (EoT). When fixed prompt mutation is selected, the system chooses a single prompt template from either traditional direction prompts or Character Role Play prompts, as outlined in Section 3.2.1.

Algorithm 2: Detailed LLM Mutation Process

Function LLMMutate(*Gene*):
  // Select a random code block Y from the gene's alternative blocks
  BlockY ← SelectRandomBlock(*Gene*)
  // Probabilistic method to choose mutation type
  if *Random()* < *prob_eot* **then**
    // Perform Fixed Prompt Mutation
    Prompt ← SelectRandomPrompt(FixedPrompts ∪ RolePlayPrompts)
    MutatedBlockY ← FixedPromptMutation(*BlockY*, *Prompt*)
  else
    // Perform Evolution of Thought (EoT)
    Elite ← SPEA2Selection(*LastGeneration*, *k*)
    EliteIndividual ← SelectRandomIndividual(*Elite*)
    $(BlockX_{Elite}, BlockX_{Seed})$ ← SelectRandomBlock(*EliteIndividual* ∪ *SeedCode*)
    MutatedBlockY ← EoTMutation(*BlockY*, $BlockX_{Elite}$, $BlockX_{Seed}$)
  // Replace the original block Y with the mutated block Y
  Gene ← ReplaceBlock(*Gene*, *BlockY*, *MutatedBlockY*)
  return *Gene*

The developed evolutionary loop aims to optimally utilize the explorative capabilities of LLMs while ensuring to exploit effective adaptations.

3.4 Example Problem

In the pursuit of advancing neural architecture search methodologies, our investigation employed the CIFAR10 dataset, a seminal dataset developed by the Canadian Institute for Advanced Research, as a primary benchmark [8]. The dataset encompasses 60,000 32x32 pixel color images across 10 classes, with a standard division of 50,000 images for training and 10,000 for testing.

Our rationale for selecting CIFAR10 was twofold: firstly, the challenge presented by its low-resolution images, which necessitates sophisticated object recognition algorithms to address limited detail availability; and secondly, its extensive usage in the machine learning domain, which has resulted in a proliferation of highly effective SOTA models to benchmark our auto-enhancing GE against.

CIFAR10's balance of complexity and computational feasibility makes it an ideal candidate for evaluating new neural architectures. It demands intricate learning algorithms for accurate classification while remaining manageable in terms of data handling and computational requirements, a crucial consideration for developing from scratch a novel NAS methodology such as Guided Evolution.

Conclusively, the CIFAR10 dataset's historical significance in machine learning, combined with its unique challenges and practical usability, positions it as an optimal choice for our research. It not only serves as a fundamental benchmark but also as a catalyst for designing a new and effective NAS paradigm in a demanding and well-established domain.

3.5 State of the Art

Our study aimed to assess our framework's capacity to outperform human-engineered designs in well-researched domains. We used a state-of-the-art CIFAR-10 classifier as our seed model for autonomous evolution, chosen for its complex block and layer diversity, presenting a challenging environment for evolution.

A pivotal aspect of our methodology was the deliberate choice of models with fewer parameters. This decision was twofold: firstly, models with a lower parameter count emphasize the critical role of architectural ingenuity. Secondly, such models are more conducive to exhaustive experimentation within limited time frames, ensuring thorough evaluation.

We opted for "ExquisiteNetV2", a model innovatively crafted by Zhou and Su in 2022 (Zhou et al., 2022), as our foundational model. ExquisiteNetV2 is not only compact, ranking ninth in terms of size on the CIFAR-10 benchmark, but its architecture also comprises a variety of effective blocks. This aligns seamlessly with our goal of advancing sophisticated architectures through our evolutionary framework.

ExquisiteNetV2 is characterized by several innovative components. It incorporates Extreme-Value-Expansion (EVE) Blocks, which concatenate min and max pooling layers to rapidly discard unimportant features while retaining critical ones. The Feature-Concentrator Block integrates a 4x4 depthwise convolution with the EVE block to focus on preserving raw image features. In an effort to reduce parameter count, SE-Layer-Normalization (SE-LN) Blocks replace fully-connected layers with Layer Normalization in a Squeeze Excitation blocks to further reduce parameter count. The DFSEBV2 Block, a cornerstone of the ExquisiteNetV2 architecture, integrates pointwise convolutions, batch normalization, and depthwise convolutions for efficient feature processing. It uses SiLU and Hardswish activation functions for advanced pattern learning, and includes either an SE-LN or a standard SE block for improved channel-wise feature recalibration. This block's design is further enhanced by residual connections, ensuring optimal learning efficiency and computational economy. These elements make ExquisiteNetV2 a fitting candidate to demonstrate the capabilities of our guided evolutionary framework.

