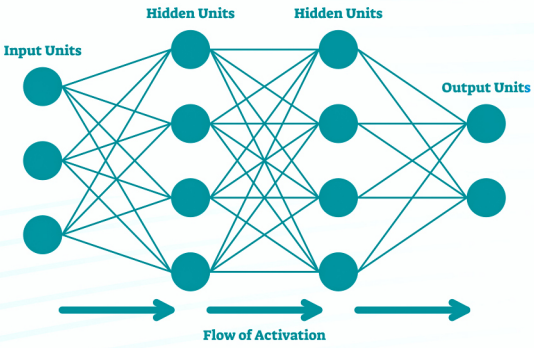
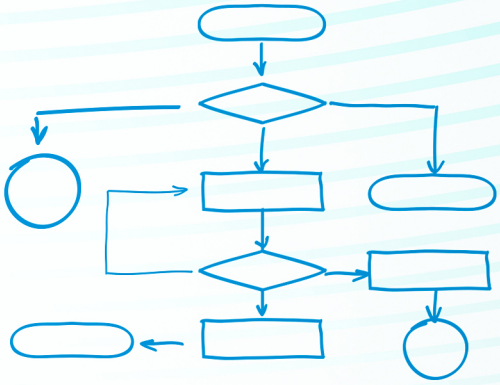
Neuro-Symbolic AI





Fusing Neural Networks and Reasoning
for Accelerating Artificial General Intelligence

MD AZIMUL HAQUE

Neuro symbolic AI

Fusing Neural Networks and Reasoning for Accelerating Artificial General Intelligence

Md Azimul Haque

This book is available at https://leanpub.com/neurosymbolicai

This version was published on 2026-06-29

Contents

Foreward

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

The Basics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Previewing and publishing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Basic formatting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Markdown and Markua

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Generate a preview version of your book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Either read a tutorial, or just go for it!

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Read the tutorial...

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

...or just go for it!

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Thanks for being a Leanpub author!

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/neurosymbolicai.

Writing in Markua

Writing in Markua is easy! You can learn most of what you need to know with just a few examples.

To make *italic text* you surround it with single asterisks. To make **bold text** you surround it with double asterisks.

Section One

You can start new sections by starting a line with two # signs and a space, and then typing your section title.

Sub-Section One

You can start new sub-sections by starting a line with three # signs and a space, and then typing your sub-section title.

Including a Chapter in the Sample Book

At the top of this file, you will also see a line at the top:

```
1 {sample: true}
```

Leanpub has the ability to make a sample book, which interested readers can download or read online. If you add this line above a chapter heading, then when you publish your book, this chapter will be included in a separate sample book for these interested readers.

Links

You can add web links easily.

Here's a link to the Leanpub homepage[1].

Images

You can add an image to your book in a similar way.

First, add the image to the "Resources" folder for your book. You will find the "Resources" folder under the "Manuscript" menu to the left.

If you look in your book's "Resources" folder right now, you will see that there is an example image there with the file name "palm-trees.jpg". Here's how you can add this image to your book:

[1] https://leanpub.com

If you want to add a figure title, you put it in quotes:

Figure 1. Palm Trees

If you want to add descriptive alt text, which is good for accessibility, you put it between the square brackets:

Figure 2. Palm Trees

You can also set the alt text and/or the figure title in an attribute list:

Figure 3. Palm Trees

Finally, if no title is provided, and the `alt-title` document setting is the default of `all`, the alt text will be used as the figure title instead of as alt text.

Figure 4. Palm Trees

You can set the important document settings at Settings > Generation Settings.

Lists

Numbered Lists

You make a numbered list like this:

1. kale
2. carrot
3. ginger

Bulleted Lists

You make a bulleted list like this:

- kale
- carrot
- ginger

Definition Lists

You can even have definition lists!

term 1
: definition 1a

: definition 1b

term 2
: definition 2

Code Samples

You can add code samples really easily. Code can be in separate files (a "local" resource) or in the manuscript itself (an "inline" resource).

Local Code Samples

Here's a local code resource:

Figure 5. Hello World in Ruby

```
1 puts "hello world"
```

Inline Code Samples

Inline code samples can either be spans or figures.

A span looks like `puts "hello world"` this.

A figure looks like this:

```
puts "hello"
```

You can also add a figure title using the title attribute:

Figure 6. Hello World in Ruby

```
puts "hello"
```

Tables

You can insert tables easily inline, using the GitHub Flavored Markdown (GFM) table syntax:

Header 1	Header 2
Content 1	Content 2
Content 3	Content 4 Can be Different Length

Tables work best for numeric tabular data involving a small number of columns containing small numbers:

Central Bank	Rate
JPY	-0.10%
EUR	0.00%
USD	0.00%
CAD	0.25%

Definition lists are preferred to tables for most use cases, since reading a large table with many columns is terrible on phones and since typing text in a table quickly gets annoying.

Math

You can easily insert math equations inline using either spans or figures.

Here's one of the kinematic equations $d = v_i t + \frac{1}{2}at^2$ inserted as a span inside a sentence.

Here's some math inserted as a figure.

$$\left|\sum_{i=1}^{n} a_i b_i\right| \le \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$$

Figure 7. Something Involving Sums

Headings

Markua supports both of Markdown's heading styles.

The preferred style, called atx headers, has the following meaning in Markua:

```
{class: part}
# Part

This is a paragraph.

# Chapter

This is a paragraph.

## Section

This is a paragraph.

### Sub-section

This is a paragraph.

#### Sub-sub-section

This is a paragraph.

##### Sub-sub-sub-section

This is a paragraph.

###### Sub-sub-sub-sub-section

This is a paragraph.
```

Note the use of three backticks in the above example, to treat the Markua like inline code (instead of actually like headers).

The other style of headers, called Setext headers, has the following headings:

```
{class: part}
Part
====

This is a paragraph.

Chapter
=======

This is a paragraph.

Section
-------

This is a paragraph.
```

Setext headers look nice, but only if you're only using chapters and sections. If you want to add sub-sections (or lower), you'll be using atx headers for at least some of your headers. My advice is to just use atx headers all the time. (The `{class: part}` attribute list on a chapter header to make a part header does actually work with Setext headers, but it's really ugly.)

Note that while it is confusing and ugly to mix and match using atx and Setext headers for chapters and sections in the same document, you can do it. However, please don't.

Block quotes, Asides and Blurbs

> Block quotes are really easy too.
>
> —Peter Armstrong, *Markua Spec*

> Asides are useful for longer text.
>
> But typing them like this isn't fun.

> Asides can be written this way, since adding a bunch of A> stuff at the beginning of each line can get annoying with longer asides.

Blurbs are useful

Blurbs are useful

There are many types of blurbs, which will be familiar to you if you've ever read a computer programming book.

This is a discussion.

You can also specify them this way:

This is a discussion

This is an error.

This is information.

This is a question. (Not a question in a Markua course; those are done differently!)

This is a tip.

This is a warning.

This is an exercise. (Not an exercise in a Markua course; those are done differently!)

Good luck, have fun!

If you've read this far, you're definitely the right type of person to be here!

Our last piece of advice is simple: once you have a couple chapters completed, publish your book in-progress!

This approach is called Lean Publishing. It's why Leanpub is called Leanpub.

If you want to learn more about Lean Publishing, read this[2] or watch this[3].

#Chapter 1: Introduction to Neuro-Symbolic AI

Motivation and Historical Context

Early Days of AI: Symbolic Approaches (1950s-1980s)

In the early days of artificial intelligence (AI), spanning from the 1950s through the 1980s, AI researchers believed that human intelligence could be replicated by manipulating symbols and logic. This period is known as the "symbolic era" of AI, where the goal was to mimic human reasoning by using structured, rule-based systems.

One of the standout innovations of this time was LISP, a programming language developed in the late 1950s by John McCarthy and his team at MIT. Why LISP? It was designed to explore AI systems that could understand and act on human-like instructions. McCarthy's vision was a system that could process "common sense" knowledge through logic and handle commands written in a natural, human-friendly way (McCarthy, 1960). To achieve this, LISP introduced a way of manipulating symbols—like numbers or words—through lists and recursive functions. While these ideas may sound complex, the breakthrough was that LISP allowed AI systems to handle abstract, human concepts with ease. It was groundbreaking at the time, and it remains one of the longest-standing programming languages in AI today.

Another key development from this era was symbolic model checking. Introduced much later in 1996 by Clarke and his colleagues,

[2] https://leanpub.com/lean/read
[3] https://youtu.be/ozO0kOnqmyA

it became a game changer for verifying whether AI systems were working as intended (Clarke et al., 1996). Think of it as an advanced checklist. Imagine you had a system with millions of moving parts—literally up to 10^30 possible states! Symbolic model checking made it possible to verify systems of that scale, quickly and accurately. This technique became essential in fields like circuit design and communication protocols, where even the smallest mistake could cause a massive failure. And the best part? It worked fast—checking through these massive systems in mere minutes. Over time, symbolic model checking evolved to handle even more complex verification tasks, like ensuring timing precision or managing systems at the word level (Clarke et al., 1996).

So, while the early days of AI were dominated by these symbolic methods, they laid the groundwork for everything that came after. LISP and symbolic model checking may seem like historical footnotes, but they represent the building blocks of AI, enabling the more sophisticated and dynamic systems we see today.

Symbolic AI's focus on logic, rules, and formal reasoning

Symbolic AI, often referred to as classical AI or GOFAI (Good Old-Fashioned AI), is the branch of artificial intelligence that focuses on using logic and explicit rules to solve problems. Think of it as the "logic-first" approach to AI, where everything revolves around well-defined rules and formal reasoning.

At its heart, Symbolic AI manipulates symbols—representations of real-world objects, concepts, or ideas—using a structured set of rules. The key here is that every problem is tackled by breaking it down into a logical sequence of steps. For example, Frame Logic (F-logic) is one of the formal systems used to represent knowledge in symbolic AI. It's designed to handle the complexities of object-oriented languages by managing things like inheritance and polymorphism (Kifer et al., 1995).

In a similar vein, propositional dynamic logic is another tool in the

Symbolic AI toolbox. It's used to check if programs are behaving as expected, ensuring they run correctly and terminate when they're supposed to (Fischer & Ladner, 1979). Picture this like a safety check for computer programs, ensuring everything is in order before hitting "go."

Symbolic AI also has a knack for reasoning through time and events. Take temporal reasoning, for example—it's used to verify programs that run in sequence or parallel (Pnueli, 1977). It's like figuring out the steps in a recipe, whether you're cooking one dish or preparing a whole banquet at once.

Now, let's talk about Binary Decision Diagrams (BDDs), which are used for representing and analyzing logical decisions. They're especially useful for things like digital circuits and systems where you need to manage a lot of "yes" or "no" decisions efficiently (Bryant, 1992). Think of BDDs as the ultimate problem-simplifying machine, helping to break down complex logic into a manageable flow.

Despite its power, Symbolic AI has its challenges. One big question is whether human thought is best modeled through symbolic rules or through patterns and associations, much like how our brains might work. There's even evidence of cultural differences in reasoning, with some studies showing that Western cultures lean toward logical, rule-based thinking, while Eastern cultures favor more holistic approaches (Nisbett et al., 2001). This just goes to show that even in AI, there's more than one way to think about thinking!

Key Breakthroughs in Symbolic AI: Expert Systems and Logic-Based Problem Solving

When we talk about symbolic AI, two major areas stand out: expert systems and logic-based problem solving. These breakthroughs paved the way for much of the intelligent reasoning we see today in AI.

Expert systems were an exciting development in AI. Essentially,

these systems were designed to mimic how human experts think and solve problems. One of the key breakthroughs here was the introduction of fuzzy logic by Lotfi Zadeh in 1983. Now, this isn't the kind of "fuzzy" where everything's unclear! Instead, fuzzy logic helps deal with uncertainty—something real-world experts face all the time. It lets systems reason with vague or incomplete information, just like humans do when they're not 100% sure about something. This made expert systems far more practical and powerful because they could handle situations where data wasn't black and white.

On the logic side of things, there's been tremendous progress as well. For example, Ordered Binary Decision Diagrams (OBDDs), introduced by Bryant in 1992, offered a new way to represent complex Boolean functions in a structured and efficient manner. Imagine trying to figure out if a certain set of conditions is true or false—that's where Boolean logic comes in. OBDDs simplified this process by representing it visually as a directed graph. This method wasn't just about making things faster—it allowed AI to tackle much larger and more complex problems, especially in areas like digital system design and mathematical logic.

Another game-changing development was symbolic model checking, a formal verification method that allows AI to analyze and verify massive systems. Think of it like running a super-fast search that can analyze trillions of possible states in a system in just minutes (Clarke et al., 1996). This made it possible to verify whether complex systems—like those used in airplanes or critical software—would behave as expected. The amazing part? Extensions like timing analysis and abstraction techniques meant this method could handle even more intricate tasks, further expanding the reach of symbolic AI in real-world applications.

Limitations of Symbolic AI: Inflexibility and Lack of Learning from Data

Symbolic AI has certainly played a key role in early artificial intel-

ligence development, especially in areas like logic-based reasoning and expert systems. However, despite its strengths, it comes with a set of limitations that make it less adaptable in today's data-driven world.

One of the biggest challenges with symbolic AI is its inflexibility. Imagine trying to analyze time-series data (like stock prices or weather patterns) using symbolic representations. According to Lin et al. (2007), symbolic methods often carry the same complexity as the original data itself. This means that as the data grows in size or dimensionality, symbolic AI struggles to keep up, making it inefficient for tasks that require scalability.

Furthermore, symbolic representations don't always line up well with the real-world data they aim to capture. As Lin et al. (2007) pointed out, when we try to calculate similarities (or "distances") between symbolic representations, those calculations don't always match up with how similar the original data points truly are. So, the symbolic model might miss important patterns or connections because it's unable to properly "understand" the data.

That said, researchers have been working on creative solutions to address these issues. For example, Lin et al. (2007) proposed a new way to represent time-series data symbolically, but with one big improvement: their method allows for dimensionality reduction. In simple terms, it can make the data smaller and more manageable, while still preserving the key patterns. Lin et al. (2003) took this idea further by creating a symbolic representation specifically for streaming data—constantly updating data like live stock prices—which also managed to shrink the data while keeping important relationships intact.

In short, while symbolic AI alone doesn't "learn" from data in the way machine learning does, researchers are finding ways to combine symbolic methods with modern techniques. These hybrid approaches could help overcome symbolic AI's rigidity while still maintaining its ability to explain decisions and reason logically,

as noted by Bassett et al. (2011) and Dede (2009). This blend of symbolic reasoning and data-driven learning is at the heart of neuro-symbolic AI, where the strengths of both worlds come together.

Rise of neural networks: Early experiments and setbacks (1960s-1980s) The story of neural networks is one of early promise, long pauses, and eventual breakthroughs. The journey really started back in the 1960s, when researchers first experimented with artificial neural networks (ANNs). However, despite the excitement surrounding these early efforts, progress was slow. Computing power was nowhere near what it is today, and large datasets were practically nonexistent, which made it difficult to train these networks effectively (Rawat & Wang, 2017).

Imagine trying to build a skyscraper without the proper tools or materials—that's what it was like for early neural network researchers. They faced challenges on multiple fronts. One big obstacle was the lack of a good training method. The early models struggled to solve complex, real-world problems because they couldn't learn in a way that would make them useful beyond simple tasks (Basheer & Hajmeer, 2000; Zhang et al., 1998).

Despite these setbacks, researchers didn't give up. The 1980s brought a glimmer of hope with the introduction of the back-propagation algorithm, a method that made training multi-layer networks much more manageable (Basheer & Hajmeer, 2000). This was a key breakthrough, allowing networks to start tackling more difficult problems. Although neural networks wouldn't hit their stride until much later, this period laid the groundwork for the remarkable progress that followed in the 2000s (Rawat & Wang, 2017).

By the time the mid-2000s arrived, neural networks were ready for their big comeback. With the rise of more powerful computers and the availability of massive datasets, what was once theoretical became practical. The resilience and determination of those early

researchers paid off, setting the stage for the neural network explosion we see today, especially after 2012 (Rawat & Wang, 2017).

Citations John McCarthy. 1960. Recursive functions of symbolic expressions and their computation by machine, Part I. Commun. ACM 3, 4 (April 1960), 184–195. https://doi.org/10.1145/367177.367199

Clarke, E., McMillan, K., Campos, S., Hartonas-Garmhausen, V. (1996). Symbolic model checking. In: Alur, R., Henzinger, T.A. (eds) Computer Aided Verification. CAV 1996. Lecture Notes in Computer Science, vol 1102. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61474-5_93

Sloman, S. A. (1996). The empirical case for two systems of reasoning. Psychological Bulletin, 119(1), 3–22. https://doi.org/10.1037/0033-2909.119.1.3

Michael Kifer, Georg Lausen, and James Wu. 1995. Logical foundations of object-oriented and frame-based languages. J. ACM 42, 4 (July 1995), 741–843. https://doi.org/10.1145/210332.210335

Michael J. Fischer, Richard E. Ladner, Propositional dynamic logic of regular programs, Journal of Computer and System Sciences, Volume 18, Issue 2, 1979, Pages 194-211, ISSN 0022-0000, https://doi.org/10.1016/0022-0000(79)90046-1.

A. Pnueli, "The temporal logic of programs," 18th Annual Symposium on Foundations of Computer Science (sfcs 1977), Providence, RI, USA, 1977, pp. 46-57, doi: 10.1109/SFCS.1977.32.

Randal E. Bryant. 1992. Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Comput. Surv. 24, 3 (Sept. 1992), 293–318. https://doi.org/10.1145/136035.136043

Sloman, S. A. (1996). The empirical case for two systems of reasoning. Psychological Bulletin, 119(1), 3–22. https://doi.org/10.1037/0033-2909.119.1.3

Nisbett, R. E., Peng, K., Choi, I., & Norenzayan, A. (2001). Culture and systems of thought: Holistic versus analytic cognition. Psychological Review, 108(2), 291–310. https://doi.org/10.1037/0033-295X.108.2.291

The MIT Press, Introduction to Statistical Relational Learning, ISBN electronic: 9780262256230, 2007. https://doi.org/10.7551/mitpress/7432.001.0001

Zadeh, L. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. Fuzzy Sets and Systems, 11(1–3), 199–227. https://doi.org/10.1016/s0165-0114(83)80081-5

Bryant, R. E. (1992). Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys, 24(3), 293–318. https://doi.org/10.1145/136035.136043

Clarke, E., McMillan, K., Campos, S., & Hartonas-Garmhausen, V. (1996). Symbolic model checking. In Lecture notes in computer science (pp. 419–422). https://doi.org/10.1007/3-540-61474-5_93

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007b). Experiencing SAX: a novel symbolic representation of time series. Data Mining and Knowledge Discovery, 15(2), 107–144. https://doi.org/10.1007/s10618-007-0064-z

Bassett, D. S., Wymbs, N. F., Porter, M. A., Mucha, P. J., Carlson, J. M., & Grafton, S. T. (2011). Dynamic reconfiguration of human brain networks during learning. Proceedings of the National Academy of Sciences, 108(18), 7641–7646. https://doi.org/10.1073/pnas.1018985108

author: MD_AZIMUL_HAQUE date: 2025-08-26 identifier: “urn:uuid:a833ebd1-a23e-47fc-be71-0461d540f542” language: en-US title: Neuro-Symbolic_AI

[]{#Neuro-Symbolic-AI-EPUB.xhtml}

Basic-Text-Frame [Neuro-Symbolic AI]

[ ]

[

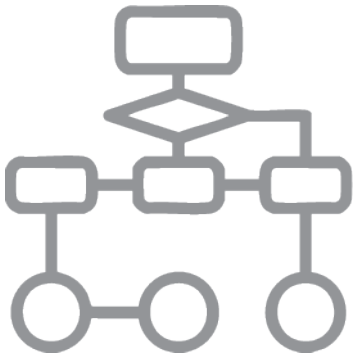


Figure 8. 164155.png

]

[ ]

[ ]

Fusing Neural Networks and Reasoning for Accelerating Artificial General Intelligence

MD AZIMUL HAQUE

[]{#Neuro-Symbolic-AI-EPUB-1.xhtml}

Basic-Text-Frame While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

[]{#Neuro-Symbolic-AI-EPUB-2.xhtml}

Basic-Text-Frame [Foreword]{.Bold}

Figure 9. 161427.png

In a world where machine learning has evolved from rudimentary algorithms to transformative, intelligent systems, neurosymbolic AI emerges as the next groundbreaking frontier. I still remember the early days when machine learning was merely a curious blend of statistics and computing power, an embryonic form of a vision that today is reshaping industries. Now, as we stand at the precipice of achieving Artificial General Intelligence (AGI), the urgency to integrate structured reasoning with deep learning has never been more palpable.

