Research Proposal: All Agents are Evolving Translators

Kevin You
Project Link: `https://blog.csdn.net/m0_62984100/article/details/140054725`

***K*eywords** Machine Learning · Interpretable Learning · Decision Maker · Lightweight · Fast Algorithm

1 Introduction

To build a GUI Agent able to adapt to users' needs or an Automated Trading System, it's important to construct an architecture that is fast, interpretable, of high performance, and professional-oriented [1]. Despite the prominent achievements of deep learning in Large Language Models (LLMs), the interpretability of neural network architectures is still poor, which affects their credibility and hence limits the deployments of risk-sensitive fields. In certain scenario-specific domains with scarce data, rapidly obtaining a large number of supervised learning labels is challenging, and the workload of manually labeling data would be enormous. Catastrophic forgetting in neural networks further leads to low data utilization rates. In situations where swift responses are vital, the density of the model makes local deployment difficult and the response time long, which is not conducive to local applications of these fields. To tackle these problems, this research is designed to construct a lightweight and interpretable architecture consisting of these 3 parts: encoder or decoder, decision-making algorithm, and continuous self-evolving mechanism.

2 Related Works

Deep learning, as a data-driven method, has drawn great attention because of its convenience in deployment. Early foundational work can be traced back to Hopfield [2], who introduced neural networks and their computational abilities. Then, Rumelhart et al. [3] pioneered the back-propagation algorithm, which remains a cornerstone in training neural networks.

Transformer-based models have revolutionized the Natural Language Processing (NLP) field, with Radford et al. [6] demonstrating the power of unsupervised multitask learning with language models, and Sanh et al. [7] introducing DistilBERT, a smaller, faster, and more efficient version of BERT. Raffel et al. [4] explored the limits of transfer learning using a unified text-to-text transformer, and Chowdhery et al. [5] scaling language modeling with the Pathways approach.

Encoding modules, as an important component of translators, have also evolved, with Sennrich [8] addressing rare word translation using subword units, and Cer et al. [9] developing a sentence encoder instead of a word encoder. Wu et al. [10] highlighted the importance of capturing graph structures in natural language processing through graph neural networks. This greatly affected our selection of techniques.

As demonstrated by Kudo [11] with SentencePiece, tokenizing languages that don't split words with spaces is possible and necessary. Tian and Zhao [12] proposed a word similarity algorithm based on Tongyici Cilin for semantic web adaptive learning systems. Dong et al. [13] explored using state space as part of the model architecture in their Hybrid-head Architecture for Small Language Models, gaining relatively high performance.

Text normalization and semantic enrichment frameworks have been developed by Jing et al. [14], enhancing text matching and thereby boosting machine reasoning capabilities. Ye et al. [15] showed that large language models are versatile decomposers, improving table-based reasoning.

Hierarchical and nested approaches have been explored, with our previous work [1] proposing a fast text normalization algorithm and semantic parsing framework, and Neitemeier et al. [16] combining byte- and word-level processing for robust language models.

To construct lightweight AI models, the reasoning ability promised a better outlook of a novel dimension of scaling law. Expert systems and human thought simulation have a rich history, with McCulloch and Pitts [17] laying the groundwork for logical calculus in nervous activity, and Newell [18] simulating human thought with the Production Rule Systems. Jacobs et al. [19] introduced adaptive mixtures of local experts, a precursor to modern mixture-of-experts (MoE) models. Shen et al. [20] combined MoE with instruction tuning for large language models.

So far, research on LLM inference has been a popular area (until January 2025 or later). Logical reasoning and chain-of-thought prompting have been shown to enhance reasoning in large language models by Wei et al. [21]. In contrast, Grosnit et al. [23] demonstrated structured reasoning achieving high-performance levels. Zhang and Liu [22] explored navigating and expanding thought space for model reasoning. Test-time training for abstract reasoning was effectively utilized by Akyürek et al. [24].

Applications of LLMs have been expanded greatly, from autonomous coding to quantitative analysis. Chen et al. [25] evaluated large language models trained on code, and Niu et al. [26] and Lu et al. [27] explored combining RPA and AI for GUI agents. Hong et al. [28] applied LLMs to build GUI agents, and Quan et al. [29] used LLMs for mathematical proof verification.

Nevertheless, interpretability has been a considerable issue in deep learning, with Ji et al. [30] surveying techniques and applications. Vaswani [31] revised the attention mechanism, while Lundberg [32] and Ribeiro et al. [33] developed SHAP and LIME for model interpretability. New neural network architectures, such as Kolmogorov-Arnold Networks (KAN) by Liu et al., have been proposed, with applications in scientific domains [34]. Knowledge graphs and few-shot learning have been advanced by Li et al. [36].