4 RESULTS

In this investigation, our primary objective was to leverage LLM Guided Evolution for the generation of ExquisiteNetV2 variants, with the dual goals of maintaining high test accuracy while constraining model size. Furthermore, this research endeavored to evaluate the contributions of two novel methodologies: Evolution of Thought (EoT) and Character Role Play (CRP), within the context of this evolutionary framework.

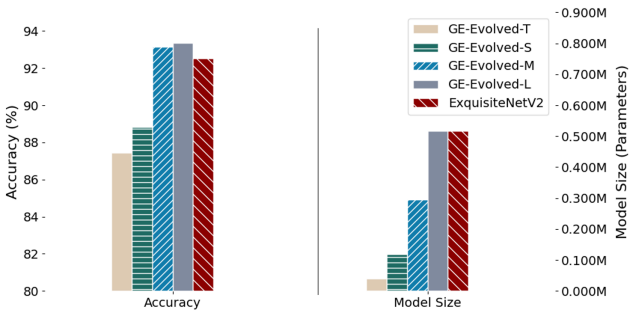

Figure 3: Metrics of evolved ExquisiteNetV2 models.

The ensuing sections will delve into the specifics of our findings. Section 4.0.1 will detail the performance metrics of the most effective models derived from this study. Subsequent to this, Section 4.0.2 will present an ablation study that explores the individual and combined influences of the EoT and CRP methodologies. This latter section aims to offer observations regarding their respective impact on the GE framework.

4.0.1 Evolved Seeds: The experimental results from our Guided Evolution approach are promising. The autonomous framework has successfully evolved ExquisiteNetV2 variants, surpassing the original model's benchmarks of 92.52% accuracy and a parameter count of 0.518230M.

A notable example is the "GE-Evolved-L" variant, achieving an accuracy of 93.34% with a parameter count of 0.518230M, aligning exactly with the original ExquisiteNetV2's size. Figure 3 provides a comparative illustration of this variant against the seed model, illustrating the capacity of our GE framework to autonomously enhance neural architectures with human-like expertise.

Moreover, significant progress was made in enhancing parameter efficiency. For instance, the "GE-Evolved-M" variant achieved a remarkable 43.1% reduction in parameter count compared to ExquisiteNetV2, alongside an improved test accuracy of 93.16%. This efficiency and performance are visually represented in Figure 3. Additionally, the "GE-Evolved-S" and "GE-Evolved-T" variants, while slightly less accurate than the seed model at 88.83% and 87.45% respectively, showcased a dramatic improvement in model size to 0.119254M and 0.039190M, illustrating a substantial advancement in model compactness without significant accuracy trade-offs.

To illustrate the performance improvements achieved through Guided Evolution, we compare ExquisiteNetV2's enhanced performance with other leading low-parameter CIFAR-10 classifiers in Figure 4. This comparison provides a contextual understanding of the advancements made by our autonomous framework. Looking ahead, we plan to expand the application of the GE methodology to a broader range of seed models in future research endeavors.

To illustrate the types of augmentations produced by the Genetic Engineering (GE) process, Figure 5 showcases an example of an augmented code block from an effective individual (GE-Evolved-M).

In addition to the notable performance improvements observed in this study, the GE framework also demonstrated potential for

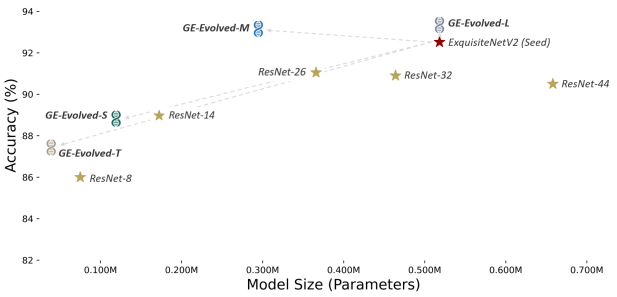

Figure 4: Evolved ExquisiteNetV2 models compared to State of the Art.