With over 14 years of hands-on experience leading transformative projects at global institutions, and as an IEEE Senior Member, I have witnessed firsthand the relentless evolution of our field. My journey has been driven by the conviction that true intelligence in machines is achieved not by isolated techniques, but through their harmonious integration.

This book encapsulates that conviction. It bridges the power of symbolic logic with the adaptive prowess of neural networks, laying the foundational bedrock for AGI. We have only scratched the surface of what AI can achieve, and integrating symbolic reasoning is the key to unlocking true general intelligence. AGI will unlock unprecedented potential in solving real-world problems and the integration of learning and reasoning is the key to unlocking the true potential of AI.

This foreword is an invitation to you, the reader, to embark on a journey into a realm where theory meets practice. Within these pages, you will discover innovative methodologies, strategies, and perspectives. Whether you are an industry veteran, an academic researcher, or an aspiring entrepreneur, this book offers a transformative roadmap, guiding you to harness the full potential of neurosymbolic AI as the cornerstone of AGI.

I encourage you to dive in, explore the rich insights contained herein, and join the vanguard of those who are poised to redefine the future of intelligence. Let us together embrace the transformative potential of this new era in AI, where every breakthrough is a step closer to machines that think, learn, and understand as profoundly as we do.

Welcome to the future of artificial intelligence.

[]{#Neuro-Symbolic-AI-EPUB-3.xhtml}

Basic-Text-Frame [Table of Contents]{.Bold}

Figure 10. 161430.png

Figure 11. 161436.png

[]{#Neuro-Symbolic-AI-EPUB-4.xhtml#x1.1-Motivation-and-Historical-Context}[]{#Neuro-Symbolic-AI-EPUB-4.xhtml#x.161841}1.1 Motivation and Historical Context

Thanks to modern hardware like GPUs and TPUs, we can now train deep neural networks on enormous amounts of data. This has led to breakthroughs in tasks such as image recognition and natural language processing. However, these deep models often lack transparency, as they deliver results without explaining their reasoning. As a result, artificial Intelligence is at a turning point.

[The Rise of Neuro-Symbolic AI]{.Bold}

Instead of relying solely on deep learning (which excels at finding patterns in huge datasets but acts like a "black box"), there's a renewed interest in [neuro-symbolic AI]. The goal is to build AI systems that are not only powerful and accurate but also interpretable and capable of reasoning in a human-like way. This hybrid approach combines [neural networks] and [symbolic reasoning]. Neural networks are great at processing raw data (like images and text) and recognizing complex patterns. Symbolic reasoning uses explicit, human-readable rules and logic to represent knowledge and make decisions. This allows the AI to learn from vast datasets and also incorporate existing knowledge in a structured way, that go beyond mere pattern recognition.

[Divergent Paths in Early AI Research]{.Bold}

In the beginning, AI research split into two groups. One group focused on symbolic AI, which favored using explicit rules and logical reasoning. The other group developed neural networks that learned from data. Each had its strengths and weaknesses. Symbolic systems were rigid and couldn't learn new information easily, while neural networks often lacked transparency and clear reasoning. Over time, researchers realized that combining these

methods could overcome their individual limitations. Advances in computing power and new algorithms made it practical to integrate learning (from data) with reasoning (using logic). Modern systems are now being designed to not only learn efficiently from vast amounts of data but also to explain and reason about their decisions, making them more robust and trustworthy.

[Insights from Cognitive Science and Linguistics]{.Bold}

Research in cognitive science and linguistics gives us insights into how humans think and communicate. Cognitive science explores how we perceive, remember, and reason, which helps design AI that mimics human thought processes. Linguistics helps AI understand and process language, including context and nuance. By embedding principles from these fields into AI systems, neuro-symbolic approaches can handle language, abstract reasoning, and decision-making in a way that is more similar to human intelligence.

[Comparing Neuro-Symbolic AI with Probabilistic Graphical Models]{.Bold}

It's helpful to see how neuro-symbolic AI compares against another similar approach known as [Probabilistic graphical models (PGMs)]. PGMs represent uncertainties using graphs. However, it can become very complex and can be hard to scale with high-dimensional data. Neuro-symbolic AI, on the other hand, can learn directly from data with neural networks. In addition to this, it can handle unstructured information (like text and images) more efficiently.

[Applications and Practical Benefits]{.Bold}

A neuro-symbolic system can analyze medical images and lab results, and then use the symbolic rules to interpret findings in the context of known medical guidelines. This not only improves diagnostic accuracy but also makes the decision process transparent for clinicians. Neuro-symbolic models offer transparency by showing clear decision paths. They can even incorporate ethical guidelines

directly into their reasoning processes, ensuring AI systems act in line with societal values and legal requirements (like the GDPR). AI systems usually need massive, high-quality datasets to learn well. Gathering such data can however be expensive and at times can be impractical. This is more so evident in fields such as healthcare and cybersecurity. If we include explicit symbolic knowledge, such as rules, ontologies, and heuristics into the learning process, the system can achieve better results even with less data. The symbolic part can help the AI "fill in the blanks" using logical inference and domain knowledge

[]{#Neuro-Symbolic-AI-EPUB-4.xhtml#x1.2-Defining-Neuro-Symbolic-AI}[]{#Neuro-Symbolic-AI-EPUB-4.xhtml#x.161844}1.2 Defining Neuro-Symbolic AI

[Core Building Blocks]{.Bold}

The basic building blocks of neuro-symbolic AI are Hybrid Architecture, Differentiable Reasoning, Object-Centric Representations. [Hybrid architecture] design fuses neural networks (for data learning) with symbolic systems (for logical reasoning) and acts as a two-in-one system that uses the best of both worlds.

[Differentiable reasoning] makes the reasoning part "differentiable", which allows it to be fine-tuned using the same mathematical methods that train neural networks, enabling the whole system to be optimized together.

[Object-centric representations] work by representing data and the relationships between them as objects. This makes reasoning more natural, much like how we understand a scene by identifying objects and their interactions.

[Coupling Neural Networks and Symbolic Reasoning]{.Bold}

If we consider the type of coupling between neural networks and symbolic reasoning, there are 3 types of coupling, known as loose coupling, tight integration, and unified architectures. In a [loosely coupled system], the neural part and the symbolic part

work separately. They talk to each other using clear interfaces. For example, A neural network might look at raw data (like images or text) to detect patterns. Then it sends these patterns to a symbolic module that makes higher-level decisions using logical rules.

[Tight integration] works by creating "distributed" representations of logical ideas directly inside the network. However, it makes the system less modular and harder to interpret.

[Unified architectures] fully merge neural and symbolic processes into a single system and one cohesive framework. In such systems, neural networks handle tasks like processing raw sensory data (e.g., images or sounds), while the same framework uses symbolic logic for complex reasoning and decision-making. It can lead to superior performance, however, designing these systems is very complex.

[Innovative Approaches in Neuro-Symbolic Learning]{.Bold}

If we consider how machines learn from data with how they reason logically, there are a few innovative approaches. [Neural systems with symbolic latent structures] use techniques like reinforcement learning to let the AI pick up symbolic relationships from data, even if there isn't direct supervision.

[Neuro-symbolic forward reasoner (NSFR)] uses differentiable forward-chaining reasoning along with object-centric learning and can make logical inferences from raw inputs (like images or text) by weighing different logical rules, effectively bridging the gap between what it perceives and how it reasons.

[Neuro-Symbolic Inductive Learner (NSIL)] trains neural networks to discover abstract concepts hidden in raw data and, at the same time, learns symbolic rules that link these concepts to the desired outcomes. This dual training ensures that both the learning and reasoning parts improve together, leading to outcomes that are both accurate and explainable.

[Knowledge-infused learning (KiL)] involves adding explicit, human-curated knowledge, such as domain-specific rules or

ontologies, directly into the deep learning process. This method improves performance and makes the AI's decisions more understandable and useful in fields that require strict adherence to regulatory standards.

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml}

Basic-Text-Frame []{#Neuro-Symbolic-AI-EPUB-5.xhtml#Chapter-2-}[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161848}Chapter 2:

Foundations of Neural Networks

Figure 12. 162574.png

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.1-Basics-of-Neural-Networks}[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161851}2.1 Basics of Neural Networks

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.1.1-Neurons—Activation-Functions—and-Layers}[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161854}2.1.1 Neurons, Activation Functions, and Layers

The foundation of neural networks lies in the concept of the [neuron], which processes information and passes it on to other neurons. In the early days of artificial intelligence, the McCulloch-Pitts neuron was introduced as a simple model of a biological neuron. It operated on binary inputs and produced a binary output, effectively functioning as a logic gate. This laid the groundwork for neural networks. However, it could not handle continuous inputs or learn from data, as it lacked adjustable weights and used a fixed threshold for activation. Modern neurons overcome these limitations by incorporating continuous activation functions and differentiable properties, allowing them to learn and adapt through processes like gradient descent.

[Activation functions] introduce non-linearity into the network,

enabling it to model complex patterns and relationships in data. These are mathematical equations that determine a neuron's output based on its input. The choice of activation function can significantly impact a neural network's ability to learn from data. Below are some examples of activation functions.

[• Sigmoid Function] squashes input values into a range between 0 and 1. It's defined as [σ]). It suffers from vanishing gradients, which can hinder learning in deep networks.

[• Tanh Function], otherwise known as hyperbolic tangent function has output values between -1 and 1. It addresses some limitations of the sigmoid but still faces vanishing gradient issues.

[• ReLU (Rectified Linear Unit)]is defined as f(x)=max[]{.Times}(0,x), with outputs zero for negative inputs and the input itself, when positive. It helps mitigate the vanishing gradient problem. However, it can suffer from "dead neurons", which stop learning if they consistently receive negative inputs.

[• Leaky ReLU] allows a small, non-zero output for negative inputs. It's defined as f(x)=max[]{.Times}([α]{.Times}x,x), where [α]{.Times} is a small constant (e.g., 0.01). This prevents neurons from dying and enhances learning.

[• Swish]is defined as f(x)=x[·]{.Arial-Unicode}[σ]{.Times}(x). It combines linear and non-linear properties, enabling smoother transitions and often outperforming ReLU in deep networks.

[• E-Swish and EIS Functions] are further refinements that introduce parameters to adjust the activation curve, providing flexibility to improve model performance in specific tasks.

[Neurons] are organized into layers, forming the architecture of a neural network. Each layer transforms the input data, extracting features and patterns that contribute to the final output. The most basic building blocks of the neural network are input layer, hidden layer, and output layer. In addition to this, we can have dropout layers, which can prevent overfitting, and normalization layers that

can mitigate issues such as internal covariate shift. Together, this can help improve the predictive power of the neural networks. Let's learn about these briefly.

[• Input Layer] receives the raw data. For example, in image recognition, each pixel's value might be an input neuron.

[• Hidden Layers] process inputs from the previous layer, performing computations that extract higher-level features. The term "deep learning" arises from networks with many hidden layers, known as deep neural networks.

[• Output Layer]is the final layer that produces the network's output, such as class probabilities in classification tasks or predicted values in regression tasks.

[• Dropout Layers] are regularization techniques that randomly "drops out" a fraction of neurons during training. By preventing neurons from co-adapting too much, dropout reduces overfitting and improves the network's ability to generalize to new data. For instance, setting a dropout rate of 0.5 means each neuron has a 50% chance of being ignored during a training step. This forces the network to learn redundant representations, making it more robust.

[• Batch Normalization (BatchNorm)] normalizes the inputs of each mini-batch to have a mean of zero and a variance of one. This stabilizes learning by reducing internal covariate shift, allowing for higher learning rates and faster convergence.

[• Layer Normalization (LayerNorm)] normalizes across the features within each data point. This is particularly useful in recurrent neural networks and transformer architectures where batch sizes may be small.

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.1.2-Forward-and-Backward-Propagation}[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161857}2.1.2 Forward and Backward Propagation

In the forward propagation process, input data moves layer by layer to produce an output. Imagine you're teaching a neural network to

recognize handwritten digits. Each image of a digit is transformed into numerical pixel values, which serve as inputs to the network.

The data first enters the [input layer] and then passes to the [hidden layer], where each neuron calculates a weighted sum of its inputs and applies an activation function to introduce non-linearity. This allows the network to learn complex patterns. This process repeats across all hidden layers until the data reaches the output layer, which generates the final prediction, identifying the digit in the image.

Mathematically, for each neuron, the output is computed as:

$$a = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Figure 13. Eq-1.jpg

Where:

• a is the output value

• x[i] are the input values

• w[i] are the weights

[• b] is the bias

[• f] is the activation function

[Backward Propagation]{.Bold}

[Backward propagation], or [backpropagation] adjusts the network's weights and biases based on the error of its predictions, effectively teaching the network to improve over time. Starting from the output layer, we propagate the error backwards through the network, layer by layer, adjusting weights along the way. This is where the term "backward propagation" originates.

After the forward pass produces an output, we compare it to the true label using a [loss function], which quantifies the discrepancy between the predicted and actual values. Common loss functions include [Mean Squared Error (MSE)] for regression tasks and [Cross-Entropy Loss] for classification.

The core idea of backpropagation is to calculate the gradient of the loss function with respect to each weight, denoted as ∂L/∂w. This gradient tells us how much a small change in weight would affect the loss. [Chain rule] from calculus is employed to compute these gradients efficiently.

[Challenges with Backpropagation]{.Bold}

While backpropagation is powerful, it faces significant challenges, particularly in deep networks with many layers. Two major problems are [vanishing gradient] and [exploding gradient] issues.

As gradients are propagated backward, they can diminish exponentially, becoming too small to make meaningful updates to the weights in earlier layers. This leads to [vanishing gradients]. This hampers the network's ability to learn long-range dependencies. Activation functions like the sigmoid and tanh are especially susceptible to this problem because their derivatives are less than one.

Conversely, gradients can grow exponentially, becoming so large that they cause numerical instability and prevent the network from converging. This is known as [exploding gradients]. This is more common when dealing with recurrent neural networks (RNNs) or very deep feedforward networks.

[Optimizers]{.Bold}

Optimizers adjust the network's weights during the training process to lower errors measured by the loss function. They do this by following the gradients calculated during backpropagation.

[Stochastic Gradient Descent (SGD)] updates the weights by looking at the gradient of the loss for one or a few examples at a time. This randomness (processing a small batch instead of the

whole dataset) can help the network explore different solutions. It is simple to understand, however, it can be slow to find the best solution and may get stuck in a suboptimal spot. We should use it when we are working with a very large dataset and want the model to generalize well. This is common in image recognition tasks with convolutional neural networks (CNNs).

[Adam] optimizer stands for [Adaptive Moment Estimation]. It builds on SGD by keeping a running average of past gradients and squared gradients. This means it adjusts the learning rate for each weight automatically, allowing it to take bigger or smaller steps as needed. It converges faster than SGD and works well when the data is complex or when the gradients (the changes needed) are sparse. It is useful in complex networks and when the data has lots of zeros or sparse information.

[RMSprop] also adapts the learning rate for each weight. It divides the learning rate by a running average of the squared gradients. This helps stabilize the learning process, especially when the loss function changes over time. It is effective when dealing with non-stationary problems, where the data patterns change over time, such as text or time series.

[Weight Initialization]{.Bold}

Weight initialization can help maintain stable gradients during training, allowing deeper networks to learn more effectively. Poor initialization can exacerbate vanishing or exploding gradients, hindering learning from the outset.

[• Xavier Initialization] is used for networks using sigmoid or tanh activation functions. It sets the initial weights based on the number of input and output neurons with a random distribution, ensuring that the variance of the outputs remains consistent across layers.

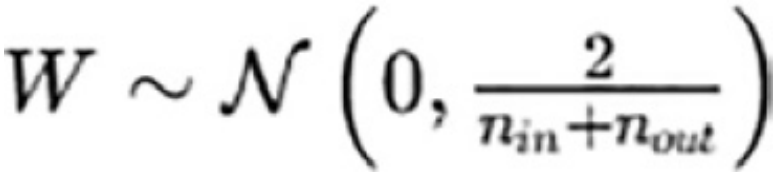


Figure 14. Eq-2.jpg

Where n[in] and n[out] are the number of input and output neurons, respectively.

[• He Initialization] is used for ReLU activation functions. It sets the weights to maintain variance, accounting for the fact that ReLU outputs zero for half of the inputs. It is initialized as below.

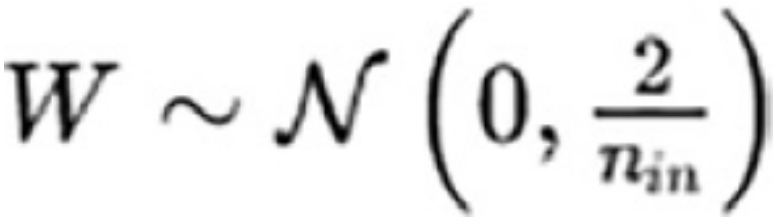


Figure 15. Eq-3.jpg

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.1.3-Regularization-Techniques}2.1[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161860}.3 Regularization Techniques

[Overfitting] happens when the neural network learns not only the useful patterns but also all the tiny, irrelevant details (or noise) in the training data. As a result, the model may do a great job on the training data but struggle with new examples because it has "memorized" the noise instead of understanding the true patterns. To mitigate this, [regularization] can prevent the model from getting too carried away with memorizing every detail. It does this by adding a "penalty" during the learning process, which

discourages the model from becoming overly complex.

[L1 and L2 Regularization]{.Bold}

[Regularization] can help mitigate overfitting. Two common regularization methods are [L1 Regularization (Lasso)] and [L2 Regularization (Ridge)]. Lasso sums the absolute values of all the model's weights to the loss function. Think of it as a small extra cost for having larger weights. This extra cost pushes many weights to become zero, resulting in model to only keep the most important features. Ridge adds the sum of the squares of the weights to the loss function, which discourages the weights from becoming too large but doesn't force them to zero. It keeps the weights smaller and more evenly spread out. It is useful for making the model less sensitive to slight changes in the training data, which helps when the data might have some noise or when features are very similar (a problem called multicollinearity).

[Dropout]{.Bold}

[Dropout] helps the network learn a more general set of features rather than memorizing the training examples. It does so by randomly turning off some neurons during the training process. For example, with a dropout rate of 0.5, each neuron has a 50% chance of being ignored during a training step. By randomly deactivating neurons, the network is forced to use many different neurons to solve the problem. This means it doesn't become too dependent on any single neuron, making it more robust and better at handling new, unseen data. When the model is being used to make predictions, dropout is not applied. However, to balance the fact that some neurons were missing during training, the weights are scaled accordingly. This ensures the model's predictions remain consistent.

[Data Augmentation]{.Bold}

[Data augmentation]can enable us to obtain more examples to train on without having to collect new data. Instead of just using the original set of images, you create new, slightly altered versions

of them. For example, we can rotate the original images a little bit, flip horizontally or vertically, adjust brightness or contrast, and add a little noise (small random changes). As a result, the model learns to recognize the digit even when it's slightly rotated, brighter, or has a little noise. This reduces overfitting and ensures that the model won't just memorize the training images but will be ready for real-world variations and thereby model doesn't become too attached to the specifics of the original images.