Ensemble learning techniques, including Random Forests [37], Extremely Randomized Trees [38], and various ensemble methods [39, 40, 41, 42, 43], have shown the benefits of diversity in ensemble performance. Gal and Ghahramani [44] used Monte Carlo Dropout for Bayesian approximation in machine learning.

Evolutionary algorithms and self-evolving models have been explored by Luo et al. [45] and Lee et al. [46], showing improved response and reduced costs. Reinforcement learning advancements include reward centering by Naik et al. [47] and natural language reinforcement learning by Feng et al. [48]. Shao et al. [49] pushed the limits of mathematical reasoning in open language models, emphasizing resource efficiency and enhanced reasoning capabilities.

3 Problem Formulation

We abstract the elements of sequences. From the following context, "Word" and "Sentence" are defined as follows.

Definition 3.1: Word A *word* is a sequence segment with a specific meaning that originates from a division of a sequence according to certain rules.

An n-length sequence $e_1 e_2 \cdots e_n$ is divided into $(e_1, \cdots, e_{i_1}), \cdots, (e_k, \cdots, e_{i_k})$ according to predefined principles, where $i_k = n$. Then $(e_j, \cdots, e_{i_j}),\ 1 \leq j \leq k$ is a *word.*

Take a sequence of tuples as an example:

$$(1, 2, 3)(4, 5, 6)(2, 3, 4)(5, 6, 7)$$

If we join the tuples together to form *words* according to the rule of increasing numbers between consecutive tuple intervals (in this case: 3<4 and 4<5), then both "$(1, 2, 3)(4, 5, 6)$" and "$(2, 3, 4)(5, 6, 7)$" are *words* of the above sequence.

Definition 3.2: Sentence A *sentence* is a specific transformed *word* sequence that adheres to certain rules.

If $S = (e_{j_1}, \cdots, e_{i_{j_1}}), \cdots, (e_{j_m}, \cdots, e_{i_{j_m}})$ obeying preset principles, then S is a *sentence*.

For instance, if we define that a *sentence* is exactly composed of 4 *words*, then

$$\begin{gathered}(1, 2, 3)(4, 5, 6)\\(2, 3, 4)(5, 6, 7)\\(2, 3, 4)(5, 6, 7)\\(1, 2, 3)(4, 5, 6)\end{gathered}$$

is a *sentence*. We denote it as S_1. The *sentences* S_1 at the original level can be reconsidered as *words* W_2 when we elevate our horizons to a higher level (i.e., $W_2 := S_1$). Upon ascending a level, *word* W_2 can also form a sequence, such as the repetition of W_2 five times:

$$S_2 := W_2W_2W_2W_2W_2 = S_1S_1S_1S_1S_1$$

Now, we can consider the *sentence* of the original level S_1 as a *word* W_2 at a new higher level.

When discussing *words* and *sentences*, it is advisable to indicate the level they belong to, unless the context makes it explicitly clear.

Encoders are designed to transform any sequence into a highly standardized form, i.e., a sequence of preset contents, or to identify a *word* beyond knowledge, while decoders are vice versa. Now we introduce two important concepts of highly normalized sequences.

Definition 3.3: Sentence Framework A *sentence framework* is a fixed pattern with predetermined vacancies that, after filling these vacancies with specified types of *words*, a *sentence* is composed.

Definition 3.4: Data Word A *data word* is a *word* whose patterns and contents may be diverse but can be classified by identifiers, though it can be hard to predict directly without given contexts.

Definition 3.5: Keyword *Keywords* are the *words* but not vacancies that compose *sentence frameworks*.

Given the abundance of *data words*, it is often necessary to categorize them. To classify them, we shall label them and indicate their types (they are the *subclasses* of *data words*).

In Python codes, for instance, the following commands can be found:

import numpy as np

From this, we can extract the *sentence framework*:

import [module name] as [alias]

In this text, "import" and "as" are *keywords*, while "numpy" and "np" represent the module name and the alias of the module, respectively. Both "numpy" and "np" are *data words*. Similarly, we can have:

import tensorflow as tf
import matplotlib.pyplot as plt

Both of their *sentence frameworks* are:

import [module name] as [alias]

Data words (i.e, [module name][alias]) respectively, are:

"tensorflow" "tf" and "matplotlib.pyplot" "plt"

Again, they share the same *keywords* "import" and "as".

From maneuvering computers to autonomously trading, we need an architecture that takes streaming sentences as inputs and outputs new sentences wisely, just like a music player that plays his/her role and joins in the orchestration. In essence, any dynamic decision-making process is just a translation process. Hence, we introduce the definition of *orchestration chart* below:

Definition 3.6: Orchestration Chart An *orchestration chart* is a generalized version of the Gantt chart that turns the time dimension into any other one. All tasks in a Gantt chart are called *tracks* in the perspective of *orchestration charts*. These tracks accept sequences independently.