Example Seed Block

```python
class FCT(nn.Module):
    def __init__(self, cin, cout):
        super().__init__()
        self.dw = nn.Conv2d(cin, cin, 4, 2, 1, groups=cin, bias=False)
        self.maxpool = nn.MaxPool2d(2, ceil_mode=True)
        self.minpool = MinPool2d_y(2, ceil_mode=True)
        self.pw = nn.Conv2d(3*cin, cout, 1, 1, bias=False)
        self.bn = nn.BatchNorm2d(cout)

    def forward(self, x):
        x = torch.cat((
            self.maxpool(x),
            self.minpool(x),
            self.dw(x),
        ), 1)
        x = self.pw(x)
        x = self.bn(x)
        return x
```

Example Augmented Block

```python
class FCT(nn.Module):
    def __init__(self, cin, cout, dropout_rate=0.1):
        super().__init__()
        self.dw = nn.Conv2d(cin, cin, 4, 2, 1, groups=cin, bias=False)
        self.maxpool = nn.MaxPool2d(2, ceil_mode=True)
        self.minpool = MinPool2d_y(2, ceil_mode=True)
        self.pw = nn.Conv2d(3*cin, cout, 1, 1, bias=False)
        self.bn = nn.BatchNorm2d(cout)
        self.swish = nn.SiLU() # Add swish activation function
        self.dropout = nn.Dropout2d(dropout_rate) # Add dropout layer

    def forward(self, x):
        x = torch.cat((
            self.maxpool(x),
            self.minpool(x),
            self.swish(self.dw(x)), # Add swish activation function
        ), 1)
        x = self.pw(x)
        x = self.swish(self.bn(x)) # Add swish activation function
        x = self.dropout(x) # Add dropout layer
        return x
```

Figure 5: Example of an augmented code block.

further advancement over longer evolutionary periods. This extended potential can likely be attributed to the vast array of design pathways that the LLM-guided framework is capable of exploring.

It is important to highlight that the cutoff point of the evolutionary process in this study was due to time constraints and not because a significant convergence point was reached.

4.0.2 Ablation Study: A pivotal aspect of our study was assessing the impact of Evolution of Thought (EoT) and Character Role Play on the Guided Evolution process. To this end, we conducted variant experiments: one excluding EoT and another excluding both EoT and Character Role Play. The results demonstrated that the incorporation of EoT and Character Role Play substantially influenced the evolution trajectory. Notably, the inclusion of EoT accelerated the convergence towards optimal solutions. This enhancement is

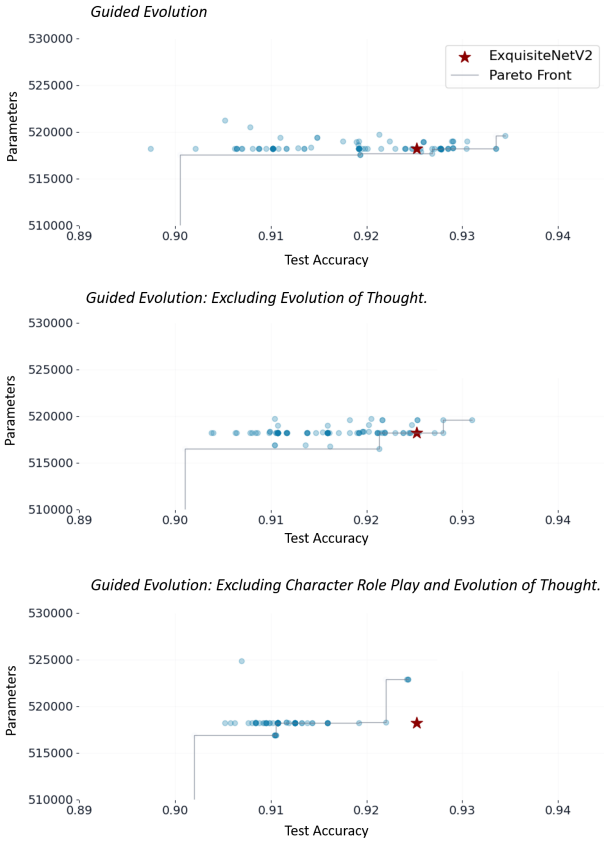

Figure 6: Pareto Frontier of Guided Evolution variants: Generation 9

graphically represented through the comparison of Pareto frontiers at generation 9, as illustrated in Figure 6.

To further elucidate the development trajectory of individuals that conformed to our primary objective—enhancing accuracy without increasing the parameter count beyond that of ExquisiteNetV2—we tracked the progression of the most accurate model at each generation within the three distinct Guided Evolution runs that maintained the same or fewer parameters than ExquisiteNetV2. The models that ultimately achieved the highest accuracy, while adhering to the parameter constraints of ExquisiteNetV2, are highlighted in Figure 7. This figure depicts the influence of the EoT and Character Role Play methodologies on the trajectory and efficiency of the Guided Evolution process.

Our findings show that the GE framework quickly identified model overfitting. Implementing various regularization methods early on led to significant improvements. While crossover facilitated horizontal transfer of updates across gene blocks, the EoT feedback mechanism enabled vertical propagation of these changes. Thus, if adding regularization to an elite example's code block X improved performance, EoT could extend these updates to another genome's block Y. Consequently, EoT ensures intrinsic feedback and synchronicity across a genome's disparate code blocks.