[Early Stopping]{.Bold}

[Early stopping] is a technique where you stop training as soon as the model stops improving during training epochs, on a separate set of data, called the validation set. If we train too long, the model might memorize the data and overfit. By stopping at the optimal point where the model performs best on unseen data, it is ensured that the model generalizes well.

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.2-Deep-Learning-Architectures}2.2[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161863} Deep Learning Architectures

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.2.1-Convolutional-Neural-Networks—CNNs-}2.2[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161866}.1 Convolutional Neural Networks (CNNs)

CNNs are designed to automatically learn and recognize patterns in images. They work much like our visual cortex, allowing computers to detect edges, textures, shapes, and even complex objects. It has 3 layers, known as convolutional layer, pooling layer, the fully connected layer.

[Convolutional Layer]{.Bold}

[Colvolutional layer] extracts features from the data and works like a sliding a small window (called a kernel or filter) over an image. At every position, the filter multiplies its numbers with the corresponding pixels and adds them up. This process creates a new image (called a feature map) that highlights specific patterns (like

edges or textures). By using multiple filters, the network can detect various features, each capturing different aspects of the image. [Pooling layers] reduce the size and emphasizes on key features.

[Pooling Layer]{.Bold}

[Pooling layers] reduce the size of the feature maps after the feature extraction is complete. For example, max pooling takes a small patch of the feature map and picks the highest value. This way, the network keeps only the most important information. Reducing the size helps decrease the computational load and makes the network less sensitive to small shifts or distortions in the image.

[Fully Connected Layer]{.Bold}

[Fully connected layer] interprets the features and outputs the final classification or prediction. For example, it decides what the image represents (e.g., cat, dog, or car).

[Advanced CNN Architectures]{.Bold}

[ResNet] is an advanced CNN design, which uses skip connections that let some layers bypass others. This helps the network learn better by preventing problems that occur when the network gets too deep, such as vanishing gradient problem. As a result, it allows training very deep networks, with dozens or even hundreds of layers, without losing important information.

[Inception networks]run several different-sized filters, such as 1x1, 3x3, and 5x5 in parallel on the same input. The results are then combined. This helps the network capture details at multiple scales, both fine details and broader features, without a huge increase in computational cost.

[MobileNet] is designed for devices with limited computing power, such as smartphones. It uses depthwise separable convolutions, which applies one filter per input channel to capture spatial details, and finally uses 1x1 filters to combine these details across channels. Doing this greatly reduces the number of computations and parameters, making the network lightweight and efficient.

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.2.2-Recurrent-Neural-Networks—RNNs—and-Transformers}2.2[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161869}.2 Recurrent Neural Networks (RNNs) and Transformers

Unlike regular neural networks that treat each input independently, RNNs have loops. These loops let them "remember" information from earlier in the sequence when processing the current input. Think of it like having a notebook where the network jots down important details from previous steps.

[Mathematical Foundations of RNNs]{.Bold}

For each time step [t], an RNN calculates a hidden state h[t] that captures the memory.

[h][= tanh(][hx][*][t][+][hh][* h][+ b][)]{.char-style-override-5}

[x][t] is the current input, an example can be the current word in a sentence)

[h][t−1] is the previous memory

[W][hx] and [W][hh] are weight matrices learned during training

[b][h] is a bias vector

[Challenges with Basic RNNs]{.Bold}

While this design is clever, basic RNNs face two major problems when dealing with long sequences. During training, the influence of early inputs can fade away, like whispers getting softer. This makes it hard for the network to learn long-term relationships in the data. This is known as [vanishing gradients]. Sometimes, the values can become too large, causing instability in the training process. Think of it as the network "shouting" too loudly, which also makes learning difficult. This is known as [exploding gradients].

[Advanced RNN Architectures: LSTMs and GRUs]{.Bold}

To overcome these challenges, researchers developed more advanced versions of RNNs that include special components called

[gates]. These gates help the network decide what information to keep or discard.

[Long Short-Term Memory (LSTM)] networks have a dedicated cell that holds the information over time. These are effective at remembering information for long periods because these gates carefully control the flow of information. There are 3 gates. [Forget gate] decides what old information to erase. [Input gate] and [candidate state] decide what new information to add. [Output gate] controls what part of the cell state to output.

[Gated Recurrent Units (GRU)][ ]simplify the LSTM design by combining some of the gates. The [update gate] determines how much of the past information to keep. The reset gate decides how much of the past information to forget. GRU calculates a candidate activation and then updates the hidden state. GRUs are generally faster and computationally more efficient while performing comparably to LSTMs.

[The Transformer Revolution]{.Bold}

[Transformers] represent a breakthrough in deep learning, fundamentally changing how computers handle [sequential data], especially in the realm of [language]. Unlike previous models that process words one at a time, these architectures process the entire sequence simultaneously. This innovative approach has dramatically improved efficiency and accuracy in tasks like translation and summarization.

At the core of this paradigm shift is the contrast between [traditional RNNs] (including [LSTMs] and [GRUs]) and [Transformers]. Older models process data [sequentially], which not only slows down training but also makes it difficult to capture relationships between words that are far apart in a sentence. In contrast, transformers leverage parallel processing, enabling them to learn [long-range dependencies] much more effectively by handling all parts of the sequence at once.

[Self-Attention and Parallel Processing]{.Bold}

One of the key innovations in [Transformers] is the [self-attention mechanism]. Instead of processing each word in isolation, the model examines the entire sequence and computes three sets of numbers: [queries], [keys], and [values] for every word. The [queries] and [keys] determine the importance of each word relative to others, while the [values] carry the essential information forward. This mechanism empowers the model to weigh the relevance of every word in context, irrespective of its position.

[Enhancements in Transformer Architecture]{.Bold}

Building on this, the concept of [multi-head attention] further enhances the architecture. By employing multiple attention heads, each with its own parameters, the model can capture various types of relationships, such as [syntax], [meaning], or even stylistic nuances from different representation subspaces. This multi-faceted approach enables a richer and more detailed understanding of the input data.

Another critical component is [positional encoding]. Since transformers do not inherently process data in order (unlike sequential models), [positional encoding] is introduced to inject information about the position of each token in the sequence. This technique acts like a set of bookmarks, ensuring that the model maintains awareness of the sequence order while still benefiting from parallel processing.

Additionally, [feed-forward networks] are applied in a position-wise manner after the attention mechanisms. These networks perform simple yet effective mathematical transformations, often using functions like ReLU to refine the information at each position. This step is crucial for integrating and transforming the learned features before the final output is generated.

[Advantages and Applications of Transformers]{.Bold}

The advantages of the [Transformer] architecture are multifold. With its [parallel processing] capability, it leverages modern GPUs for significantly faster training. The [self-attention] mechanism

enables the capture of long-range dependencies by directly linking any two positions in the sequence, regardless of their distance. Furthermore, the design is highly [scalable], which has paved the way for massive models like [BERT] and [GPT-3] that excel in a variety of tasks, including [NLP], text summarization, and even [computer vision] through adaptations like [Vision Transformers].

In practical applications, transformers have revolutionized the field of natural language processing (NLP) by setting new standards in machine translation, question answering, and text generation. Their impact is also evident in the development of powerful [language models] that generate human-like text. Moreover, the architecture's versatility extends to [computer vision], where [Vision Transformers] are achieving competitive results compared to traditional convolutional networks.

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.2.3-Hybrid-Architectures-and-Multimodal-Learning}2.2[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161872}.3 Hybrid Architectures and Multimodal Learning

The evolution of deep learning has paved the way for [hybrid architectures] that not only excel at individual tasks but also integrate multiple modalities of data. This integration, often referred to as [multimodal learning], combines different neural network models to process complex data types simultaneously. Such approaches are crucial for tasks that require an understanding of [spatial], [temporal], and even [semantic] relationships within data.

[Integration of CNNs and RNNs]{.Bold}

One prominent example of this integration is the combination of [CNNs] and [RNNs], which bridges the gap between [space] and [time]. Convolutional Neural Networks (CNNs) have revolutionized the interpretation of visual data by capturing spatial hierarchies in images. Meanwhile, Recurrent Neural Networks (RNNs) and in particular, its advanced forms like [LSTMs] and [GRUs] are designed to handle sequential data by retaining information over

time. In a hybrid [CNN-RNN] model, [CNN] layers first extract spatial features from each frame of a video or image, and these features are then fed into [RNN] layers that capture the temporal dynamics. This fusion enables the model to not only understand what is present in each frame but also how the content evolves over time, making it ideal for applications such as action recognition and image captioning.

[Multimodal Neural Networks]{.Bold}

[Multimodal neural networks] take this concept a step further by integrating data from various sources, such as images, text, and audio to form a more comprehensive understanding. For instance, in autonomous driving, a vehicle must process visual data from cameras, spatial data from LiDAR sensors, and auditory signals from microphones. By fusing these diverse inputs, a multimodal network can achieve a level of situational awareness akin to human perception. Similarly, in speech recognition and emotion detection, combining auditory cues with visual signals like lip movements helps enhance performance, especially in noisy environments.

[Advancements Beyond Traditional Models]{.Bold}

Beyond traditional models, architectures such as [Capsule Networks] and [Vision Transformers (ViT)] are addressing the limitations of conventional [CNNs]. While [CNNs] excel at detecting features, they can lose important spatial relationships due to pooling layers. [Capsule Networks] overcome this by encapsulating both the probability and pose information, such as orientation and spatial position of features, allowing for more robust object recognition under various transformations. In parallel, [Vision Transformers] adapt the Transformer model, originally developed for language processing in order to image recognition tasks by treating image patches as tokens. This approach enables the capture of global context across the entire image, regardless of where key features are located.

[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x2.2.4-Transfer-

Learning-and-Meta-Learning}2.2[]{#Neuro-Symbolic-AI-EPUB-5.xhtml#x.161875}.4 Transfer Learning and Meta-Learning

The evolution of deep learning has paved the way for powerful paradigms that enable neural networks to adapt rapidly to new tasks, even with minimal additional data. Among these, [transfer learning] and [meta-learning] have emerged as transformative approaches that not only reduce training times but also boost performance across various domains.

[Transfer Learning]{.Bold}

In [transfer learning], a neural network that has been pre-trained on a large dataset for one task is fine-tuned for a different, yet related, task. Imagine you've mastered playing the piano and now wish to learn the violin; your existing understanding of musical notation and rhythm provides a strong foundation for the new instrument. Similarly, a network like [ResNet], originally trained on the vast [ImageNet] dataset, has learned rich feature representations that can be adapted for specialized tasks such as medical image classification. Pre-trained models like [VGG], [ResNet], [BERT], and [GPT] have become cornerstones in this approach, particularly in fields like healthcare and natural language processing.

[Addressing Domain Adaptation]{.Bold}

One significant challenge in [transfer learning] is the issue of [domain adaptation], which arises when the source domain (the data the model was initially trained on) differs from the target domain (the new data). For example, a model trained on daytime images might struggle with nighttime images due to variations in lighting. Techniques such as [Domain Adversarial Neural Networks (DANN)] are designed to address this problem by learning feature representations that are consistent across different domains. Another innovative method is [style transfer], which transforms the style of the source domain to more closely match that of the target domain, enhancing the model's applicability in scenarios

like adapting simulated environments to real-world conditions in autonomous driving.

[Meta-Learning: Learning to Learn]{.Bold}

While [transfer learning] builds on existing knowledge, [meta-learning], often described as "learning to learn" takes the concept a step further by training models to rapidly adapt to entirely new tasks with minimal data. A notable approach in this area is [Model-Agnostic Meta-Learning (MAML)], which finds a set of model parameters that are especially sensitive to changes, allowing for quick fine-tuning with only a few examples. For instance, a speech recognition system leveraging [MAML] can quickly adjust to a new accent with just a small amount of user-specific speech data, thereby improving accuracy without extensive retraining.

[Implications and Applications]{.Bold}

The implications of these paradigms are profound. In [healthcare], for example, the combination of transfer learning and meta-learning enables the development of diagnostic models that can accurately detect diseases from a limited number of medical images, potentially saving lives through early detection. In [robotics], meta-learning empowers systems to acquire new skills, such as navigating unfamiliar terrain or manipulating novel objects, with minimal additional training. Moreover, in personalized user experiences like recommendation systems, these techniques allow AI to rapidly tailor its outputs to individual preferences, even when starting from limited data.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml}

Basic-Text-Frame []{#Neuro-Symbolic-AI-EPUB-6.xhtml#Chapter-3-}Cha[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161879}pter 3:

Foundations of Symbolic AI

Figure 16. 162620.png

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.1-Logic-and-Knowledge-Representation}3.1[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161882} Logic and Knowledge Representation

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.1.1-Propositional-and-First-Order-Logic}3.1[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161885}.1 Propositional and First-Order Logic

Logic serves as the cornerstone of symbolic artificial intelligence, offering a formal framework to represent and reason about knowledge. [Propositional logic] and [first-order logic] are fundamental systems that enable AI to model complex relationships, draw inferences, and make informed decisions. By exploring these logical systems, we gain valuable insights into how machines can mimic human reasoning and understanding.

In [propositional logic], we work with simple, declarative statements known as propositions, which can be either true or false but not both. These propositions are connected using logical connectives:

[• Conjunction (AND,][∧]

[• Disjunction (OR,][∨]

[• Negation (NOT, ¬)]

[• Implication (IMPLIES,][⟶]

[• Biconditional (IF AND ONLY IF,][↔️]

For example, consider the propositions:

[• A]: “The sky is clear.”

[• B]: “It will not rain.”

Using propositional logic, we can form complex statements like [A ∧ B]{.Arial-Unicode}, which means “The sky is clear, and it will not rain.”

The [syntax] refers to the formal structure and rules governing how symbols and formulas are constructed. It dictates that such formulas are constructed correctly according to the rules of the language.

The [semantics] deals with the meaning and interpretation of these symbols within a specific context or model. It assigns truth values to these formulas based on the truth values of their constituent propositions. If both [A] and [B] are true, then [A ∧ B]{.Arial-Unicode} is true; otherwise, it is false.

Moving beyond propositional logic, [first-order logic] (also known as predicate logic) introduces quantifiers and predicates, allowing us to express statements about objects and their properties or relations. This adds significant expressive power, enabling the representation of more complex scenarios.

Key components of first-order logic include:

[• Predicates]: Functions that return true or false, representing properties or relationships (e.g., [Loves(Alice, Bob)] means "Alice loves Bob").

[• Quantifiers]:

•[Universal Quantifier (][∀]: Indicates that a statement applies to all elements in a domain (e.g., [∀] means "P holds for all x").

•[Existential Quantifier (][∃]: Indicates that there exists at least one element for which the statement holds (e.g., [∃] means "There exists an x such that P holds").

An example of a first-order logic statement is:

[•][∀][→]

This reads as "For all x, if x is a bird, then x can fly." Here, [Bird(x)] and [CanFly(x)] are predicates applied to the variable [x].

The syntax of first-order logic involves rules for combining variables, quantifiers, predicates, and logical connectives into well-

formed formulas. The semantics involve interpreting these formulas over a domain of discourse, assigning meanings to the symbols, and determining the truth of statements based on that interpretation.

Logic is not just about stating facts; it's also about deriving new knowledge from existing information. This process is facilitated by [inference rules] and [proof techniques] that enable logical reasoning.

One fundamental inference rule is [Modus Ponens], which allows us to infer a conclusion from a conditional statement and its antecedent. Formally, Modus Ponens can be expressed as:

• Given:

• [P][→] (If P then Q)

[• P] (P is true)

• Therefore:

• [Q] (Q is true)

For instance, suppose we know that "If it rains, the streets will be wet" ([P][→]) and "It is raining" ([P]). Using Modus Ponens, we can conclude that "The streets will be wet" ([Q]).

Another essential inference technique is [Resolution], particularly used in automated theorem proving and logic programming. Resolution works by refuting the negation of the statement to be proved, using a process of unifying clauses to derive a contradiction. This method is powerful in propositional logic and can be extended to first-order logic with the use of unification algorithms.

These inference rules form the basis of logical proofs, where a sequence of justified steps leads from premises to a conclusion. They are instrumental in various AI applications, such as verifying the correctness of algorithms, reasoning in expert systems, and enabling machines to make deductions similar to human reasoning.

While propositional and first-order logic provide robust frameworks for knowledge representation and reasoning, they have inherent limitations that restrict their expressiveness in certain contexts.

[Limitations:]{.Bold}

[• Propositional Logic]: Lacks the ability to express general statements about groups of objects or to represent relationships between objects. It can only handle specific, concrete propositions without quantifiers or variables.

[• First-Order Logic]: Cannot quantify over predicates or functions, meaning it cannot make statements about properties themselves or express higher-level abstractions.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.1.2-Ontologies-and-Semantic-Networks}3.1[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161889}.2 Ontologies and Semantic Networks

Ontologies and semantic networks play a crucial role in providing structured frameworks that represent information and relationships within a domain. These tools form the backbone of the Semantic Web and are essential for advanced knowledge representation in AI.

An [ontology] is a formal, explicit specification of a shared conceptualization. In simpler terms, it's a structured framework that defines the types of entities within a domain and how they relate to each other. Ontologies are composed of several key components.

[1. Classes (Concepts):] These are the abstract groups or categories of objects within a domain. For example, in a medical ontology, classes might include [Patient], [Doctor], [Disease], and [Treatment]. Classes are often organized hierarchically, forming a taxonomy where subclasses inherit properties from their parent classes.

[2. Relations (Properties):] Relations define how classes and instances interact with each other. For instance, the relation [treats]

might connect a [Doctor] to a [Patient].

[3. Functions and Attributes:] [Functions] are special types of relations that provide outputs based on inputs. This is similar to mathematical functions. [Attributes] are properties that provide specific details about classes or instances, such as a patient's age or a disease's symptom list.

[4. Axioms:] [Axioms] are assertions or rules that are always true within the context of the ontology. They impose constraints and enable logical reasoning. For example, an axiom might state that "All patients must have at least one symptom," establishing a rule that can be used to validate data or infer new information.

These components together create a rich and interconnected representation of knowledge. This enables machines to perform complex tasks.

Building an ontology requires a combination of domain expertise and technical skills. One widely recognized methodology is [Ontology Development 101][[1]], which outlines the following steps:

[• Define the Domain and Scope][:] Clearly specify what the ontology will cover and its purpose by identifying key questions the ontology should answer.

[• Consider Reusing Existing Ontologies][:] Evaluate existing ontologies to determine if they can be extended or integrated, to save development time.

[• Enumerate Important Terms][:] List all relevant concepts, relations, and attributes that need to be represented.

[• Define Classes and Class Hierarchy][:] Organize the concepts into a hierarchical structure, establishing superclass and subclass relationships.

[• Define Properties and Constraints][:] Specify the relationships between classes and any constraints or rules (axioms) that apply.

[• Create Instances][:] Populate the ontology with actual data, providing examples that illustrate how the ontology functions in practice.

Throughout this process, it's essential to collaborate with domain experts to ensure the ontology accurately reflects the real-world concepts and relationships. Additionally, ontologies should be regularly reviewed and updated to accommodate new knowledge and changes in the domain.

While ontologies provide a formal structure, [semantic networks] offer a graphical representation of knowledge. In a semantic network, [nodes] represent concepts or entities, and [edges] represent the relationships between them. For example, a semantic network in the field of zoology might show how different species are related, their habitats, and their dietary habits.

The [Semantic Web] is an extension of the current web, where information is given well-defined meaning, enabling computers and people to work in cooperation. To achieve this, several key technologies are used:

[• Resource Description Framework (RDF)][:] RDF is a standard model for data interchange on the web. It structures data into triples consisting of a subject, predicate, and object. It enables data to be merged even if the underlying schemas differ.

[• Web Ontology Language (OWL)][:] Built upon RDF, OWL adds more vocabulary for describing properties and classes, including relationships between classes. This allows for more complex and nuanced ontologies, facilitating advanced reasoning.

[• SPARQL Protocol and RDF Query Language (SPARQL)][:] SPARQL is a query language for RDF datasets. It enables users to write queries that can retrieve and manipulate data stored in RDF format, similar to how SQL is used for relational databases.