Orchestration charts are prevalent in music composition, video creation, task allocation, time management, etc.

To keep up with the pulse of the fast-evolving world, agents should be equipped with the ability to learn continuously, just like our lifelong learning. We also need this architecture capable of accumulating knowledge while maintaining a critical mind.

4 Methodology

4.1 Perception: Parallel Nested State Spaces Framework Ensemble

We can use the *Basic-Element Theory of Extenics* coupled with other *Representation Learning* techniques to model multimodal sequences and build a universal sequence representation. Inspired by "synonyms", we use states (specifically, the nodes of such a graph) to represent these words. In a given *State Space* where all words share the same "meaning" (i.e., they are in the same *Equivalence Class*), different forms of words are considered as various states of the "meaning". In Figure 1, E_l represents the *Equivalence Relation* of level l and $W_{l,i,j}$ stands for the j_{th} word of level l that in the i_{th} equivalence class. For a fixed i, let $W_{l,i,0}$ be the corresponding *Representative Element* and all words in the same state space $S_{l,i}$ (tagged by $T_{l,i}$) are connected by edges, where red bidirectional arrows show their commonalities with auxiliary dotted edges illustrating all transformations of each word. These transformations, on which the generation rule $G_{l,i}$ is based, navigate the state generation in the same equivalence class. The state space generation step is immediately followed by the *Tokenization* step. After that, an input sequence of level l will be normalized, by turning all words to their respective representative elements, or, an output sequence of level l will be generated, by converting words into other ones in the same equivalence classes.

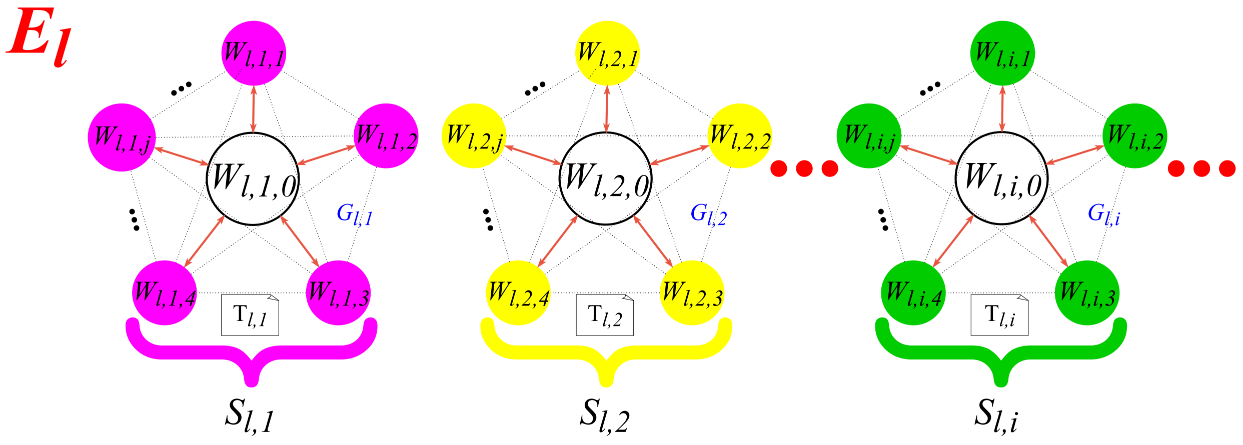

Figure 1: State Spaces of Words at Level l. Graphs from Left to Right Are $S_{l,1}$, $S_{l,2}$, ..., $S_{l,i}$, ...

As we elevate our horizons to level $l + 1$ (shown in Figure 2), the sentences (made of state spaces and made from words at level l) at level l are now words at level $l + 1$, which can be written as $W_{l+1,i,j}$. Similar state spaces can be constructed but on this elevated level. This is why this framework is called "Nested State Spaces" - they are state spaces of lower levels in state spaces of higher levels. Please refer to the right part of Figure 2 for more details.

If we parallelize these frameworks and form an ensemble, we can asynchronously output sequences of sentences on different tracks. As illustrated in the middle of Figure 3, at least m nested state spaces frameworks form an ensemble. These output sequences are without interference, like the development of parallel universes, hence the name "Parallel Nested State Spaces Framework Ensemble". In the orchestration chart of the output, $O_{m,k}$ is the k_{th} output sequence of the m_{th} nested state spaces framework, and a_m means the position number of the first output sequence seen by us to the right of the ellipses. Similarly, we denote $I_{m,k}$ as the k_{th} input sequence of the m_{th} track of the respective chart.