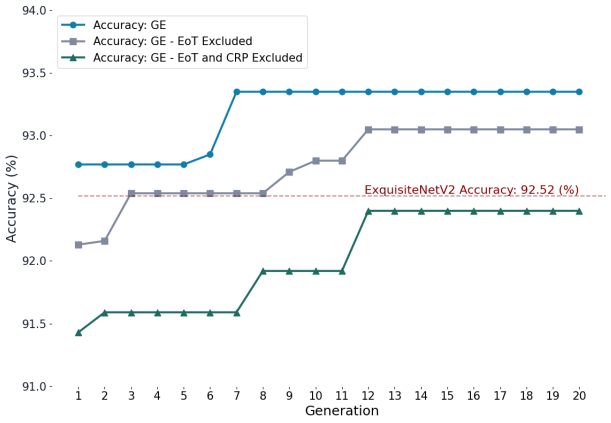

Figure 7: Guided Evolution exclusionary study.

5 CONCLUSION

In this study, we introduced Guided Evolution (GE), a novel methodology for harnessing the power of Large Language Models (LLMs) in advancing neural architectures. Central to GE are two innovative concepts: Evolution of Thought (EoT) and Character Role Play (CRP), which significantly contribute to the intelligent evolution of model architectures and infuse diversity and creativity into the evolutionary process, respectively.

Our findings, especially with models such as "GE-Evolved-L", "GE-Evolved-M", "GE-Evolved-S" and "GE-Evolved-T" demonstrate GE's potential in autonomously enhancing state-of-the-art neural architectures. These outcomes underline the efficacy of EoT and CRP within the GE framework.

It's important to note that our ablation studies, conducted with a single trial for each variant due to time constraints, serve as an initial exploration. Future work is needed to fully assess the impact of EoT and CRP within this algorithm framework.

Future research will aim to expand GE across various datasets and domains, fully harnessing its capabilities in machine learning. We will explore the impact of GE on the efficiency of Neural Architecture Search (NAS), the role of instructive commenting in code gene sequences, and the effects of different LLM models on GE's parameter space exploration.

In conclusion, our study lays a foundational framework for future exploration in GE, with the anticipation that advancements in NLP and LLM development will significantly enhance Guided Evolution's performance. This work opens new horizons for the exploitation of LLMs in genetic algorithms and multi-objective optimization, promising an exciting future for models that advance models. For access to our code, datasets, and supplementary materials, visit our GitHub repository: https://github.com/YourGitHubUsername/GE-Project. We invite the community to explore, replicate, and contribute to GE's development.

6 ACKNOWLEDGEMENTS

We thank [Name Later] for his expert guidance on genetic algorithms, which was pivotal to our research's success. His insights significantly enriched our approach, demonstrating both depth and practical applicability.

REFERENCES

[1] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823* (2017).

[2] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. 2018. Path-level network transformation for efficient architecture search. In *International Conference on Machine Learning*. PMLR, 678–687.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. https://doi.org/10.1109/4235.996017

[4] Giorgio Guariso and Matteo Sangiorgio. 2020. Improving the performance of multiobjective genetic algorithms: An elitism-based approach. *Information* 11, 12 (2020), 587.

[5] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. arXiv:2311.05232 [cs.CL]

[6] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG]

[7] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[8] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[9] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[10] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the genetic and evolutionary computation conference*. 419–427.

[11] Kevin Ma, Daniele Grandi, Christopher McComb, and Kosa Goucher-Lambert. 2023. Conceptual Design Generation Using Large Language Models. arXiv:2306.01779 [cs.CL]

[12] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. 2017. Evolving Deep Neural Networks. arXiv:1703.00548 [cs.NE]

[13] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*. PMLR, 4095–4104.

[14] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for Activation Functions. arXiv:1710.05941 [cs.NE]

[15] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 4780–4789.

[16] Dingming Yang, Zeyu Yu, Hongqiang Yuan, and Yanrong Cui. 2022. An improved genetic algorithm and its application in neural network adversarial attack. *Plos one* 17, 5 (2022), e0267970.

[17] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs.CL]

[18] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493* (2022).

[19] E. Zitzler, M. Laumanns, and L. Thiele. 2001. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems* (19–21 September 2001), K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty (Eds.). International Center for Numerical Methods in Engineering, Athens, Greece, 95–100.

[20] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

[21] Jason Zutty, Daniel Long, Heyward Adams, Gisele Bennett, and Christina Baxter. 2015. Multiple objective vector-based genetic programming using human-derived primitives. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 1127–1134.