Ontologies and semantic networks have widespread applications across various domains:

[• Healthcare][:] Ontologies like the Gene Ontology or SNOMED CT help standardize medical terminology, enabling interoperability between different healthcare systems and supporting advanced data analytics for patient care.

[• E-commerce][:] Online retailers use ontologies to categorize products, manage inventories, and enhance recommendation systems, improving the shopping experience by providing more accurate search results and personalized suggestions.

[• Knowledge Management][:] Organizations leverage ontologies to structure internal knowledge bases, making it easier to retrieve information, facilitate training, and support decision-making processes.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.1.3-Knowledge-Representation-Techniques}3.1[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161892}.3 Knowledge Representation Techniques

At a foundational level, knowledge can be represented symbolically with [frames and scripts], [production systems], [semantic networks], and [conceptual graphs]. Each offers a unique approach to modeling information.

[Frames and Scripts]{.Bold}

Imagine walking into a restaurant. You will expect to be seated, handed a menu, place an order, eat, and then pay the bill. This sequence of events is so familiar that it doesn't require conscious thought. In AI, [frames]and[scripts] are used to model such stereotypical situations and event sequences, capturing common experiences to facilitate understanding and reasoning.

[Frames] are data structures for representing a stereotyped situation, like a concept or an entity. For example, a "Restaurant" frame might include slots like "Menu," "Waiter," "Meal," and "Bill." Each slot provides specific information or actions associated with that concept.

[Scripts] describe a sequence of events or actions in a particular

context. They are akin to a play's script, detailing the actors, settings, and the order of events. A "Dining Out" script might outline the steps from entering the restaurant to leaving after the meal. Scripts help AI systems predict what happens next in a given situation, filling in missing information and resolving ambiguities.

[Production Systems]{.Bold}

[Production systems] are a rule-based architecture that uses condition-action pairs to represent knowledge and drive problem-solving. It excels in domains where expert knowledge can be codified into clear, actionable rules. It can become complex and difficult to manage as the number of rules grows. Maintaining consistency and handling exceptions require careful design. These systems consist of three main components:

[1. A Set of Production Rules][:] Each rule has a condition (if) and an action (then). When the condition is met, the corresponding action is executed. For instance, a medical diagnosis system might have a rule: "If the patient has a fever and cough, then consider the possibility of influenza."

[2. A Working Memory][:] This stores facts or assertions about the current state of the world. The production system matches the conditions in the production rules against this working memory to determine which actions to execute.

[3. An Inference Engine][:] This controls the application of production rules, deciding which rules to fire based on the current working memory and handling conflicts when multiple rules are applicable.

To represent knowledge in a way that highlights the relationships between concepts, [semantic networks] are employed. These are graphical structures consisting of nodes (representing concepts or entities) connected by edges (representing relationships). It provides a visual and intuitive way to represent and reason about knowledge

For example, consider a semantic network for animals. Nodes might include "Bird," "Fish," "Mammal," "Penguin," and "Canary." Edges would represent relationships like "is a" (hierarchical relationships), "can" (capabilities), or "has" (attributes). So, "Penguin" might be connected to "Bird" with an "is a" link and to "Cannot Fly" with a "has" link.

To combine the visual clarity of semantic networks with the formal rigor of predicate logic, [conceptual graphs] are introduced. It is composed of two types of nodes:

[1. Concept Nodes][:] Represent entities, objects, or ideas.

[2. Relation Nodes][:] Represent the relationships between concepts.

Edges connect these nodes to form a bipartite graph. For example, the sentence "The cat sits on the mat" can be represented with concepts for "Cat" and "Mat," connected by a relation "Sits On."

Conceptual graphs allow for complex expressions and can capture nuances such as quantifiers, negation, and modalities. They can be translated into predicate logic, enabling formal reasoning and proof techniques.

The knowledge representation techniques discussed play a pivotal role in various AI applications:

[• Expert Systems][:] Production systems enable the encoding of expert knowledge into actionable rules, supporting decision-making in fields like medicine and engineering.

[• Cognitive Modeling][:] Semantic networks and conceptual graphs model human memory and reasoning processes, contributing to the development of more human-like AI systems.

[• Education and Training][:] These techniques aid in creating intelligent tutoring systems that can adapt to learners' needs by understanding the knowledge domain structurally.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.2-Symbolic-Reasoning-and-Inference}3.2[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161895} Symbolic Reasoning and Inference

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.2.1-Rule-Based-Systems-and-Expert-Systems}3.2[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161898}.1 Rule-Based Systems and Expert Systems

Expert systems are designed to mimic the reasoning patterns of human experts. This architecture is typically composed of three fundamental components:

[1) Knowledge Base]: It is a repository and contains domain-specific information, represented through facts, rules, heuristics, and relationships pertinent to a particular field. For instance, in a medical expert system, the knowledge base would include symptoms, diagnoses, treatment protocols, and medical guidelines. It utilizes [Production Rules], [Semantic Networks], and [Frames]

[2) Inference Engine]: It is the reasoning mechanism of expert systems. It interprets and applies the knowledge from the knowledge base to solve problems and draw conclusions. It uses [forward chaining], which is a data-driven approach that starts with known facts and applies rules to infer conclusions, and [backward chaining], which is a goal-driven approach that starts with a hypothesis and works backward to find supporting evidence.

[Forward Chaining Example:] In an agricultural expert system, if we know that a plant's leaves are yellowing. The system uses forward chaining to apply rules:

Rule 1[:] IF leaves are yellowing AND soil is moist THEN possible nitrogen deficiency.

Rule 2[:] IF possible nitrogen deficiency THEN recommend nitrogen-rich fertilizer.

By sequentially applying these rules, the system concludes that a nitrogen-rich fertilizer should be recommended.

[Backward Chaining Example:] In the same agricultural system, if the goal is to determine whether to recommend a nitrogen-rich fertilizer, backward chaining would:

• Look for rules that conclude with recommending nitrogen-rich fertilizer.

• Identify the conditions needed for those rules (e.g., possible nitrogen deficiency).

• Check if the known facts (e.g., yellowing leaves, moist soil) satisfy these conditions.

[3) User Interface:]It is the bridge between the user and the expert system and allows users to input queries, and provide data. It returns explanations or recommendations.

Rule-based and expert systems have found applications across diverse fields:

[• Medicine]: Systems like MYCIN assist doctors in diagnosing infections and recommending treatments based on symptoms and lab results.

[• Finance]: Expert systems evaluate credit risks, detect fraudulent activities, and provide investment advice.

[• Manufacturing]: They monitor equipment, predict failures, and optimize production processes.

[• Customer Service]: Rule-based chatbots provide instant support by troubleshooting common issues.

[• Legal]: Systems interpret regulations and assist in contract analysis and compliance checks.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.2.2-Automated-Theorem-Proving}3.2[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161901}.2 Automated Theorem Proving

[Automated theorem proving] is the process by which computer programs prove or disprove logical statements, without human

intervention. It draws heavily on [symbolic logic], particularly first-order logic, to represent and manipulate propositions and predicates systematically. For example, Z3 SMT Solver[[2]], developed by Microsoft Research, is a high-performance theorem prover used extensively in software verification and analysis tools.

Two fundamental techniques underpin the functionality of automated theorem provers: [resolution] and [unification]. Together, resolution and unification enable automated reasoning systems to manipulate and simplify logical expressions.

[Resolution]{.Bold}

[Resolution] is used to deduce new clauses from existing ones. It is especially effective in propositional and first-order logic, forming the backbone of many theorem-proving algorithms. It can reduce complex logical formulas to simpler ones.For instance, consider the clauses:

[1. A] [∨]

2. ¬B [∨]

Resolution allows us to infer a new clause by eliminating [B]. Resolving clauses (1) and (2) on [B] yields [A][∨]

[Unification]{.Bold}

[Unification] finds a substitution of variables that makes the expressions equal. It finds the most general unifier (MGU), which is the simplest substitution that satisfies the equality of expressions.

For example, given the predicates:

• P(x,a)

• P(b,y)

Unification seeks substitutions for x and y such that P(x,a) = P(b,y). The MGU here is x = b and y = a, unifying the two predicates.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.2.3-Reasoning-Under-Uncertainty}3.2[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161904}.3

Reasoning Under Uncertainty

Traditional logic operates on absolutes. The statements are either true or false. However, many real-world scenarios involve ambiguity where truth values are not black and white. Such as our daily lives, where we make decisions without complete information, relying on probabilities, and instincts.

[Probabilistic logic] bridges the gap between classical logic and probability theory, allowing AI systems to reason with degrees of belief rather than certainties. It does so by assigning probabilities to logical statements, reflecting the likelihood that a statement is true.

For example, if we say there is a 70% chance it will rain tomorrow, the probability quantifies uncertainty. We can integrate these probabilities into logical reasoning. This will allow AI systems to make informed decisions, even when data is incomplete or ambiguous. This is helpful in fields such as medical diagnosis, where doctors encounter symptoms, which can indicate multiple conditions. Probabilistic logic can help us in assessing the likelihood of various diagnoses based on available symptoms.

[Dempster-Shafer Theory]{.Bold}

When precise probabilities are hard to come by, [Dempster-Shafer Theory] (DST) can be used for the representation of uncertainty, without the need to specify exact probabilities. It does so by using [belief functions] to quantify the confidence in a proposition. Belief functions provide sets of possibilities rather than individual outcomes.

For example, if we are using an AI system for diagnosing machine malfunction, where we lack sufficient data to determine the exact fault. DST can assign belief to a group of potential issues rather than a single cause.

DST can combine evidence from multiple sources through [Dempster's Rule of Combination], which updates the belief function as

new evidence becomes available, refining the system's confidence in various hypotheses.

For instance, if two sensors provide different but overlapping data about a system's state, DST can merge this information to form a more accurate assessment than either source alone. It is helpful in scenarios such as sensor fusion, which is used in autonomous vehicles, where data from LIDAR, camera, and radar improves environmental understanding. It can also help with identifying issues in complex systems with ambiguous or partial data.

In classical logic, once a conclusion is reached based on certain premises, adding more information cannot invalidate that conclusion. This is known as [monotonic reasoning]. However, in real-world situations, new information can contradict previous beliefs.

[Non-Monotonic Reasoning]{.Bold}

[Non-monotonic reasoning] allows the revision of conclusions when faced with new, potentially conflicting information. It models human common sense by permitting the withdrawal of inferences.

For example, initially, we might conclude that "Birds can fly". However, upon learning that "Penguins are birds that cannot fly," we retract the previous inference for penguins.

This is essential in domains where information is constantly evolving, such as legal reasoning, where [l]aws may have exceptions. Non-monotonic reasoning can help us interpret statutes that change with new precedents.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.3-Planning-and-Decision-Making}3.3[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161907} Planning and Decision Making

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.3.1-Search-Algorithms}3.3[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161911}.1 Search Algorithms

Search algorithms enable machines to navigate vast possibilities and make informed decisions. Three fundamental techniques are [Depth-First Search (DFS)], [Breadth-First Search (BFS)], and [Uniform Cost Search (UCS)].

[Depth-First Search (DFS)]{.Bold}

[Depth-First Search (DFS)] dives deep into a problem, exploring as far as possible along a branch before backtracking.

Imagine a maze where you always take the first unexplored path until you hit a dead end, then retrace your steps to the last decision point. This method is memory-efficient since it needs to store only a single path from the root to a leaf node.

However, it doesn't guarantee the shortest path to a solution and can get lost in infinite paths if not careful.

[Breadth-First Search (DFS)]{.Bold}

[Breadth-First Search (BFS)] takes a different approach, exploring all neighboring nodes at the current depth before moving to the next level.

Think of it as casting a wide net, ensuring that all possibilities at one level are examined before proceeding. BFS guarantees the finding the shortest path in an unweighted graph but can consume significant memory, as it needs to store all nodes at a given depth.

[Uniform Cost Search (UCS)]{.Bold}

[Uniform Cost Search (UCS)] extends BFS by considering the cost of actions. It always expands the node with the lowest cumulative cost, making it ideal for finding the least expensive path. In scenarios where actions have different costs, like navigating a city where roads have varying lengths, UCS ensures the most efficient route is found.

[Heuristic Search: A* and Greedy Best-First Search]{.Bold}

While classical search methods are systematic, they can be inefficient for large or complex problems. [Heuristic search algorithms

]on the other hand are more efficient, as they use domain-specific knowledge to guide the search process more intelligently.

A* algorithm, for example, combines the strengths of UCS with heuristics to estimate the cost from a node to the goal. The function [f(n) = g(n) + h(n)] guides A*, where [g(n)] is the actual cost from the start node to node n, and [h(n)] is the heuristic estimate of the cheapest path from [n] to the goal.

By selecting the node with the lowest [f(n)] value, A* improvises on the optimal path. For instance, when plotting a route on a GPS, A* can use the straight-line distance (as the crow flies) to estimate [h(n)], guiding the search toward the destination.

[Greedy Best-First Search] is another heuristic method that prioritizes nodes based solely on the heuristic [h(n)]. It can be faster than A* but doesn't guarantee the shortest path, as it may overlook the actual cost from the start node. It's akin to always heading toward the visible peak in a mountain range, which might lead you up a steep, impassable cliff.

[Local Search Algorithms]{.Bold}

When the search space becomes enormous, traditional search methods may falter. [Local search algorithms] offer a solution by focusing on iterative improvement, making them suitable for optimization problems. Algorithms such as hill climbing, simmulated annealing, and genetic algorithm can help us perform local search. Let's understand each of these algorithms.

1) [Hill Climbing] starts from an initial state, and moves to neighboring states that improve the objective function. It is comparable to climbing a hill in foggy weather, where we always step upward to reach the peak. This method can however get stuck in local maxima and miss the highest peak.

2) [Simulated Annealing] introduces randomness into the search. Inspired by the annealing process in metallurgy, it occasionally allows moves to the worse states to escape local maxima. The

probability of accepting worse states decreases over time, balancing exploration and exploitation. It's like wandering around the hill with decreasing randomness, hoping to find a higher peak.

3)[ ][Genetic Algorithms][(GAs)] take inspiration from biological evolution. It maintains a population of candidate solutions that evolve over generations. Operators like selection, crossover, and mutation guide the evolution, where [selection] chooses fitter individuals for reproduction, [crossover]combines parts of two parents to create offspring, and [mutation] introduces random changes to individuals. GA is useful when the search space is vast and complex. It is used in fields ranging from engineering design to machine learning hyperparameter tuning.

[Adversarial Search: The Minimax Algorithm]{.Bold}

In environments where decisions are influenced by an opponent, such as in games, [adversarial search algorithms] are essential. The [Minimax] algorithm models two-player, turn-based games as a tree, where nodes represent game states. One player aims to maximize the score, while the opponent aims to minimize it. By recursively exploring possible moves, Minimax identifies the optimal strategy assuming perfect play from both sides.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.3.2-Constraint-Satisfaction-Problems—CSPs-}3.3[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161914}.2 Constraint Satisfaction Problems (CSPs)

[Constraint Satisfaction Problems (CSPs)] enable systems to make decisions by exploring configurations that meet specified constraints. It is a structured way to represent complex problems and find solutions that satisfy all specified requirements.

The fundamental components of CSP are [variables],[domains],[ ]and[constraints]. [Variables] are the elements that need to be assigned values. [Domains] are the possible values that each variable can take. [Constraints] are the rules that define acceptable combinations of values for the variables. They represent the limitations or requirements that must be met.

[A Classic Example]{.Bold}

Consider the classic example of coloring a map with the least number of colors so that no neighboring regions share the same color. Here, the regions are variables, the colors are domains, and the rule that neighboring regions must have different colors is the constraint.

[Constraint Propagation]{.Bold}

One of the key strategies in solving CSPs is [constraint propagation]. It is a technique that reduces search space by eliminating values that cannot possibly be part of a valid solution.

[Arc Consistency] is a fundamental form of constraint propagation. It involves considering pairs of variables connected by a constraint (arcs) and ensuring that for every value in the domain of one variable, there is a consistent value in the domain of the connected variable. If not, inconsistent values are pruned from the domains.

[Path Consistency] extends this concept by considering triples of variables and ensuring consistency across paths in the constraint network. It further examines the relationships between variables over longer chains, leading to more effective pruning.

[Backtracking and Forward Checking]{.Bold}

In order to thoroughly explore the remaining options after narrowing down possibilities, we need to use [backtracking] and [forward checking].

[Backtracking and Forward Checking]{.Bold}

[Backtracking] is a depth-first search algorithm that incrementally builds candidates for the solutions and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution. It can be inefficient if the search space is large. In such cases, [forward checking] is often used in conjunction with backtracking. [Forward Checking] works by keeping track of remaining legal values for unassigned

variables. After assigning a value to a variable, it eliminates inconsistent values from the domains of neighboring unassigned variables. If any domain becomes empty, the algorithm backtracks immediately, avoiding futile explorations. This helps us avoid the number of dead ends encountered during the search, significantly enhancing performance.

[Real-World Applications]{.Bold}

Constraint Satisfaction Problems are instrumental in solving practical problems across various domains, such as [scheduling and planning] in industries like aviation, manufacturing, and education. For example, airlines must schedule flights considering crew availability, aircraft maintenance, and airport regulations. Schools create timetables that accommodate teacher schedules, room availability, and course requirements. In robotics, CSPs is used for planning paths that avoid obstacles, comply with kinematic constraints, and optimize travel time.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.3.3-Automated-Planning}3.3[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161917}.3 Automated Planning

Automated planning enables systems to sequence actions, achieve goals, and adapt to dynamic environments, without any active human intervention. From autonomous robots navigating complex terrains to sophisticated software optimizing logistical operations, automated planning algorithms form the invisible framework that powers these technologies.

Automated planning began with classical algorithms that laid the groundwork for modern AI applications. Stanford Research Institute Problem Solver (STRIPS)[[3]] is a pioneering framework that revolutionized how machines approach problem-solving. It models planning problems using a set of actions defined by their preconditions and effects. It represents the world in terms of states and transitions. In the next step, STRIPS searches for a sequence of actions that transforms an initial state into a desired goal state.

[Partial-Order Planning]{.Bold}

Building upon the foundations of STRIPS, [Partial-Order Planning (POP)] introduced a new dimension of flexibility. POP allows for actions to be partially ordered. This means that some actions can occur independently or in parallel, as long as their necessary conditions are met.

By employing a least-commitment strategy, POP defers decisions about the exact ordering of actions until more information is available or until it becomes necessary. This reduces unnecessary constraints and leads to more efficient and adaptable plans, especially in complex or multi-agent environments.

[Hierarchical Task Networks]{.Bold}

[Hierarchical Task Networks (HTNs)] are suitable for complex planning, as it breaks down large tasks into smaller, more manageable subtasks. In HTN planning, tasks are decomposed recursively using methods that specify how a task can be accomplished through a combination of subtasks. It allows planners to incorporate domain-specific knowledge at various levels of abstraction.

Imagine organizing a music festival, where we have components such as securing a venue, booking artists, arranging accommodations, marketing, and ensuring security. HTNs can tackle this by decomposing the main task into these subtasks, which can be further broken down. For example, "securing a venue" might involve negotiating contracts, arranging permits, and coordinating logistics.

[Temporal and Probabilistic Planning]{.Bold}

In real-world environments, time constraints and uncertainty are inherent. There are planning methods that have evolved to incorporate such temporal and probabilistic considerations. [Temporal planning] introduces the concept of time into the planning process, where actions have durations and temporal relationships with other actions.

This allows planners to schedule actions more effectively, ensuring that time-dependent constraints are met. For instance, in project management, certain tasks cannot begin until others are completed, and resources may only be available during specific time windows. Temporal planning algorithms can model these constraints, optimizing the schedule to meet deadlines and resource limitations.

[Probabilistic planning] considers the likelihood of different outcomes and the stochastic nature of the environment. Actions may have uncertain effects, or external events may influence the state of the world in unpredictable ways. Probabilistic planning methods, such as Probabilistic Simple Temporal Networks (pSTNs), model these uncertainties to generate plans that are robust against variability.