Such an ensemble can encode all concepts of all levels into a *Knowledge Graph*. Likewise, we have decoders (especially generative ones that need diverse outputs).

4.2 Decision Maker Teams: Believability-Weighted Expert Teams Algorithm

4.2.1 Inference Engine

In this section, we will explore *Production Rule Based Expert Systems* (or *MoE*) and other useful tools in *Automated Reasoning* including *Causal Inference*, *Mathematical Logic*, *Algebraic Propositional Logic*, etc.

A simple rule-based automated reasoning system can be expressed as:

$$[\mathbb{P}(B), V, L, Thm] \tag{1}$$

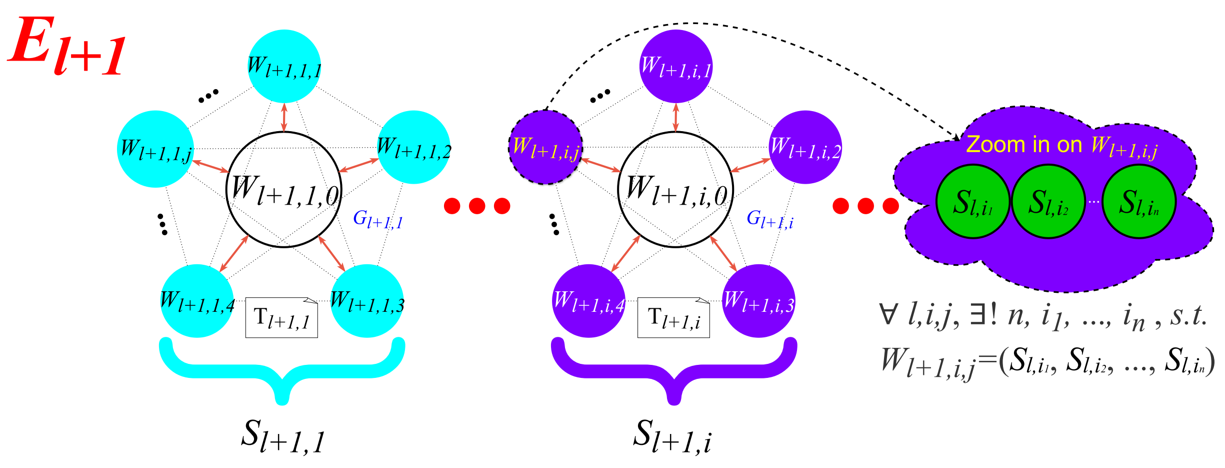

Figure 2: Nested State Spaces.

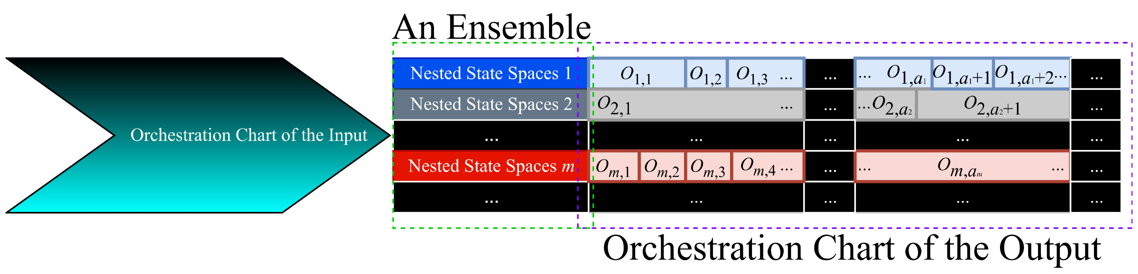
An Ensemble
Orchestration Chart of the Input
Nested State Spaces 1
Nested State Spaces 2
Nested State Spaces m
$o_{1,1}$ $o_{1,2}$ $o_{1,3}$ ...
... o_{1,a_1} o_{1,a_1+1} o_{1,a_1+2}...
$o_{2,1}$ ...
...o_{2,a_2} o_{2,a_2+1}
$o_{m,1}$ $o_{m,2}$ $o_{m,3}$ $o_{m,4}$ ...
... o_{m,a_m} ...
Orchestration Chart of the Output

Figure 3: Orchestration of Parallel Nested State Spaces Framework Ensemble.

Primarily, given a non-empty set of N_1 independent simple propositions $B = \{P_1, P_2, ..., P_{N_1}\}$, with the "$\wedge$" and "$\vee$" operations, we can yield a proposition space $\mathbb{P}$ from its base B. $V := \{-1, 0, 1\}$ is the value set that represents the state of a proposition, where -1 means "false", 0 is "not given", and 1 represents "true". $L := \{AND, OR\}$, and we will clarify L's AND and OR functions later. $Thm = \{thm_1, ..., thm_{N_2}\} \neq \emptyset$ is a finite theorem set that stores some theorems built up from the elements of $\mathbb{P}$ and is given before the algorithm starts.