For example, in robotics, a robot navigating a cluttered environment must account for the possibility of obstacles appearing unexpectedly or sensors providing noisy data.

[Planning Domain Definition Language (PDDL)]{.Bold}

The [Planning Domain Definition Language (PDDL)] is one of the most widely used languages in the field of automated planning. It provides a formal syntax and semantics for defining the elements of a planning problem, including actions, states, goals, and constraints. This in turn facilitates collaboration and benchmarking within the research community. Despite its advantages, modeling complex domains through PDDL requires a deep understanding of both the problem and the language's intricacies. Writing efficient and accurate PDDL representations can be time-consuming and error-prone.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.3.4-Decision-Theory-and-Rational-Agents}3.3[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161921}.4 Decision Theory and Rational Agents

Utility, as a concept, is used as a measure of satisfaction an agent assigns to different outcomes. As per this theory, rational agents

make choices to maximize their expected utility by evaluating possible actions by considering the utility of their potential outcomes and the probabilities of those outcomes occurring.

For instance, consider an autonomous vehicle navigating through traffic and trying to decide whether to overtake a slow-moving truck or stay behind it. It assesses the potential utilities of overtaking, which could save time, but may involve risk if the oncoming lane isn't clear. Staying behind is safer, but slower. By calculating the expected utility of both actions, the vehicle chooses the one that maximizes its expected benefit.

Real-world decision-making often deviates from pure expected utility maximization. Humans, for example, exhibit behaviors like risk aversion and preference biases. This paradox shows that people's choices can violate expected utility principles due to inconsistent risk preferences. Such insights have led to alternative models like prospect theory, which accounts for how people actually perceive gains and losses.

[Markov Decision Processes (MDPs)]{.Bold}

Markov Decision Processes (MDPs) offer a robust mathematical framework that can handle uncertainty and sequential decisions. It is helpful for decision-making in situations where outcomes are partly random and partly under the control of the decision-maker. An MDP consists of [states], [actions], [transition probabilities], and [rewards].

[States] have all possible situations the agent can be in. [Actions] are the choices available to the agent in each state. [Transition Probabilities] are the likelihood of moving from one state to another, given an action. [Rewards] are the immediate returns received after transitioning between states.

In a delivery drone example, each delivery location represents a state, and actions include moving north, south, east, or west. Transition probabilities account for factors like wind or obstacles that may affect movement. Rewards are assigned based on successful

deliveries or energy consumption. By utilizing MDPs, the drone can plan optimal routes that maximize efficiency and reliability.

MDPs however assume that the agent is aware of all possible states and actions. This is not always true in real-world scenarios. Extensions like MDPs with unawareness (MDPUs) address this by allowing agents to operate even when they lack complete knowledge. Additionally, Risk-Sensitive MDPs introduce the agent's risk preferences into the decision-making process, enabling more tailored strategies in uncertain environments.

[Game Theory in Multi-Agent Systems]{.Bold}

[Game theory] is helpful in multiple agent environments, where each one's decisions can significantly impact the others. It studies the interaction between agents, providing insights into how rational agents behave in competitive or cooperative settings. It helps agents anticipate and adapt to the actions of others. This is helpful where strategy is the key, in fields such as automated trading, network security etc.

Consider two autonomous cars approaching a narrow bridge from opposite sides. Both must decide whether to proceed or yield. Game theory models this situation, predicting outcomes based on the strategies each agent employs. [Nash equilibrium], states that no agent can benefit by unilaterally changing their strategy if the others keep theirs unchanged.

In our example, the optimal outcome is for one car to yield while the other proceeds, avoiding a stalemate or collision.

Alternative approaches such as [virtual bargaining] suggest that agents implicitly negotiate and agree on strategies that lead to mutually beneficial outcomes, even without explicit communication.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.4-Knowledge-Acquisition-and-Engineering}3.4[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161924} Knowledge Acquisition and Engineering

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.4.1-Knowledge-Elicitation-Techniques}3.4[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161927}.1 Knowledge Elicitation Techniques

For developing intelligent systems, deep knowledge of the subject matter is foundational. Whether it's a doctor diagnosing a complex condition, an engineer troubleshooting a critical system failure, or a pilot navigating through unexpected weather conditions. The challenge, however, lies in capturing this knowledge and converting it into a form that artificial intelligence (AI) systems can utilize. Knowledge elicitation techniques are helpful in achieving this.

[Direct interaction] with domain experts is one of the most effective ways to gather specialized knowledge. Interviews and surveys are the most commonly used methods for direct interaction.

[Interviews] offer a qualitative approach, where we engage with experts in conversation, we explore their thought processes, decision-making strategies, and problem-solving techniques in depth. For example, we can interview experienced lawyers to create an AI system for assisting in legal case analysis. We can probe how they interpret laws, weigh precedents, and build arguments through open-ended questions. Interviews can be unstructured, where conversations flow naturally without a strict agenda, allowing experts to guide the discussion to areas they find most significant. It can be structured, when we have a predefined set of questions to ensure consistency, in order to make it easier for comparing responses from multiple experts. It can be Semi-Structured, where it balances structure with flexibility, using guided questions while allowing room for spontaneous insights. Unstructured interviews can uncover unexpected insights, while structured interviews are useful for data analysis. Semi-structured interviews offer the best of both worlds, providing a framework while remaining adaptable to the expert's responses.

[Surveys] can help us reach a wider audience, where we can collect quantitative data and opinions from many experts simultaneously.

This makes it useful for validating findings on a larger scale, where we identify patterns among different experts.

For instance, in developing a new software tool for data scientists, a survey could collect input on preferred features, common pain points, and desired improvements. This information can guide the development process to align with the actual needs of users. A survey can include multiple-choice questions for quantitative analysis or open-ended questions for qualitative insights. However, surveys may not capture the depth of knowledge that interviews can provide. They also rely on the willingness of experts to participate and the clarity of the questions posed.

[Protocol analysis] can help us gain a deeper understanding of how experts think and make decisions, as it offers a window into their cognitive processes. In this method, experts verbalize their thoughts while performing tasks, we can observe the reasoning behind their actions in real-time.

[Think-aloud protocols]require experts to articulate their thought process as they engage in a task. It captures the sequence of thoughts, decisions, and problem-solving strategies directly as they occur. It can help us find Implicit knowledge that experts may not mention in interviews or surveys, decision points and criteria used during problem-solving, strategies and heuristics that guide expert performance.

This is especially useful in domains where tasks are complex and cognitive processes are not easily observable. For example, a seasoned detective solving a case might verbalize: "I'm looking at the crime scene photos... The position of the objects suggests a struggle... This could indicate the victim knew the assailant..." This real-time narration provides valuable insights into their analytical approach.

[Retrospective protocols], require experts to describe their thought processes after completing a task. They reflect on what they did, why they did it, and how they arrived at certain decisions.

For instance, a chess grandmaster might review a game they just played, explaining the reasoning behind each move, the strategies they employed, and how they adapted to their opponent's tactics. This reflection can reveal higher-level thinking and planning.

[Machine learning] can be leveraged to extract knowledge directly from data sources, using algorithms. This can help us identify patterns, relationships, and rules within large datasets.[ ]Techniques such as decision trees, neural networks, and clustering algorithms enable systems to learn from examples and improve over time. Such solutions are scalable, due to it's ability to process and analyze large datasets. Results from such methods are consistent, as algorithms apply the same criteria across all data, reducing human error. Also, the models can be updated with new data to reflect changing patterns or information.

For instance, in legal tech, NLP can process vast amounts of legal documents to extract clauses, obligations, and case precedents. This automated extraction can accelerate legal research and help lawyers identify relevant information quickly. The most effective knowledge acquisition often combines expert input with automated methods.

Experts can guide machine learning models in feature selection by identifying which data attributes are most relevant. Experts can also help in assessing the outputs of models for accuracy and relevance. This can result in more robust and reliable AI systems.

However, machine learning models are only as good as the data they are trained on. Poor-quality data can lead to inaccurate models, and algorithms may not capture the nuances of expert reasoning without proper guidance.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.4.2-Knowledge-Representation-Languages}3.4[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161930}.2 Knowledge Representation Languages

the way knowledge is represented significantly influences how effectively machines can reason and learn. Knowledge represen-

tation languages are the tools that allow AI systems to process complex information, making them indispensable in symbolic AI.

[Logic Programming Languages]{.Bold}

[Logic programming languages] provide a framework where for using logic to express computations. It can model complex relationships and support deductive reasoning.

[Prolog], short for "Programming in Logic" revolutionized the way developers approached problem-solving by allowing them to declare [what] needs to be done rather than detailing [how] to do it. It's declarative nature makes it well-suited for tasks involving symbolic reasoning and manipulation.

It can parse and interpret human language by defining grammatical rules and relationships. It can also handle recursive queries and pattern matching, making it ideal for developing expert systems.

[A-Prolog] and [Answer Set Programming]languages extend the capabilities of traditional logic programming. It does so by incorporating stable model semantics. This allows for more expressive representations of knowledge.

[Description Logics]{.Bold}

[Description Logics (DLs)] is a family of formalisms, designed to strike a balance between expressivity and computational tractability. They serve as the underpinning for ontology languages like the Web Ontology Language (OWL), which is foundational to the Semantic Web.

It enables the definition of concepts, roles, and individuals, allowing AI systems to reason about classes of objects and the relationships between them through its ability of automated reasoning.

Reasoners can infer implicit knowledge, from the explicitly defined facts, such as subclass relationships or property characteristics. This is crucial for applications like semantic search engines, where understanding the meaning behind the data leads to more accurate and relevant results.

[Fuzzy Description Logics] are an improvement on DLs. It can allow for reasoning with degrees of truth, accommodating the inherent uncertainty present in many real-world domains.

[Frame Languages]{.Bold}

[Frame Languages] offered a way to represent stereotypical situations, before the adoption of logic-based representations. [KL-ONE], is an example of such a system, which introduced structured knowledge representation through frames.

Frames are data structures for dividing knowledge into substructures by representing "types" of objects and the relationships between them.

KL-ONE allows for the creation of complex concepts through a hierarchy, supporting inheritance and facilitating reasoning about categories and their instances. This is useful in natural language understanding, where the meaning of a sentence can depend heavily on context and relationships between entities.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.4.3-Validation-and-Verification-of-Knowledge-Bases}3.4[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161933}.3 Validation and Verification of Knowledge Bases

Knowledge bases empower machines to emulate human reasoning. They store vast amounts of information that AI systems use to make decisions, and solve problems. Ensuring knowledge bases are accurate, consistent, and up-to-date is crucial for the reliability and effectiveness of AI applications, especially in the realm of neuro-symbolic AI.

[Consistency Checking]{.Bold}

[Consistency checking] is the process of examining a knowledge base to identify and resolve any conflicting information. It is not just about identifying errors, it's also about maintaining the integrity of the knowledge base over time.

Regular validation ensures that new information integrates seamlessly without disrupting existing, reliable data. This is crucial, as inconsistent data can lead to flawed reasoning, erroneous conclusions, and unreliable AI behavior.

Logical frameworks model the relationships within the knowledge base, enabling the use of formal algorithms to detect inconsistencies. There are advanced probabilistic approaches, where inconsistencies are measured, and probabilities are adjusted to restore harmony within the knowledge base.

[Completeness and Soundness]{.Bold}

[Completeness and soundness] are essential for any knowledge base.

Completeness refers to the extent to which the knowledge base contains necessary information to make accurate inferences. Soundness ensures that all the inferences and conclusions drawn from the knowledge base are valid and logically derived from the available data. Formal logic and rigorous methodologies are used to ensure that the knowledge base is thorough and that the inference mechanisms operate correctly.

In practice, absolute completeness and soundness is challenging to achieve. This is due to the complexity and constant evolution of real-world information. Therefore, aiming for a practical level of thoroughness and accuracy, while focusing on critical areas where precision is important.

Consider an AI legal advisor designed to assist in contract reviews. If the knowledge base is incomplete, missing critical laws or precedents. In such a case, the AI might overlook important legal considerations, leading to flawed advice. Conversely, if the system's reasoning processes are unsound, it might draw incorrect conclusions even with complete information, potentially causing legal missteps.

[Maintenance and Updating]{.Bold}

[Maintenance and updating] are essential processes for a knowledge base. It ensures that the knowledge base reflects the most current information, adapts to new findings, and continues to meet the needs of its users.

Maintenance also involves regular audits to identify and correct errors, redundancies, or outdated information. There are both automated and human-in-the-loop processes for updating knowledge bases.

Adaptive knowledge bases combine different reasoning approaches to refine themselves over time. They do so by leveraging historical data and real-time inputs and adjusting their models to improve accuracy and performance continually. This ability is crucial in fields like predictive maintenance, where AI systems must anticipate equipment failures based on the latest operational data.

Imagine an AI system that provides investment advice based on market trends. Financial markets are dynamic, with new data emerging constantly. If the knowledge base isn't updated regularly, the AI's recommendations could quickly become outdated, leading to poor investment decisions.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.5-Philosophical-Foundations-and-Cognitive-Modeling}3.5[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161936} Philosophical Foundations and Cognitive Modeling

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.5.1-Symbolic-AI-and-Cognitive-Science}3.5[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161939}.1 Symbolic AI and Cognitive Science

Symbolic AI seeks to replicate the way humans think, reason, and solve problems by bridging artificial intelligence and cognitive science.

[Physical Symbol Systems Hypothesis (PSSH)] is a foundational concept in AI and cognitive science. It proposes that human cognition operates through the manipulation of symbols. Much like

a computer processes data. It suggests that symbols are the building blocks of knowledge and reasoning. Human mind functions by processing these symbols according to formal rules, enabling complex thought processes such as language understanding, problem-solving, and abstract reasoning. Any system capable of intelligent action must be a physical symbol system, which can create, modify, and relate symbols to produce thought and behavior. It aligns closely with classical AI approaches, where programs are designed to process symbolic representations of the world.

For example, consider how humans solve mathematical equations. We use symbols and rules to manipulate those symbols to arrive at a solution. Symbolic AI can emulate this process by encoding knowledge into symbols and using algorithms to manipulate them.

However, PSSH has faced challenges and critiques. As human cognition cannot be fully captured by symbol manipulation alone. The human thought includes aspects like emotions, intuition, and embodied experiences that may not be easily represented symbolically. This has led to alternative theories and the integration of sub-symbolic approaches, such as neural networks, which model cognition through patterns and connections rather than explicit symbols.

[Modeling mental representations] allows AI systems to handle complex tasks that require understanding context, anticipating consequences, and adapting to new information. It enables the simulation of emotional responses, such as surprise or disappointment, by defining how changes in beliefs and desires affect intentions. Symbolic AI can encapsulate concepts like beliefs, desires, and intentions. By representing these mental constructs symbolically, AI systems can simulate decision-making processes that mirror human reasoning. Belief-Desire-Intention (BDI) model is a prominent framework, which provides a structure for artificial agents to make decisions based on their beliefs about the world, their desires or goals, and their intentions to act. It can help AI systems to plan and execute actions in a way that appears purposeful and rational.

For instance, imagine an AI personal assistant designed to manage a user's schedule. The assistant holds beliefs about the user's appointments (beliefs), aims to optimize the user's time (desires), and schedules meetings accordingly (intentions). By continuously updating its beliefs and adjusting its intentions, the assistant can adapt to changes and make decisions that align with the user's preferences.

Replicating human mental states however, is a formidable challenge. Humans often employ heuristics or simple rules of thumb rather than exhaustive logical reasoning. Factors such as emotions, cultural influences, and subconscious biases play significant roles in decision-making. Symbolic models strive to capture these nuances, but they must balance complexity with computational feasibility.

[Cognitive architectures][ ]are comprehensive frameworks, which aim to model the underlying structures and processes of the human mind, providing platforms for simulating a wide range of cognitive activities. These were developed to bridge the gap between human cognition and artificial intelligence. Two of the most common architectures are ACT-R (Adaptive Control of Thought-Rational) and Soar.

[ACT-R] models human cognition through a set of modules corresponding to different cognitive functions, such as memory, perception, and motor actions. It operates on the principle that human thought arises from the interaction of these modules, governed by production rules that determine behavior. It has been applied in various fields, from psychology to human-computer interaction, demonstrating its versatility in modeling cognitive tasks.

[Soar] utilizes a symbolic representation of knowledge and employs mechanisms like chunking to learn from experience. It focuses on general intelligence and problem-solving in dynamic environments. It is designed to be a unified theory of cognition, capable of simulating any aspect of human intellectual ability. It can

simulate complex processes, such as strategic planning in military simulations or troubleshooting in technical support systems.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.5.2-The-Frame-Problem}3.5[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161943}.2 The Frame Problem

The frame problem is about how an AI system can efficiently determine what changes and what remains the same after an action occurs.

In everyday life, humans effortlessly infer that most aspects of the world stay constant unless acted upon. For example, when you move a chair from the kitchen to the living room, you inherently know that the color of the chair hasn't changed, nor has the position of the sofa in the living room. Unless, of course, something else moved it.

Translating this intuitive understanding into a computational model is challenging. AI systems need a way to represent the direct effects of actions and the vast array of things that remain unaffected. Enumerating all the unchanged aspects is impractical due to the sheer amount of information.

There are solutions to address the frame problem. Two of the most influential solutions are [Situation Calculus] and, the [Event Calculus].

[Situation Calculus]{.Bold}

[Situation Calculus] is a mathematical logic language, which models the world as a series of situations resulting from sequences of actions. Each action transforms one situation into another, capturing the cause-and-effect relationship.

Successor state axioms[ ]specify the conditions under which a particular property holds in the next situation. It assumes that everything else remains unchanged unless affected by an action.

It provides a robust foundation for modeling hypothetical scenarios

and counterfactual reasoning. It can however, become unwieldy when dealing with nondeterministic actions or concurrent events.

For example, if we have a fluent [DoorOpen] representing whether a door is open, a successor state axiom might state that [DoorOpen] will be true after performing the action [OpenDoor]. It will remain true unless an action [CloseDoor] is performed. This reduces the complexity of reasoning about states by concentrating on relevant changes.

[Event Calculus]{.Bold}

[Event Calculus][ ]offers an alternative by focusing on events and their effects over time. It represents the world's history as a timeline of events.

It uses predicates to describe when properties hold. It is suited for reasoning about actual events that have occurred and their consequences. It overcomes some limitations of the situation calculus, especially in applications requiring detailed temporal reasoning, such as scheduling.

For instance, consider modeling the temperature of a room. An event like [TurnOnHeater] would initiate a period where the temperature increases until another event [TurnOffHeater] occurs. The event calculus can represent this continuous change over time, which is more challenging in the situation calculus.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x3.5.3-Symbol-Grounding-Problem}3.5[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161946}.3 Symbol Grounding Problem

[Symbol Grounding Problem] questions how symbols and abstract representations manipulated by AI systems acquire meaningful connections to the real world.

Consider a language translation AI that processes the word "apple". It might simply be a string of characters for the AI, as it can't taste the sweetness of the fruit, see its vibrant colors, or feel its texture.

AI systems need mechanisms to [ground symbols], in order to associate them with real-world referents through perception and interaction.

[Perceptual Grounding]{.Bold}

One approach to achieve this is by linking symbols to inputs from cameras, microphones, and other sensors. Thereby allowing the AI to form associations between symbols and the physical world.

For example, a robot that both sees and feels an object can associate the symbol "soft" with tactile sensations and visual cues, developing a grounded understanding of the concept.

[Social Interaction and Convention]{.Bold}

Another way to approach this is through social interaction and convention. Humans learn the meanings of symbols through language and context.

AI systems that engage in dialogues with humans can acquire symbol meanings by interpreting usage patterns, context, and feedback. Much like a child learns language through conversation.

[Embodied Cognition]{.Bold}

[Embodied cognition] offers a transformative lens and posits that cognition isn't confined to abstract computations in the brain but is deeply rooted in the body's interactions with the physical world. Our thoughts, and understanding emerge from the interplay of neural processes, bodily experiences, and environmental contexts.