To enable contrapositive reasoning and handle potential contradictions, we introduce the concept of conjugate. Define the relation between p and $\neg p$ as a "conjugate relation". So do the operations "$\wedge$" and "$\vee$", and functions AND and OR. Thus, we denote $\neg AND = OR$, $\neg OR = AND$, $\neg\wedge = \vee$, and $\neg\vee = \wedge$. Also, we let $\neg S = \{\neg x | x \in S\}$ for any set S.

To be precise, the proposition space $\mathbb{P}(B)$ is defined as follows:

Definition 4.1: Proposition Space A proposition space $\mathbb{P}(B)$ generated by B is a set satisfying these 3 properties:

1. $B \subseteq \mathbb{P}(B)$;
2. $\forall p_1, p_2 \in \mathbb{P}(B),\ p_1 \wedge p_2 \in \mathbb{P}(B),\ p_1 \vee\ p_2 \in \mathbb{P}(B)$;
3. for any set $\mathbb{P}'(B)$ satisfying both $B \subseteq \mathbb{P}'(B)$ and $\forall p_1, p_2 \in \mathbb{P}'(B),\ p_1 \wedge p_2 \in \mathbb{P}'(B),\ p_1 \vee\ p_2 \in \mathbb{P}'(B)$, it yields $\mathbb{P}(B) \subseteq \mathbb{P}'(B)$.

From the above definition, $\mathbb{P}(B)$ exists and is unique. We call $\mathbb{P}(B)$ is symmetric if $\neg\mathbb{P}(B) = \mathbb{P}(B)$. Let's illustrate the definition of AND and OR:

$$AND(\{p_{i_1}, ..., p_{i_k}\}) := p_{i_1} \wedge ... \wedge p_{i_k},\ \forall k \in \mathbb{N}^*, \forall p_{i_1}, ..., p_{i_k} \in \mathbb{P}(B) \tag{2}$$

$$OR(\{p_{i_1}, ..., p_{i_k}\}) := p_{i_1} \vee ... \vee p_{i_k},\ \forall k \in \mathbb{N}^*, \forall p_{i_1}, ..., p_{i_k} \in \mathbb{P}(B) \tag{3}$$

Then, the following property reveals what the elements of $\mathbb{P}(B)$ look like.

Property 4.1: Recursive Representation of a Proposition Given a proposition space $\mathbb{P}(B)$, $\forall p \in \mathbb{P}(B)$, we have either $p \in B$, or, $\exists M \in \mathbb{N}^*$, $\tau_1, ..., \tau_M \in L$, sets $H_1, ..., H_{M+1}, S_1, ..., S_M$, $s.t.$

- $H_1 = B$;
- $S_i \in 2^{H_i} \setminus \{\emptyset\}$, $H_{i+1} = H_i \cup \{\tau_i(S_i)\}$, $1 \le i \le M$;
- $p = \tau_M(S_M)$.

Remark: This representation is not always unique. The law of distribution can give examples.

Theorem 4.1: General De Morgan's Laws $\forall H \in 2^{\mathbb{P}(B)} \setminus \{\emptyset\}, \forall \tau \in L, \neg[\tau(H)] = (\neg\tau)(\neg H)$.

Theorem 4.2: Generation of Conjugate Proposition Space $\neg\mathbb{P}(B) = \mathbb{P}(\neg B)$.

Property 4.2: Smallest Test For any set S satisfying both $(B \cup \neg B) \subseteq S$ and $\forall p_1, p_2 \in S,\ p_1 \wedge p_2 \in S,\ p_1 \vee\ p_2 \in S,\ \neg p_1 \in S$, it yields $\mathbb{P}(B \cup \neg B) \subseteq S$.

A theorem thm satisfies this: $\forall thm \in Thm$, thm states that $P(1) \Rightarrow Q(1)$, $Q(-1) \Rightarrow P(-1)$, $(\neg Q)(1) \Rightarrow (\neg P)(1)$, and $(\neg P)(-1) \Rightarrow (\neg Q)(-1)$, where $P, Q \in \mathbb{P}(B \cup \neg B)$, and the numbers in between these parentheses exhibit the states of P, Q.