Take the example of learning to ride a bicycle. This isn't learnt solely through reading instructions or observing others. It requires physical practice and adjusting in response to real-world feedback.

The knowledge of how to ride becomes embodied, integrating sensory inputs, motor actions, and cognitive adjustments.

Robots or virtual entities with sensors and actuators can interact with their environment and learn the concept of "obstacle", not just

as a data point but as something it must physically detect and avoid. This becomes meaningful through robot's embodied encounters.

By situating AI systems within rich, interactive environments, we enable them to develop more nuanced and context-sensitive understandings of symbols.

[Environmental Context and Sensorimotor Contingencies]{.Bold}

Embodied cognition also underscores the importance of the environment in shaping cognition. The surroundings provide stimuli and challenges that influence how symbols are interpreted. By situating AI systems within rich, interactive environments, we enable them to develop more nuanced and context-sensitive understandings of symbols.

[Sensorimotor contingencies] endows the understanding of how actions affect perceptions. AI that knows turning its camera changes its visual input can ground spatial symbols like "left" and "right" through experience, enhancing its navigational abilities.

[Impact on Developing Robust AI Systems]{.Bold}

Grounding symbols has far-reaching implications for the advancement of AI:

1. Grounded AI systems can learn more effectively from their environments, and can adapt to new situations and generalize knowledge across contexts. This is useful for tasks like autonomous driving, where AI must interpret and react to a complex, ever-changing world.

2. AI systems can communicate more naturally with humans, as they can disambiguate words with multiple meanings based on context, leading to more accurate translations and responses.

3. It allows AI to reason about the physical world with greater accuracy. An AI that understands "fragile" is not just a label but also a property affecting how objects should be handled, can make better decisions in tasks involving manipulation or safety considerations.

4. Grounded AI can better comprehend human values and social norms by experiencing and interpreting them within context. This is helpful for AI systems deployed in settings where empathy and ethical considerations are paramount.

[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#Citations}Cit[]{#Neuro-Symbolic-AI-EPUB-6.xhtml#x.161949}ations

[1] Noy, Natasha. "Ontology Development 101: A Guide to Creating Your First Ontology." (2001)

[2] de Moura, L., Bjørner, N. (2008). Z3: An Efficient SMT Solver. In: Ramakrishnan, C.R., Rehof, J. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2008. Lecture Notes in Computer Science, vol 4963. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-78800-3_24

[3] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence, 2(3-4):189—208, 1971

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml}

Basic-Text-Frame []{#Neuro-Symbolic-AI-EPUB-7.xhtml#Chapter-4-}Cha[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161953}pter 4:

Neuro-Symbolic Integration Techniques

Figure 17. 162658.png

[Symbolic] approaches excel at handling rule-based knowledge and provide semantic interpretability. It also helps with performing logical inference. [Neural networks] can process unstructured raw data, such as images, text, and audio to identify complex patterns without explicitly stated rules.

[Neuro-Symbolic Integration Techniques] aim to blend the strengths of neural models and symbolic systems. Integration

can be done with simple coupling mechanisms to elaborate hybrid systems. These integration techniques are like "recipes" or "strategies" to blend the two different ways of thinking into a more powerful whole.

Think of it as making the "neural" part and the "symbolic" part work together as a team. For example, building an AI that can both understand patterns (like recognizing cats in photos) and apply reasoning (like understanding that "if it's a cat, and cats are mammals, then it's a mammal").

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.1-Embedding-Symbolic-Knowledge-into-Neural-Networks}4.1[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161956} Embedding Symbolic Knowledge into Neural Networks

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.1.1-Techniques-for-Embedding-Symbolic-Knowledge}4.1[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161959}.1 Techniques for Embedding Symbolic Knowledge

Embedding symbolic knowledge into neural architectures empowers AI systems to leverage structured, human-understandable knowledge while benefiting from the pattern recognition capabilities of deep learning.

One approach to do this is by using [tensor embeddings]. Tensors are multi-dimensional arrays that provide a mathematical framework for representing the complex relationships in symbolic data. Entities and their interrelations are mapped into continuous vector spaces.

Tensor embeddings excel at handling heterogeneous information from knowledge graphs, which are rich repositories of symbolic knowledge. They capture multiple types of semantic information simultaneously, enhancing tasks like knowledge graph completion and link prediction. For instance, in natural language processing, tensor embeddings help models comprehend the nuances of lan-

guage by preserving the relationships between words, phrases, and their meanings.

Consider a multi-disease diagnosis system in healthcare. Such a system must understand the intricate relationships between symptoms, diseases, and treatments. Tensor embeddings can represent these relationships numerically, allowing the neural network to recognize patterns and make informed predictions. This method retains the semantic integrity of symbolic relations, ensuring that the AI's reasoning aligns with established medical knowledge.

[Graph Neural Networks (GNNs)] are designed to operate directly on graph-structured data, making them ideal for modeling knowledge bases that naturally form networks of interconnected entities.

Imagine a recommendation system that suggests products to users based on their preferences and social connections. Users, products, and their interactions can be represented as nodes and edges in a graph. GNNs can learn embeddings that capture both the features of individual nodes and the overall structure of the graph. This can allow for more accurate recommendations by considering the complex web of relationships in the data.

GNNs can also integrate symbolic reasoning with neural learning. By incorporating logical rules and constraints from symbolic knowledge bases, GNNs enhance their predictive capabilities. For example, in drug discovery, GNNs can model molecular structures as graphs, embedding symbolic chemical knowledge to predict the efficacy of new compounds.

Some advanced GNN models explore embeddings beyond traditional Euclidean space. Techniques like [Quaternion Graph Neural Networks] leverage quaternion algebra to capture more complex relational patterns, providing richer representations of the symbolic knowledge embedded within the graph.

[Matrix factorization] facilitates the embedding of symbolic knowledge into neural networks. It involves decomposing large matrices representing relationships between entities into products of smaller

matrices, uncovering latent factors that capture underlying structures in the data.

When combined with neural networks, matrix factorization techniques enhance the AI's ability to model complex, non-linear relationships present in symbolic knowledge bases. For instance, in recommendation systems, integrating deep neural networks with matrix factorization allows the model to account for intricate user-item interactions and contextual information, leading to more personalized and accurate recommendations.

In knowledge representation, matrix factorization helps extract meaningful patterns from large, sparse symbolic data. Neural networks refine these patterns, enabling tasks like knowledge base completion and anomaly detection. This hybrid approach addresses the limitations of traditional matrix factorization, such as handling sparse data and capturing non-linear dependencies, by leveraging the representational power of neural networks.

[Ontologies] represent knowledge domains through a hierarchy of concepts and relationships, providing structured and semantic-rich symbolic knowledge. Embedding ontologies into neural networks requires methods that preserve this hierarchical structure.

[Hierarchical embedding techniques] map entities and their relationships into neural models while maintaining the ontological hierarchy. One effective method involves embedding the ontology into a hyperbolic space, which naturally accommodates hierarchical data due to its geometric properties. In hyperbolic space, distances grow exponentially, mirroring the branching nature of ontological hierarchies.

For example, in a semantic search application, embedding an ontology of concepts into a neural network allows the AI to understand queries at different levels of abstraction. A search for "electronic devices" can return results ranging from general information about electronics to specific products like smartphones or laptops, depending on the user's intent.

By embedding ontologies hierarchically, neural networks can better capture the relationships between general categories and specific instances. This enhances the AI's ability to perform reasoning tasks, such as the inheritance of properties in object-oriented representations, and improves performance in applications like natural language understanding and image recognition.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.1.2-Symbolic-Knowledge-Injection-Methods}4.1[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161962}.2 Symbolic Knowledge Injection Methods

Symbolic knowledge injection methods are essential techniques that embed human-understandable rules and relationships directly into neural architectures. There are three pivotal methods: injecting symbolic rules via constrained optimization, enhancing neural models with knowledge graph embeddings, and utilizing attention-based mechanisms to prioritize symbolic knowledge.

In [constrained optimization], symbolic rules are translated into mathematical constraints in the optimization process that guide the training of neural networks. Through this, the neural network learns not only from the data but also adheres to predefined logical relationships.

Consider a neural network designed to diagnose diseases based on patient symptoms. Medical knowledge, such as the fact that a high fever and rash often indicate measles, represents symbolic rules derived from clinical expertise. By formulating these rules as constraints, the network's learning algorithm is adjusted to prioritize solutions consistent with medical knowledge. Mathematically, this involves adding a constraint term to the loss function:

Total Loss = Data Loss + [λ]{.Times} × Constraint Loss

Here, [λ]{.Times} is a hyperparameter that balances the influence of the symbolic constraints against the empirical data loss. The constraint loss quantifies how well the network's predictions comply with the symbolic rules. This helps us get models that not only fit the data but also respect established domain knowledge.

This helps in achieving improved generalization and enhances the interpretability of the model's decisions. This is especially helpful in scenarios with limited data.

Another method for injecting symbolic knowledge into neural networks is through [knowledge graph embeddings]. Knowledge graphs are structured representations of entities and their interrelations, capturing rich semantic information about the domain. Embedding these graphs into neural networks allows models to leverage this structured knowledge during learning and inference.

For instance, in natural language processing tasks like question answering, incorporating a knowledge graph about world facts enables the AI to provide more accurate and contextually relevant responses. If a user asks, "What is the capital of France?", the AI can reference the embedded knowledge graph to confidently answer "Paris".

Knowledge graph embeddings transform entities and relationships from the symbolic graph into continuous vector spaces that neural networks can process. Techniques such as TransE, DistMult, and ComplEx are commonly used to create these embeddings. The embedded vectors capture the semantic proximity and relational patterns among entities, which the neural network can exploit to enhance its predictions.

By integrating these embeddings, neural networks gain access to a wealth of background knowledge, improving performance in tasks like recommendation systems, semantic search, and entity recognition. This allows AI models to reason with embedded knowledge.

The third method involves using [attention mechanisms] to prioritize symbolic knowledge within neural networks. Attention mechanisms enable models to focus selectively on important parts of the input data or knowledge sources when making predictions. By guiding the network's attention toward relevant symbolic information, the model can make more informed and accurate decisions.

For example, an AI system for legal document analysis needs to interpret complex texts while adhering to specific legal statutes (symbolic knowledge). Attention-based methods allow the network to concentrate on critical clauses and regulations pertinent to the case at hand, effectively integrating symbolic legal knowledge into the analysis.

In practice, symbolic knowledge can be encoded into the network as additional input features or embedded layers. The attention mechanism assigns higher weights to these symbolic components during training and inference. This means the network learns to give more importance to the symbolic knowledge when processing information.

The attention weights can be learned jointly with the network parameters, ensuring that the model dynamically adjusts its focus based on the context. This approach enhances the model's ability to handle tasks where certain rules or relationships are crucial, such as compliance checking, ethical decision-making, and domain-specific reasoning.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.2-Symbolic-Reasoning-with-Neural-Networks}4.2[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161965} Symbolic Reasoning with Neural Networks

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.2.1-Differentiable-Reasoning-Models}4.2[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161968}.1 Differentiable Reasoning Models

Differentiable reasoning models rely on the integration of logic operators within neural networks. This integration allows neural networks to process both symbolic and non-symbolic information.

One innovative approach involves embedding [differentiable logic gates] into neural architectures. By representing logical operations such as AND, OR, and NOT as differentiable functions, neural networks can learn logical relationships through gradient-based optimization. This method transforms traditional logic gates into

trainable components, enabling the network to adjust and refine its reasoning patterns during the learning process.

For instance, in [constraint satisfaction problems], neural networks equipped with differentiable logic operators can efficiently explore the solution space. The network learns to satisfy a set of constraints represented by logical expressions, adjusting its parameters to minimize violations. This approach leverages the parallel processing capabilities of neural networks, offering a scalable solution to problems that are computationally intensive for traditional symbolic methods.

In robotics, integrating differentiable logic operators allows robots to navigate complex environments by combining sensory data processing with logical decision-making. A robot can, for example, learn to recognize obstacles (through neural perception) and make decisions about movement based on logical rules (implemented via differentiable logic gates). This hybrid approach enhances the robot's ability to operate autonomously in dynamic settings.

Moreover, [hardware implementations] of neural networks with embedded logic operators have demonstrated the feasibility of constructing neural circuits that perform logical operations. These implementations showcase the versatility of neural networks, as they can be configured to replicate the behavior of standard logic gates while retaining the ability to learn and adapt.

While integrating logic operators into neural networks provides a structural foundation for reasoning. Training neural models to execute logical operations extends their capabilities even further. These models learn to perform reasoning tasks by processing data in a way that mimics logical inference. One challenge in this domain is ensuring that neural networks can generalize logical reasoning to new, unseen scenarios. Traditional neural networks excel at pattern recognition but often struggle with tasks requiring systematic reasoning, especially when faced with out-of-distribution data.

To address this, researchers have developed neural architectures that combine sequence processing with spatial alignment mechanisms. For example, a neural [sequence-to-grid] module can automatically segment and organize input sequences into a grid format, facilitating operations that require an understanding of the structure and relationships within the data. This approach has shown promise in solving arithmetic and algorithmic problems, where the network must follow a series of logical steps to arrive at the correct solution.

In [natural language processing], for instance, neural models trained with logical reasoning capabilities can interpret and answer complex questions that require understanding implications, negations, and logical relationships between statements. By learning to map linguistic input to logical forms and perform reasoning over these forms, the models achieve a deeper level of comprehension.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.2.2-Neural-Theorem-Provers-and-Reasoning-Pipelines}4.2[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161971}.2 Neural Theorem Provers and Reasoning Pipelines

Consider a scenario in [mathematical theorem proving]. Traditional symbolic theorem provers can methodically explore logical deductions but may falter when faced with vast or ambiguous datasets. By incorporating neural networks, the system gains the ability to identify promising paths through the space of possible proofs, guided by learned patterns from previous successful proofs.

This hybrid model leverages the strengths of both neural learning and symbolic reasoning, enabling more efficient and effective theorem proving.

Moreover, these integrated models can generalize learned reasoning strategies to new problems. As the neural component learns to recognize the structural features of logical arguments, it can assist the symbolic prover in navigating complex reasoning tasks

that would be computationally intensive using symbolic methods alone.

Beyond theorem proving, It can help AI systems that interact with the real world. By combining neural networks' ability to process raw sensory data with the logical reasoning of symbolic systems, AI can achieve a deeper understanding and more sophisticated decision-making capabilities.

Take, for instance, an AI system designed for [autonomous navigation]. The neural network component processes visual inputs, recognizing objects like pedestrians, vehicles, and traffic signs. Simultaneously, the symbolic reasoning module interprets this information within the context of traffic rules and safety protocols. The integrated system can then make informed decisions, such as yielding to a pedestrian or stopping at a red light, by reasoning about the implications of its perceptions.

Another application lies in [natural language understanding]. Neural networks often lack the ability to reason about the content they process. By integrating symbolic reasoning, AI systems can not only parse sentences but also infer meaning, detect contradictions, and draw logical conclusions from textual information.

For example, an AI assistant could read a series of medical articles and, through neural processing, extract relevant information about symptoms and treatments. The [symbolic reasoning] component could then infer potential diagnoses based on this information, providing valuable insights to healthcare professionals.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.3-Neuro-Symbolic-Frameworks-and-Hybrid-Learning}4.3[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161975} Neuro-Symbolic Frameworks and Hybrid Learning

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.3.1-Neuro-Symbolic-Frameworks}4.3[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161978}.1 Neuro-Symbolic Frameworks

Neuro-symbolic frameworks aim to meld the learning capabilities of neural networks with the interpretability and reasoning abilities of symbolic logic.

[Logic Tensor Networks (LTNs)]{.Bold}

[Logic Tensor Networks] [(LTNs)], for example, can integrate logical reasoning with neural networks by introducing [Real Logic], where logical formulas are assigned truth values between 0 and 1. This helps blend fuzzy logic with continuous neural computations. This, in turn, allows LTNs to perform deductive reasoning while leveraging the learning capabilities of neural networks.

LTNs ground logical constants as real-valued feature vectors. This grounding enables the network to handle complex relational data and perform reasoning tasks, that were traditionally the domain of symbolic AI. For instance, in multi-label classification, LTNs can learn relationships between different labels, improving accuracy by considering the logical connections among them.

Consider an application in semantic image interpretation. An LTN can analyze an image, recognize objects within it, and understand the relationships between those objects based on predefined logical rules.

For example, if an image contains a "cat" and a "sofa," the LTN can infer the likelihood of the cat sitting on the sofa by applying logical reasoning to the recognized objects.

Fig 4.3.1[[1]] shows an LTN architecture designed to compute the truth value of the logical implication "P(x, y) [⟶]{.Times} A(y)". In this figure, the inputs are the grounded vectors (v for x and u for y), which represent the numerical features of the objects.

Two separate branches process these inputs. One branch computes a value related to the predicate P, with operations that eventually yield "1 minus the truth value of P(x, y)", and the other branch directly computes the truth value of A(y).

These two outputs are then combined, using a max operation,

following fuzzy logic principles to determine the overall truth value of the implication.

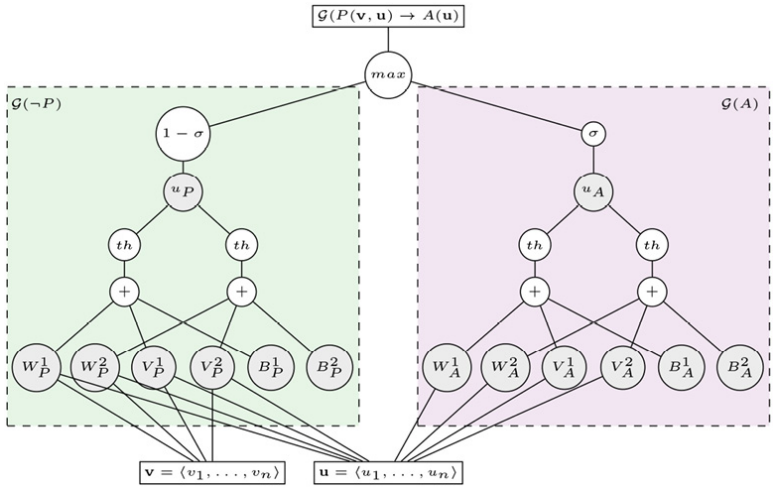


Figure 18. Fig_01.jpg

Fig 4.3.1

[DeepProbLog]{.Bold}

[DeepProbLog] is a framework that fuses probabilistic logic programming with neural networks by introducing probabilistic reasoning to traditional logic programming.

Certain components of the logic program are computed using neural networks, which can process raw data inputs like images or audio signals. The probabilistic aspect enables the framework to assign likelihoods to different logical outcomes, thereby managing uncertainty.

For example, in a digit recognition task, DeepProbLog can process images of handwritten digits using a neural network. It then incorporates logical constraints, such as the rules of arithmetic,

to reason about the digits. If the task involves solving equations, the system not only recognizes the digits but also understands the mathematical relationships between them.

One of the challenges with DeepProbLog is the computational complexity of combining neural network evaluations with probabilistic logical inference. To address this, researchers have developed approximate inference algorithms that significantly speed up computation without sacrificing much accuracy. These advancements have made DeepProbLog more practical for large-scale applications.

[Learning Interpretable Concepts]{.Bold}

[Learning interpretable concepts] helps AI systems become more transparent, allowing users to understand and trust their decisions.

It can be achieved by training neural networks to extract high-level concepts from raw data and then using symbolic reasoning to interpret these concepts. This method contrasts with traditional deep learning models, which act as "black boxes" with limited interpretability.

For instance, in time series classification, a neuro-symbolic model might use signal temporal logic to represent patterns over time. The neural network processes the raw time series data, extracting features like trends or anomalies. The symbolic component then expresses these features in human-readable formulas, providing clear explanations for the classification decisions.