Therefore, we have Production Rule Based Expert Systems. We use the list TP to nest the thinking path of this inference engine. Denote V_0, V_c, V_{pr}, V_d as the lists that record all the states of propositions in B, where $V_0 = \{v_1, ..., v_{N_1}\}$ is the initial state list, subscripts c for "current", pr for "previous", and d for "minor deduce switch". The *inference* function takes the current state list V_c and a theorem thm as inputs, and after reasoning based on that theorem and the states of propositions, it outputs the new state list. The current state list may change or stay unchanged, depending on whether the corresponding theorem is used. The *difference* function records the change between these state lists. See the pseudocode Algorithm 1 for a complete explanation.

Algorithm 1 Production Rule Based Expert System

```
1: $V_0 \leftarrow \{v_1, ..., v_{N_1}\}$
2: $Thm \leftarrow \{thm_1, ..., thm_{N_2}\}$
3: $V_c \leftarrow V_0$
4: $TP \leftarrow$ empty list
5: repeat
6:     $V_{pr} \leftarrow V_c$
7:     for $thm$ in $Thm$ do
8:         $V_d \leftarrow V_c$
9:         $V_c \leftarrow inference(V_c, thm)$
10:        if $V_c \neq V_d$ then
11:            add $\{thm, difference(V_c, V_d)\}$ to $TP$
12:        end if
13:    end for
14: until $V_c = V_{pr}$
15: return $difference(V_c, V_0), TP$
```

To make the reasoning process more efficient, instead of using the nested loop method in Algorithm 1, we may explore more optimization algorithms such as reinforcement learning to reconstruct the thinking path.

4.2.2 Believability Weighing

Inspired by Ray Dalio's *Priciples*, we think a novel metric called *Believability* (a sort of internal evaluation mechanism similar to the attention mechanism) will be an effective tool for evaluating decisions made by all members internally in the inference process. We combine the techniques like *Voting* in *Ensemble Methods* when valuing all these decisions and making weighted decisions. Let Stc_i be the sentence outputted by the i_{th} expert $(0 \le i \le n)$, where $i = 0$ stands for the final weighted sentence, and denote w_i as the corresponding weight. Noticeably, the weights may not be simple numbers and can rely on the input instance of the whole architecture. Moreover, the thinking paths, history performances, and the architectures of the *Parallel Nested State Spaces Framework Ensembles* can greatly impact these weights. The *product* and *sum* functions below should be defined carefully.

$$Stc_0 = \textit{sum}[\textit{product}(w_i, Stc_i), 1 \leq i \leq n] \tag{4}$$

For instance, if experts are requested to perform the forecasting or classification tasks, the Stc_i and Stc_0 in Equation 4 are usually matrices of numbers. In ensemble learning classification, performance will improve when the weighted items are probabilities instead of categories. In this case, w_i is a non-negative number satisfying $\sum_{i=1}^{n} w_i = 1$. Both *product* and *sum* are self-evident functions. However, this is not often the case when generating texts.

During training, *Deep Learning* techniques (including *Dropout*) will be used to promote model robustness. Additionally, when data is difficult to acquire (especially if we don't have an evaluation set), causal inference may be an alternative evaluation instrument.

4.3 Continuous Self-Evolving Mechanism: Independent Agent Learning Algorithm

This is the component on which agents will not rely directly when making decisions (so we loosen the lightweight requirement while handling this part). Nevertheless, to consistently improve performance, agents should be equipped with a learning mechanism, as performance is entirely subject to agents' knowledge. Agents should "practice" in various real-world scenarios to truly "extract knowledge", use generative techniques to generate "hypotheses", test their validity, retain those that perform well in the dataset as knowledge, and question existing hypotheses that may be incorrect. The decision-making module works similarly. So does the perception module. Through this approach, we can create an AI brain that "learns on its own", capable of sharing knowledge with us.

Concretely, during the training process, agents can improve their perception module by:

- adding or removing one equivalence class at a particular level
- editing the generation rule of a given state space
- changing the selection of a representative element in the equivalence class
- modifying a tag of a state space

To boost their inference ability, agents will:

- append a proposition to the base of their proposition space or delete one
- emend the content of a basic proposition
- establish a theorem or break one
- revise a theorem

After a series of actions above, if agents' performance on the whole training set improves, such actions are effective. Our task is to design an algorithm giving such a series of actions. To discover potential contradictions, agents can turn again to causal inference. The contents of hypotheses and test cases can be generated by *Generative Models*, the assumption-experiment research paradigm automated. The agents then need to perform *Information Extraction* to organize the output of those generative models. Finally, we may find *Extension Logic* (a building block of *Extenics*) helpful for agents in solving potential contradictions between knowledge and facts, with *Bayesian Methods* (including *Bayesian Networks*) combined to modify both the perception module and the decision-making module that act as containers of knowledge. Error analysis can also be automated in this way.