In another example, consider an AI system designed for medical diagnosis. The neural network analyzes patient data, such as medical images or test results, identifying patterns indicative of certain conditions. The symbolic reasoning module then maps these patterns to medical concepts and diagnoses, presenting the findings in a way that doctors can understand and validate.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.3.2-Hybrid-Learning-and-Reasoning-Mechanisms}4.3.[]{#Neuro-Symbolic-AI-EPUB-

7.xhtml#x.161981}2 Hybrid Learning and Reasoning Mechanisms

Hybrid learning and reasoning mechanisms combine the pattern recognition strengths of neural networks with the logical precision of symbolic systems. These are general frameworks for building AI systems that require both learning from data and reasoning with knowledge. They provide a pathway to develop AI that can adapt to new situations while adhering to established rules and logic.

[Unified Training and Mutual Supervision]

[Unified training] methods optimize both neural networks and symbolic reasoning modules simultaneously. These methods enable mutual supervision, where the neural and symbolic components inform and improve each other during training. The two most common methods are [DeepLogic] and [Deep Symbolic Learning].

[DeepLogic], for example, integrates neural perception and logical reasoning into a single framework. It allows for the exchange of supervision signals between the neural network and the reasoning module. This mutual learning leads to significant performance improvements over traditional deep neural network baselines.

For natural language understanding, it can process raw textual data through a neural network, extracting linguistic features and patterns. Simultaneously, it employs a symbolic reasoning module that understands grammar rules and logical relationships within the text.

By training both components together, the system not only learns to recognize language patterns but also to reason about the meaning and context of the text, resulting in more accurate and interpretable language models.

[Deep Symbolic Learning (DSL)], on the other hand, learns neuro-symbolic functions within a differentiable neural network pipeline.

It creates interpretable symbolic representations that are directly mapped to perception inputs. As a result, the system can auto-

matically select the most relevant symbols that explain the data, enhancing both performance and interpretability.

For image recognition tasks, it can identify visual features and associate them with symbolic concepts like shapes or objects. If the system recognizes a circular shape with spokes, it can symbolically represent it as a "wheel." This makes the reasoning process transparent and understandable to humans.

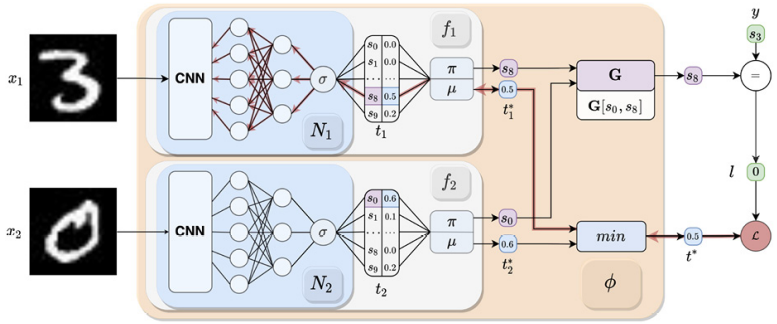


Figure 19. Fig_02.jpg

Fig 4.3.2

Fig 4.3.2[[2]] illustrates how DSL combines a perception module, typically a neural network that converts input images, such as handwritten digits, into symbolic representations, with a symbolic module.

It is a modular system where raw visual inputs, such as images of digits are first converted into discrete symbols via neural networks. These symbols are then combined through a learned symbolic rule, which acts like an addition table to produce a final result.

The entire process is differentiable, meaning the system can be trained end-to-end by propagating error signals, back through all components.

[Hybrid Architectures in Action]{.Bold}

[Hybrid architectures] integrate neural networks with symbolic reasoning modules, wherein neural network components handle perception tasks, and the symbolic reasoning module manages high-level cognitive functions such as planning, problem-solving, and logical inference.

An example could be advanced video surveillance systems. These systems use neural networks, specifically virtual neural sensors, to detect and recognize objects and activities within video feeds. The symbolic reasoning module then interprets these detections, applying logical rules to understand complex scenarios.

For instance, if the neural network detects a person entering a restricted area, the symbolic reasoning module can infer that a security protocol should be initiated. It can consider additional context, such as the time of day or the person's identity, to decide the appropriate response. This combination allows the system to operate efficiently, making real-time decisions based on both perceptual data and logical reasoning.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.4-Evaluation—Benchmarking—and-Optimization}4.4 E[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161985}valuation, Benchmarking, and Optimization

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.4.1-Evaluation-Metrics-and-Benchmarking}4.4.1[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161988} Evaluation Metrics and Benchmarking

Traditional evaluation metrics often fall short when applied to neuro-symbolic systems because they typically measure either the learning performance of neural networks or the reasoning accuracy of symbolic components, but not the synergy between them. Therefore, developing metrics that specifically gauge the effectiveness of neuro-symbolic integration is essential.

Developing specialized metrics requires a deep understanding of both neural and symbolic components, ensuring that the evaluation captures the nuances of their integration. As the field pro-

gresses, standardized metrics will become increasingly important for comparing different neuro-symbolic approaches and fostering advancements.

[Neuroscore and Human-Aligned Evaluation]{.Bold}

[Neuroscore] is an approach, which comes from generative adversarial networks (GANs). This leverages brain signals to evaluate the quality of generated images, aligning machine evaluation more closely with human perception. Taking inspiration from this, we could develop metrics that assess how well the integration enhances reasoning and decision-making capabilities.

For instance, metrics could measure the system's ability to generalize knowledge from data, the interpretability of its reasoning processes, or its proficiency in handling ambiguous or incomplete information.

[Statistical Measures and Human Benchmarks]{.Bold}

[Statistical correlation measures] can enable us to evaluate the consistency between system's outputs and established benchmarks or human judgments. This can help verify whether the neuro-symbolic system maintains performance improvements over purely neural or symbolic counterparts.

[Benchmark Datasets for Evaluation]{.Bold}

[Benchmarking datasets] such as CLEVR and bAbI have become standard tools for assessing visual reasoning and natural language understanding capabilities in neuro-symbolic systems.

[CLEVR] is a synthetic dataset designed to evaluate compositional visual reasoning. It comprises images of simple 3D objects and a set of questions that require understanding the relationships between these objects.

The controlled complexity of CLEVR makes it ideal for testing the reasoning abilities of AI systems without the noise and unpredictability of real-world images. Neuro-symbolic models have

leveraged CLEVR to demonstrate their proficiency in tasks that require both perception and logical reasoning.

For example, a system might be asked, “Are there more red cubes than yellow spheres?” Answering this question involves recognizing objects, their colors and shapes, and comparing quantities based on these attributes.

Extensions of CLEVR, such as [CLEVR-X] and [CLEVR-Hans], have been developed to push the boundaries further. CLEVR-X adds natural language explanations to the dataset, allowing researchers to assess a system’s ability to generate human-like explanations for its answers. This is crucial for interpretability and trust in AI systems. CLEVR-Hans focuses on object-centric reasoning in 3D environments, challenging models to understand spatial relationships and dynamics.

[bAbI] dataset targets textual reasoning and natural language understanding. It comprises a set of synthetic stories and questions designed to test various aspects of logical reasoning, such as deduction, induction, and coreference resolution.

For instance, a story might involve a sequence of events, and the system is asked to answer questions that require understanding the temporal order or causal relationships. Neuro-symbolic systems use bAbI to evaluate their ability to comprehend and reason about textual information in a logical manner.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.4.2-Scalability-and-Optimization-Techniques}4.4.2[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161991} Scalability and Optimization Techniques

Neuro-symbolic AI systems are inherently resource-intensive. They combine the data-driven learning of neural networks with the intricate reasoning of symbolic logic, leading to significant computational demands. To tackle this, distributed computing and model pruning emerge as a pivotal strategy for scaling.

[Distributed Computing for Scalability]{.Bold}

[Distributed computing] involves spreading computational tasks across multiple machines or processing units. For neuro-symbolic AI, this means dividing both the neural network computations and the symbolic reasoning processes to run concurrently across a distributed infrastructure.

This approach not only accelerates processing times but also enables the handling of larger datasets and more complex reasoning tasks.

For example, a real-time language translation and understanding system needs to process vast amounts of linguistic data while simultaneously applying grammatical rules and contextual reasoning. Distributed computing can allow the system to partition the neural language models and symbolic grammar rules across different servers. This division allows for parallel processing, significantly reducing latency and improving throughput.

However, distributing neuro-symbolic computations is not without challenges. Ensuring consistency across distributed neural networks and maintaining the integrity of symbolic reasoning processes requires sophisticated synchronization mechanisms.

[Model Pruning and Compression Techniques]{.Bold}

[Model pruning] and compression techniques are another crucial aspect of scaling neuro-symbolic AI. Traditionally, these were used in deep learning. It involves removing redundant or less significant components of a neural network, such as unnecessary neurons, layers, or connections.

This process results in a sparser network that requires less memory and computational power. Pruned models can sometimes evolve into different representations while maintaining similar accuracy levels.

This phenomenon suggests that pruning doesn't merely shrink the model but can lead to a form of reconfiguration, potentially uncovering more efficient architectures within the original model.

In the context of neuro-symbolic AI, pruning can help streamline the neural components, making them more efficient.

[Compression techniques], such as quantization and knowledge distillation, further optimize models by reducing the precision of weights or transferring knowledge from larger models to smaller ones. Applying these techniques to neuro-symbolic models helps in deploying them on resource-constrained devices like smartphones or embedded systems.

Combining pruning and compression techniques across both neural and symbolic components leads to more efficient neuro-symbolic models. These optimized models are not only faster and more resource-efficient but also easier to train and deploy, broadening the potential applications of neuro-symbolic AI.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.5-Integration-with-External-Knowledge-and-Security}4.5 I[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161994}ntegration with External Knowledge and Security

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.5.1-Leveraging-External-Knowledge-Sources}4.5.1[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.161997} Leveraging External Knowledge Sources

Neural networks excel at pattern recognition and learning from data, but they often lack the ability to reason with explicit knowledge. By incorporating external knowledge bases, we can imbue neural models with structured information, enabling them to perform more complex tasks such as understanding context, reasoning, and making informed decisions.

[KBLSTM], an abbreviation for knowledge as continuous representations within neural networks is a method, which involves encoding knowledge as continuous representations within neural networks. For example, it can enhance recurrent neural networks by integrating background knowledge through an attention mechanism. This allows the model to selectively focus on relevant

information from knowledge bases, improving tasks like entity and event extraction.

In natural language processing, augmenting text representations with structured data from knowledge graphs has proven beneficial for question-answering systems. By aligning textual data with semantic information from external sources, neural models can better comprehend and respond to queries, especially when dealing with ambiguous or complex questions.

Dialogue systems, such as chatbots and virtual assistants, greatly benefit from such integration of external knowledge. It allows the systems to select appropriate information, understand user intent, and generate more accurate and contextually relevant responses.

For instance, when a user asks a health-related question, a dialogue system equipped with medical knowledge bases can provide precise answers, suggest further reading, or even caution against misinformation. This integration not only improves the user experience but also builds trust in AI applications.

Similarly, in image classification tasks, incorporating structured prior knowledge from knowledge graphs helps neural networks understand and categorize images more effectively. By relating visual data to known concepts and relationships, models can achieve higher accuracy and explainability.

[Domain-Specific Integration]{.Bold}

In specialized fields like medicine and law, embedding domain-specific knowledge into AI systems is not just beneficial, it's essential. These domains require a deep understanding of complex, nuanced information that goes beyond general knowledge.

In [healthcare], AI systems that incorporate medical knowledge bases can significantly improve disease diagnosis, treatment planning, and patient care by accessing up-to-date medical research, clinical guidelines, and patient data.

Such systems can provide doctors with valuable insights that

enhance decision-making. For example, an AI model designed to detect anomalies in medical images can leverage knowledge about disease patterns, anatomy, and pathology. This integration enables the model to identify subtle indicators that might be missed by less informed systems.

In the [legal domain], AI systems that incorporate legal statutes, case law, and regulatory frameworks can assist lawyers and judges in research and decision-making processes. By understanding the intricate web of legal knowledge, AI can help identify relevant precedents, analyze legal documents, and even predict case outcomes.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x4.5.2-Security-and-Robustness-in-Neuro-Symbolic-Systems}4.5.2[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.162000} Security and Robustness in Neuro-Symbolic Systems

Neural networks are renowned for their ability to learn from data, but they can be vulnerable to adversarial attacks and unexpected inputs. Ensuring that these models behave reliably under various conditions is crucial, especially in high-stakes environments like autonomous driving or healthcare. [Symbolic validation] emerges as a vital tool for enhancing the robustness of neural networks.

[Eager falsification] is one such method, which accelerates robustness verification by partitioning the verification problem into smaller subproblems. By assessing the likelihood of each subproblem leading to a falsification, the method focuses computational resources on the most probable issues.

This approach has demonstrated significant speed improvements, making robustness verification more practical for complex models.

[Statistical robustness certificates] is another method, which provides efficient measurements of a neural network's resilience against random noise and semantic perturbations.

By quantifying robustness statistically, developers can gain insights

into how their models might perform in real-world scenarios where data can be noisy or incomplete.

[Neuro-Symbolic Verification Frameworks]{.Bold}

[Neuro-symbolic verification frameworks]enable the verification of complex properties that were previously unattainable with traditional techniques. For example, in autonomous vehicles, a neuro-symbolic system can be designed to ensure that the vehicle will always stop at a stop sign, regardless of environmental variations or sensor noise.

By incorporating logical specifications directly into the verification process, neuro-symbolic frameworks provide a higher level of assurance. They allow for the expression of safety and performance requirements in a way that is both interpretable and enforceable, bridging the gap between data-driven learning and rule-based reasoning.

[Securing Symbolic Knowledge]{.Bold}

Symbolic knowledge bases require storing and processing structured information that neural networks can utilize for reasoning tasks. Protecting the confidentiality and integrity of these knowledge bases is essential to maintain trust and prevent unauthorized access or manipulation. The principles of [integrity, confidentiality, and availability] form the cornerstone of information security in this context.

[Multi-level security models] are based on trusted computing principles. It aims to manage the privileges of different system components and users, and thereby prevent unauthorized access and reduce the risk of data leaks. This granular control ensures that only authorized processes can read or modify sensitive parts of the knowledge base.

Techniques like [threshold cryptography] and [secure decentralized erasure codes] offer solutions by enhancing privacy guarantees and ensuring that data remains intact even if parts of the system

are compromised. These methods distribute data in a way that requires multiple parties to collaborate for access, thereby increasing security.

[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#References}Refer[]{#Neuro-Symbolic-AI-EPUB-7.xhtml#x.162003}ences

[1] Serafini, L., & d'Avila Garcez, A. (2016). [Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge] (arXiv preprint arXiv:1606.04422). https://doi.org/10.48550/arXiv.1606.04422

[2] Daniele, A., Campari, T., Malhotra, S., & Serafini, L. (2022). Deep symbolic learning: Discovering symbols and rules from perceptions (arXiv:2208.11561v2). Retrieved from https://arxiv.org/abs/2208.11561v2

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml}

Basic-Text-Frame []{#Neuro-Symbolic-AI-EPUB-8.xhtml#Chapter-5-}Chapt[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162006}er 5:

Learning Symbolic Representations with Neural Networks

Figure 20. 162706.png

Symbolic Representations with Neural Networks is a subset of the neuro-symbolic techniques. It enables neural networks to [encode, approximate, or discover symbolic structures or patterns,] instead of using symbolic logic and neural networks side-by-side as separate modules. It enables neural models to internally develop representations that can be interpreted or used symbolically. It is about teaching the neural component itself to become fluent in a symbolic "language", even if implicitly, so that it can serve as a foundational layer for symbolic reasoning later on.

Sometimes, we want neural networks to also learn ideas that look like symbols or concepts, such as words, objects, or mathematical

variables. This is about teaching a neural network to create or handle these symbolic ideas inside its pattern-based way of thinking. For example, you train a neural network on many sentences until it starts to represent the meaning of words in a way that's similar to symbols we use in logic or language. This is like teaching a child to not only recognize a "cat" visually, but also to understand the concept of "cat" as something that can be used in logical thinking, such as "a cat is an animal."

Some key aspects include:

[• Representation learning for concepts and relations:] Neural models learn embeddings that correspond not only to perceptual patterns but also to semantic entities, predicates, and relational structures. Essentially, "vectors as symbols."

[• Differentiable approximations of symbolic operations:] Neural models may incorporate differentiable surrogates of logical operations (e.g., differentiable approximations to AND, OR, NOT) so that backpropagation can induce symbolic reasoning-like capabilities directly in the learned representations.

[• Discovery and interpretation:] The ultimate aim is often to discover latent symbolic structure (e.g., grammar rules, logical clauses, or knowledge graph edges) from raw data, thus bridging the gap from pattern recognition to structured knowledge without explicit hand-engineering of symbolic rules.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.1-Neural-Embeddings-of-Symbols}5.1 N[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162009}eural Embeddings of Symbols

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.1.1-Translational-Models}5.1.1[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162012} Translational Models

[Translational models] are techniques used to embed [knowledge graphs], i.e. networks of entities and their relationships into continuous vector spaces (lists of numbers). They represent both

entities, such as “Alice” or “Paris” and the relationships between them, for example “lives in” as low-dimensional vectors. The key concept is that relationships act like translations in this vector space. For example, if you have a rule like “Alice lives in Paris,” you can think of the relationship “lives in” as a shift that takes the vector for “Alice” closer to the vector for “Paris.”. Translational models have evolved from TransE to TransD.

[TransE] represents both entities and relations as vectors in the same space. It works well for simple relationships. However, it struggles with more complex relationships, such as one-to-many (one entity relates to many others), many-to-one, or many-to-many scenarios.

[TransR] refines the translational models by going a step further by creating separate vector spaces for entities and relations. Entities are first represented in an “entity space” and then projected into a “relation space” where the relation-specific reasoning happens.

[TransD] introduces even more flexibility through dynamic mapping matrices that depend on both the entity and the relation. This means it can adjust the way entities are translated based on the context of each relationship, making it more effective at modeling a wide variety of interactions.

[Temporal Translational Models]{.Bold}

Many knowledge graphs include temporal information (i.e., they change over time). For example, relationships can evolve. A person might live in one city at one time and another later. Time as an element can be incorporated in the temporal knowledge graphs.

[HyTE] and [Temporal TransE] models add time by creating temporal projections or embeddings. HyTE, for instance, projects entities and relationships onto hyperplanes that change over time, capturing how connections evolve.

[Polar temporal knowledge embedding (PTKE)] goes further by using a polar coordinate system to embed time-aware facts. It

treats time as a constraint and embeds both the starting and ending timestamps along with the entities, so the model can distinguish changes over time.

[Evaluation of Translational Models]{.Bold}

Translational models can be evaluated for link prediction tasks. This can see how well these models work, researchers use tasks where the model must predict missing links in the knowledge graph (i.e., infer unknown relationships). Common evaluation metrics are Mean rank, Mean reciprocal rank, and Hits@N.

[Mean rank (MR)] for example calculates the average position (rank) of the correct answer in the model's list of predictions. [Mean reciprocal rank (MRR)] however takes the average of the reciprocal (1 divided by the rank) of the correct answers, giving more credit when the correct answer is near the top. [Hits@N] calculates the percentage of times the correct entity appears in the top N predictions.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.1.2-Graph-Embeddings-and-Knowledge-Graph-Representation}5.1.2[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162017} Graph Embeddings and Knowledge Graph Representation

[Graph Neural Networks] (GNNs) have emerged as a powerful framework for embedding and reasoning over symbolic knowledge graphs. By leveraging the inherent structure of graphs, GNNs can capture the rich relationships and properties of entities within a knowledge graph. This capability is essential for tasks like entity classification, link prediction, and answering complex queries.

GNNs operate by propagating information along the edges of a graph, allowing each node to aggregate features from its neighbors. This enables the network to learn representations that reflect both local and global graph structures.

[Hierarchies and Hyperbolic Embeddings]{.Bold}

[Ontologies] often contain hierarchical structures that represent

relationships like “is-a” or “part-of.”. Traditional Euclidean spaces may not effectively represent hierarchical data due to their linear nature.