4.4 An Overview of this Entire Architecture

This research will propose an architecture composed of 3 modules. The first is the perception module Parallel Nested State Spaces Framework Ensemble, which takes in an orchestration chart and outputs a processed orchestration chart. Each track of the output matches with 2 propositions' states (one and its conjugate) in module 2, the decision-making module Believability-Weighted Expert Teams Algorithm. These 2 modules together form the base of the architecture, which influences agents' performance directly (just like the neural network itself). Then, an independent continuous self-evolving mechanism, the Independent Agent Learning Algorithm, will train the aforementioned architecture (like the backpropagation algorithm).

5 Expected Outcomes

5.1 Frameworks and Algorithms

- *Parallel Nested State Spaces Framework Ensemble*: a sparse, knowledge-based, interpretable, and lightweight encoder or decoder for universal sequence processing
- *Believability-Weighted Expert Teams Algorithm*: a sparse, interpretable, and lightweight inference engine like an ensemble of the mixture of experts with knowledge graphs for decision-making
- *Independent Agent Learning Algorithm*: an algorithm for an agent to discover knowledge and learn continuously with a critical mind

5.2 Projects and Products

- The Natural Scripting Language "*Magic*" ("如意" in Chinese) and its intelligent development platform for fast RPA and real-time human-computer interaction can be locally deployed
- The Automated Trading System "*Pro Veritate*" ("探灵" in Chinese) can not only exhibit its decision-making process but also gain considerable profits for investors

6 Significance of This Research

6.1 Inference Interpretability

6.1.1 Expected Contributions

The black-box nature of neural network architectures undermines their credibility and limits their deployment in risk-sensitive scenarios. This research aims to improve the interpretability of these models by developing a lightweight interpretable decision-maker. This will enable humans to comprehend and correct computer decisions, fostering better collaboration between humans and machines.

6.1.2 Potential Applications

- Finance: An Automated Trading System
- Education: Teaching Students by Showing Interpretable Thinking Paths and Using Knowledge
- Research: Letting Machines Discover Knowledge for Humans
- Healthcare: Providing Trustworthy and Timely Results

6.2 Knowledge Accumulation

6.2.1 Expected Contributions

Traditional neural networks often lack mechanisms for effectively accumulating knowledge. To develop systems that can continuously update and refine their knowledge graphs, a lightweight, interpretable, and self-evolving mechanism, the "Independent Learning Algorithm", will be an alternative for local decision-making deployment. This will facilitate continuous learning, enhance decision-making capabilities, and ensure that machines can retain and utilize knowledge effectively.

6.2.2 Potential Applications

- Research
- Finance
- Government: An Automated Government Administration with Process Mining
- Robotics: Automated Robotics for Industry
- Healthcare

6.3 Execution Speed

6.3.1 Expected Contributions

In situations where rapid responses are crucial, the bulkiness of dense models can hinder local deployment and cause considerable response lags. The "Parallel Nested State Spaces", an ensemble of lightweight and sparse frameworks, helps to increase processing speed while keeping performance high.

6.3.2 Potential Applications

- Finance
- Government
- Robotics
- Healthcare
- Ads: Helping Enterprises Marketing

6.4 Robust Results

6.4.1 Expected Contributions

Catastrophic forgetting and illogical inference often result in low data utilization rates and unreliable outcomes. Ensemble inference engines "Believability-Weighted Expert Teams Algorithm", like MoE, will make the models' decision-making process robust. This improvement will promote the learning capabilities of AI systems, ensuring more reliable performance in dynamic environments.

6.4.2 Potential Applications

- Government
- Robotics
- Healthcare
- Finance
- Arts: Creating Logical Multimodal Artworks

6.5 Data Augment

6.5.1 Expected Contributions

In domains where data is hard to extract, quickly obtaining labeled examples is challenging and laborious. Rapid acquisition of various labeled data becomes feasible through the "Parallel Nested State Spaces" framework, thus reducing the need for extensive manual labeling. This will improve efficiency and introduce diversity, making AI applications more accessible in scenario-specific areas.

6.5.2 Potential Applications

- Robotics
- Healthcare
- Arts
- Ads

6.6 Model Cost

6.6.1 Expected Contributions

The high cost of AI deployment can limit its potential applications and make AI inaccessible for many users who cannot afford it. This research can make advanced AI technologies more economical, evoking a new technical revolution and productivity growth.

6.6.2 Potential Applications

- Government
- Finance
- Research
- Robotics
- Ads

References

[1] Kevin You. Digestion Algorithm in Hierarchical Symbolic Forests: A Fast Text Normalization Algorithm and Semantic Parsing Framework for Specific Scenarios and Lightweight Deployment. *arXiv preprint arXiv:2412.14054*, 2024.

[2] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[7] Victor Sanh. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[8] Rico Sennrich. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[9] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for English. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174, 2018.

[10] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.

[11] Taku Kudo. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[12] JL Tian and Wei Zhao. Words similarity algorithm based on tongyici cilin in semantic web adaptive learning system. *Journal of Jilin University*, 28(6):602–608, 2010.

[13] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. Hymba: A Hybrid-head Architecture for Small Language Models. *arXiv preprint arXiv:2411.13676*, 2024.

[14] Xie Jing, Wang Jingdong, Wu Zhenxin, Zhang Zhixiong, Wang Ying, and Ye Zhifei. Building semantic enrichment framework for scientific literature retrieval system. *Data Analysis and Knowledge Discovery*, 1(4):84–93, 2017.

[15] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184, 2023.

[16] Pit Neitemeier, Björn Deiseroth, Constantin Eichenberg, and Lukas Balles. Hierarchical Autoregressive Transformers: Combining Byte- and Word-Level Processing for Robust, Adaptable Language Models. *arXiv preprint arXiv:2501.10322*, 2025.

[17] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[18] Allen Newell. Gps, a program that simulates human thought. *Computers and Thought*, 1963.

[19] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[20] Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705*, 2023.

[21] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[22] Jinghan Zhang and Kunpeng Liu. Thought space explorer: Navigating and expanding thought space for large language model reasoning. In *2024 IEEE International Conference on Big Data (BigData)*, pages 8259–8251, 2024.

[23] Antoine Grosnit, Alexandre Maraval, James Doran, Giuseppe Paolo, Albert Thomas, Refinath Shahul Hameed Nabeezath Beevi, Jonas Gonzalez, Khyati Khandelwal, Ignacio Iacobacci, Abdelhakim Benechehab, et al. Large language models orchestrating structured reasoning achieve Kaggle grandmaster level. *arXiv preprint arXiv:2411.03562*, 2024.

[24] Ekin Akyürek, Mehul Damani, Linlu Qiu, Han Guo, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for abstract reasoning. *arXiv preprint arXiv:2411.07279*, 2024.

[25] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[26] Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. Screenagent: A vision language model-driven computer control agent. *arXiv preprint arXiv:2402.07945*, 2024.

[27] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024.

[28] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

[29] Xin Quan, Marco Valentino, Louise A Dennis, and André Freitas. Verification and Refinement of Natural Language Explanations through LLM-Symbolic Theorem Proving. *arXiv preprint arXiv:2405.01379*, 2024.

[30] Shouling Ji, Jinfeng Li, Tianyu Du, and Bo Li. A survey on techniques, applications and security of machine learning interpretability. *Journal of Computer Research and Development*, 56(10):2071–2096, 2019.

[31] Ashish Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[32] Scott Lundberg. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[34] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[35] Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. Kan 2.0: Kolmogorov-arnold networks meet science. *arXiv preprint arXiv:2408.10205*, 2024.

[36] Zhuofeng Li, Haoxiang Zhang, Qiannan Zhang, Ziyi Kou, and Shichao Pei. Learning from novel knowledge: Continual few-shot knowledge graph completion. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1326–1335, 2024.

[37] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282, 1995.

[38] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[39] Gilles Louppe and Pierre Geurts. Ensembles on random patches. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I 23*, pages 346–361, 2012.

[40] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[41] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[42] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine learning*, 36:85–103, 1999.

[43] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[44] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[45] Run Luo, Haonan Zhang, Longze Chen, Ting-En Lin, Xiong Liu, Yuchuan Wu, Min Yang, Minzheng Wang, Pengpeng Zeng, Lianli Gao, et al. Mmevol: Empowering multimodal large language models with evol-instruct. *arXiv preprint arXiv:2409.05840*, 2024.

[46] Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving Deeper LLM Thinking. *arXiv preprint arXiv:2501.09891*, 2025.

[47] Abhishek Naik, Yi Wan, Manan Tomar, and Richard S Sutton. Reward Centering. *arXiv preprint arXiv:2405.09999*, 2024.

[48] Xidong Feng, Ziyu Wan, Haotian Fu, Bo Liu, Mengyue Yang, Girish A Koushik, Zhiyuan Hu, Ying Wen, and Jun Wang. Natural language reinforcement learning. *arXiv preprint arXiv:2411.14251*, 2024.

[49] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[50] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

[51] Vittoria Dentella, Fritz Günther, Elliot Murphy, Gary Marcus, and Evelina Leivada. Testing ai on language comprehension tasks reveals insensitivity to underlying meaning. *Scientific Reports*, 14(1):28083, 2024.

[52] Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024.