To address this, researchers have turned to non-Euclidean geometries, such as [hyperbolic space]. In hyperbolic space, the distance between points grows exponentially, mirroring the exponential growth of nodes in a hierarchy. Hyperbolic embeddings leverage the properties of hyperbolic space to represent hierarchical data efficiently.

[Dynamic Graph Embeddings]{.Bold}

Knowledge graphs are not static. They evolve over time as new information becomes available and relationships change. [Dynamic graph]embeddings aim to capture these changes by updating node representations as the graph structure evolves.

This adaptability is essential for applications like social network analysis, fraud detection, and recommendation systems, where timely updates are critical. For example, [autoencoders] learn embeddings that preserve both local and global structures. As new nodes and edges are added, the model updates the embeddings for affected nodes while keeping the representations of unchanged parts stable.

[Temporal Graph Neural Networks]{.Bold}

[Temporal graph neural networks] incorporate time as an explicit dimension. They model the change in node attributes and graph structure, capturing long-term dependencies through mechanisms like temporal self-attention. This enables the model to predict future states of the graph and understand temporal patterns.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.1.3-Multi-Modal-Symbolic-Embeddings}5.1.3[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162020} Multi-Modal Symbolic Embeddings

[Multi-modal symbolic embeddings] leverage data from various modalities, such as images, text, and spatial-temporal data into a

unified representation. This fusion enables machines to perform sophisticated reasoning tasks that were once considered challenging.

Visual reasoning tasks, like [Visual Question Answering] (VQA), require models to comprehend and interpret both visual content and textual queries. Joint embeddings of images and symbols serve as a bridge, allowing models to represent and process visual and linguistic information cohesively. By constructing image-graphs and question-graphs, models can represent objects in an image and words in a question as nodes, with edges denoting relationships or interactions. This structure enables the model to perform multistep reasoning by transferring features between related nodes.

For instance, consider a VQA system that needs to answer the question, "What color is the cat sitting on the chair?" The model builds a graph where nodes represent objects like "cat" and "chair," and edges represent the "sitting on" relationship. By jointly embedding the visual features of the cat and the chair with the symbolic representation of the question, the model can reason about their interaction and arrive at the correct answer.

[Neural-Symbolic Systems and Structured Reasoning]{.Bold}

Another approach focuses on neural-symbolic systems that recover structural representations from both images and questions.

The model interprets the image to create a scene representation and parses the question into a program trace. It then executes this program on the scene representation to generate an answer. This method provides a more interpretable reasoning process, as it mirrors how humans might deconstruct a problem into logical steps.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.2-Concept-Learning-and-Induction}5.2 C[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162023}oncept Learning and Induction

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.2.1-Inductive-Logic-

Programming-with-Neural-Networks}5.2.1[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162026} Inductive Logic Programming with Neural Networks

Traditional [inductive logic programming] (ILP) systems are great at deriving logical rules but often struggle with noisy data and scalability. Neural networks, on the other hand, are adept at processing large datasets and dealing with imperfections but lack interpretability. By combining these two methods, we can create systems that learn precise, interpretable rules even from noisy or complex datasets.

For example, in recognizing printed characters from a language with intricate scripts, an ILP system can learn the structural rules of the characters. A neural network can then handle the variability and noise in the actual images, matching them to the learned rules. This synergy results in higher accuracy than either approach could achieve alone.

[Addressing Limitations of Traditional ILP]{.Bold}

Handling [noise and ambiguity] in data is a significant challenge for ILP systems. [Scalability] is another area where traditional ILP systems face limitations. The combinatorial nature of logic programming can lead to inefficiencies when dealing with large datasets or complex search spaces.

Neural networks offer solutions to these challenges, enabling ILP methods to scale effectively. Novel neural architectures have been developed to learn first-order logical rules directly from data. These models require minimal prior knowledge or language bias, making them more adaptable to various domains.

They can process extensive knowledge graphs and relational data, extracting meaningful patterns and rules that can be interpreted and applied in different contexts.

[Differentiable ILP Frameworks]{.Bold}

[Differentiable ILP] frameworks have emerged as a promising

solution, blending neural computation with symbolic reasoning. It does so by representing logical deduction in a differentiable manner, allowing the use of gradient-based optimization methods common in neural network training. As a result, these systems maintain the data efficiency and generalization abilities of ILP while gaining robustness against noise.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.2.2-Symbolic-Abstraction-in-Neural-Networks}5.2.2[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162029} Symbolic Abstraction in Neural Networks

Symbolic abstraction involves representing complex concepts as symbols and manipulating them according to logical rules. Neural networks, when introduced with abstraction layers, concept bottlenecks, and techniques for extracting symbolic features, can perform symbolic reasoning.

This makes models become more transparent and interpretable, allowing users to understand and trust their decisions. This facilitates human intervention, correction, and guidance, enhancing overall performance. The ability to manipulate symbols and follow logical rules enables networks to perform complex reasoning tasks. In addition to this, symbolic representation can be transferred between tasks, enabling networks to learn from limited data through shared concepts.

[Bridging Raw Data and Symbolic Representation]{.Bold}

Architectures that incorporate [abstraction layers] enable in creating a bridge between raw data and high-level symbolic representations. Abstraction layers function by mapping the state space of neural networks onto an abstract space where symbolic reasoning can occur.

This mapping simplifies the network's outputs into [symbolic forms] that are easier for humans to interpret. One approach involves embedding symbolic structures directly into the neural ar-

chitecture. This can be done either by layers that enforce symbolic constraints or by integrating logic programming principles.

However, abstraction can obscure low-level details that might be crucial for optimizing performance, especially on specialized hardware platforms.

[Concept Bottleneck Models (CBMs)]{.Bold}

[Concept Bottleneck Models] (CBMs) networks predict intermediate human-interpretable concepts before making final decisions. This enhances transparency and allows for human intervention and correction at the concept level. It divides a network into two main parts. The first maps input data to a set of predefined concepts, and the second uses these concepts to make a prediction.

For example, in medical diagnosis, it first identifies symptoms (concepts) from patient data before diagnosing a disease. Thus enabling practitioners to understand the symptoms, the model considered, in order to verify their correctness.

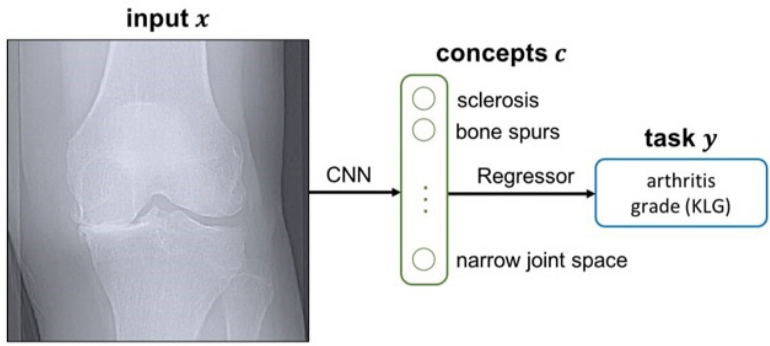


Figure 21. Fig_03.jpg

Fig 5.2.2

In fig 5.2.2[[1]], we first predict an intermediate set of human-specified concepts c, then use c to predict the final output y, for

knee x-ray grading.

Despite the advantages, CBMs require extensive labeled data for each concept, which can be labor-intensive to obtain and tends to underperform compared to traditional neural networks on complex tasks.

[Rule Extraction from Neural Models]{.Bold}

[Rule extraction algorithms] analyze the network's structure and outputs to generate symbolic rules that approximate its behavior. For example, decision tree algorithms like C4.5 can be employed to extract if-then rules that describe how input features lead to certain outputs.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.2.3-Neuro-Symbolic-Concept-Learners}5.2.3 []{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162032}Neuro-Symbolic Concept Learners

Symbolic concepts such as logical rules or expert insights can be [embedded] directly into neural networks during model training, to focus on relevant features and relationships.

Some embedding methods involve pre-training networks on essential concepts or rules, ensuring that the network has a foundational understanding before learning from data. Others convert symbolic rules into neural architectures, effectively translating expert systems into trainable neural networks.

Loss functions can be designed to penalize deviations from logical constraints, thereby aligning the network's outputs with symbolic knowledge.

[Applications and Domain Successes]{.Bold}

These hybrid approaches have shown success in various domains. In natural language processing, for example, neural language models have been enhanced by incorporating grammatical rules, leading to better language understanding and generation.

In computer vision, integrating object hierarchies and relationships helps models understand scenes at a conceptual level.

[Interactive Learning with Symbolic Feedback]{.Bold}

There are other [interactive] systems, where symbolic reasoning components provide [guidance] or [corrections] to neural networks during training. This interaction creates a dynamic learning environment where the neural model can adapt based on logical constraints or expert knowledge. Frameworks that support this [interactive learning] often include mechanisms to adjust the influence of symbolic feedback.

This adaptability is crucial because it allows the system to handle contradictory or uncertain knowledge. By learning to weigh the importance of different logical constraints, the neural network can become robust to noisy or incomplete symbolic information.

Consider a scenario in which a neural network is learning to classify legal documents. A symbolic reasoning system equipped with legal regulations and precedents can provide feedback when the neural network's predictions violate legal logic. For example, if the network incorrectly associates unrelated legal terms, the symbolic component can correct this by highlighting the logical inconsistencies.

[Hierarchical Concept Learning]{.Bold}

[Hierarchical concept learning]models aim to induce and represent concepts at multiple levels of abstraction, mirroring the way humans understand and organize knowledge. The model learns basic concepts from raw data and progressively builds more complex ideas by combining these foundational elements.

An application of hierarchical concept learning can be seen in robotics, where a robot learns to perform tasks by understanding sequences of actions. The robot might first learn individual movements, then how these movements combine to perform simple tasks, and eventually how to sequence tasks to achieve a goal.

The symbolic component helps the robot to plan and reason about actions at different levels of abstraction.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.3-Hybrid-Representations-and-Multi-Modal-Symbolic-Learning}5.3 Hy[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162035}brid Representations and Multi-Modal Symbolic Learning

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.3.1-Integrating-Symbolic-and-Subsymbolic-Representations}5.3.1 []{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162038}Integrating Symbolic and Subsymbolic Representations

Dual representation neural models serve as a unifying framework that allows neural networks to process both symbolic and subsymbolic data. These models are designed to handle structured information while learning from raw data, effectively merging two traditionally separate AI paradigms.

One approach involves using recursive neural networks capable of processing structured inputs, such as trees or graphs. These networks can represent fuzzy tree automata, enabling them to incorporate prior knowledge expressed as fuzzy state transition rules. This combination allows the model to learn unknown rules from data while adhering to predefined logical structures.

For instance, in natural language processing, understanding the grammatical structure of sentences is crucial. A dual representation model can parse sentences into parse trees (symbolic representation) while simultaneously learning word embeddings (subsymbolic representation). This enables the model to comprehend both the syntax and semantics of language, improving tasks like machine translation or sentiment analysis.

Another method utilizes neural networks to implement expert system rule conditions. In this setup, the neural network processes complex sensory inputs and outputs signals that a symbolic inference engine can use directly. These “detector predicates” act as

a bridge between the subsymbolic data processed by the neural network and the symbolic rules used for reasoning.

Furthermore, parameterized neural networks offer a way to adjust the balance between symbolic and subsymbolic processing. By tuning feedback strengths within the network, components can function either as traditional neural units or mimic logical rules and digital memory. This flexibility allows the model to adapt to different tasks, leveraging the most appropriate form of representation when needed.

These dual representation models have shown significant promise across various applications, from computer vision to robotics. They enable systems to perform high-level reasoning while maintaining the ability to learn and adapt from raw data inputs.

[Alignment Techniques for Symbolic and Subsymbolic Data]{.Bold}

Ensuring consistency between symbolic and subsymbolic representations is critical for the effective functioning of dual representation models. Alignment techniques are methods used to synchronize these two forms of data, allowing for seamless interaction within a single framework.

One innovative approach is the use of neuro-symbolic systems like [DeepProbCEP], which combines neural networks with symbolic complex event processing. This system can process subsymbolic data such as images and audio while using symbolic rules to define complex events. By injecting human knowledge into the learning process, DeepProbCEP can train effectively even with sparse data, aligning the symbolic and subsymbolic components for better performance.

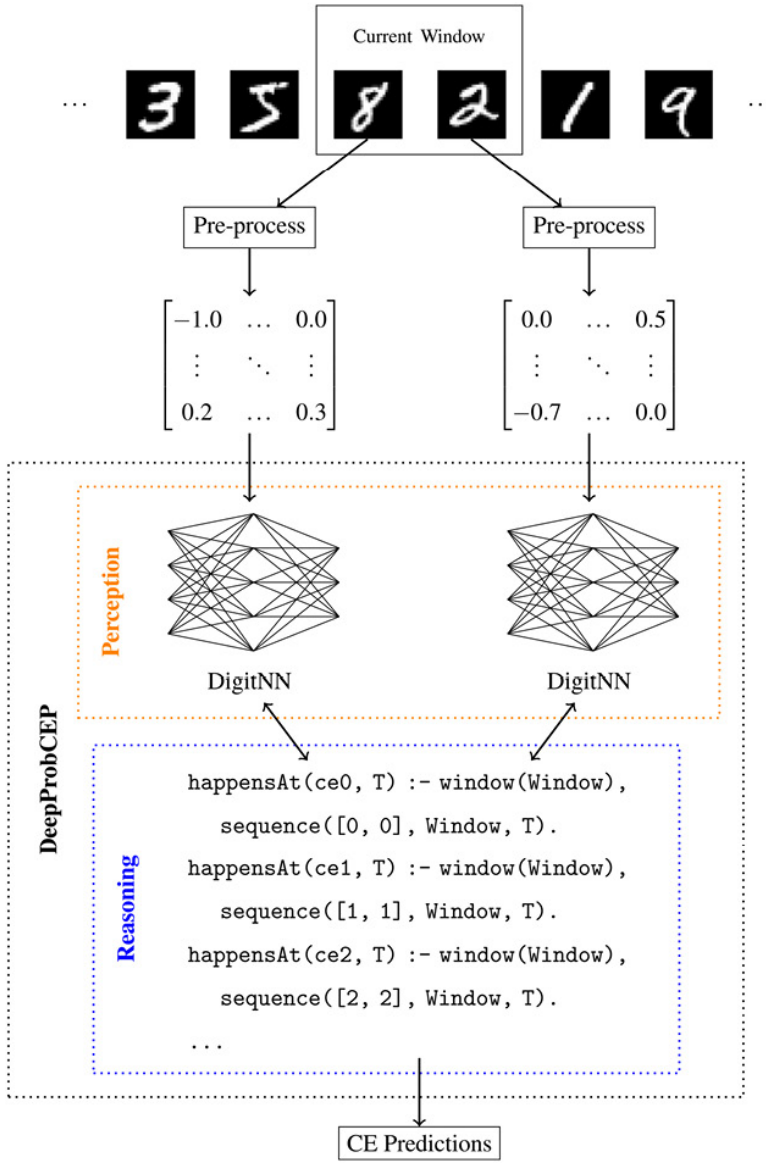


Figure 22. Fig_04.jpg

Fig 5.3.1

Figure 5.3.1[[2]] shows how DeepProbCEP integrates a neural network, in order to perceive and classify subsymbolic data like images, with a symbolic reasoning component, in order to apply complex event rules in an end-to-end system. This design allows the system to process raw data, extract meaningful information, and then reason about it to detect complex events, even when training data is sparse or when operating under adversarial conditions.

At the very top, a stream of input data, as MNIST images is first normalized to prepare it for further analysis.

The pre-processed digits then enter the perception layer, labeled as "DigitNN." This is a neural network that "sees" the images and classifies them by determining the likelihood of each digit, 0 through 9. Essentially, DigitNN converts the raw image data into a set of symbolic outputs, such as "this image is a 0 with 90% confidence".

Next, these classified outputs are passed to the reasoning layer. This layer uses predefined rules to decide whether a complex event is occurring. For instance, one rule might state that if a particular digit, for example, 0 appears at the end of a time window and also somewhere earlier in that window, then a complex event labeled "ce0" is triggered.

Other than DeepProbCEP, there is another technique, which involves generating a symbolic counterpart of a subsymbolic model automatically. For example, a neural network controlling an autonomous robot can be translated into a symbolic model that represents its decision-making process. This symbolic model can then be analyzed or modified, providing insights into the robot's behavior and ensuring it aligns with expected logical outcomes.

[Parameterization methods]also play a role in alignment. By configuring neural networks to adjust their internal parameters, it's possible to have certain components emulate logical rules while others handle subsymbolic processing. This creates a cohesive

system where symbolic reasoning and subsymbolic learning are not just coexisting but are interdependent and consistent.

These alignment techniques are essential for applications where both high-level reasoning and low-level data processing are required. They ensure that the symbolic logic governing the system's decisions is in harmony with the patterns learned from data, leading to more robust and reliable AI systems.

[Cross-Modal Symbolic Learning Methods]{.Bold}

Extending symbolic learning across different data modalities, such as text, images, and audio is a complex challenge. Cross-modal symbolic learning methods aim to create models that can understand and generate symbols irrespective of the data type, enhancing the versatility and applicability of AI systems.

Adversarial learning techniques have been effectively employed in this domain. For instance, methods like [Cross-Modal Self-Attention Learning (CMSA)] utilize self-attention networks to generate modality-specific representations. By employing adversarial learning, these models can align feature distributions across different modalities, such as images and textual descriptions, enabling effective cross-modal search and retrieval in social networks.

In the realm of audio and text, deep neural networks combined with canonical correlation analysis have been used to learn the temporal structures that correlate audio signals with corresponding lyrics. This is particularly useful in music information retrieval, where understanding the relationship between the melody and the lyrics can enhance search and recommendation systems.

Moreover, recent advancements include graph-based models that capture fine-grained semantic correspondences across modalities. These models represent data as nodes and edges in a graph, allowing for the integration of complex relationships between different types of data. Techniques like leveraging machine translation have also been used to enrich caption diversity for images, improving the performance of models in tasks like image captioning.

Interestingly, studies have shown that the effectiveness of cross-modal learning can depend on factors like study strategies and retention intervals. For example, some experiments have found that visual-only presentations with on-screen text can lead to better long-term retention compared to audio-visual materials. This highlights the importance of tailoring cross-modal learning methods to the specific requirements of the task and the characteristics of the target audience.

Overall, cross-modal symbolic learning methods are expanding the capabilities of AI systems, enabling them to understand and generate symbols across various forms of data. This is crucial for developing applications like multi-modal search engines, content recommendation systems, and advanced human-computer interaction interfaces.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.3.2-Multi-Task-Symbolic-Representation-Learning}5.3.2 M[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162041}ulti-Task Symbolic Representation Learning

Let's learn how shared symbolic representations facilitate knowledge transfer, how task-specific symbolic modules are incorporated within neural frameworks, and the techniques employed to balance generalization and specialization in symbol learning.

[Multi-task learning] (MTL) leverages the commonalities among different tasks to improve overall performance. By sharing representations, models can transfer knowledge learned from one task to another, reducing the need for large amounts of task-specific data. Symbolic embeddings play a pivotal role in this process, serving as a common language that bridges diverse tasks.

One innovative approach involves the use of hyperbolic embeddings to capture hierarchical structures within symbolic data. Unlike traditional Euclidean embeddings, hyperbolic spaces can represent complex hierarchies more efficiently, allowing models to understand relationships at multiple levels of abstraction. This

capability is particularly beneficial when dealing with tasks that involve taxonomies or ontologies, where the data naturally forms a tree-like structure.

For example, in natural language processing, understanding the relationships between words, phrases, and sentences is essential. Hyperbolic embeddings can represent these relationships more effectively, enabling models to perform tasks like semantic parsing and machine translation with greater accuracy. By sharing these embeddings across tasks, the model can apply linguistic knowledge gained from one task to improve performance on another.

Interestingly, some studies have found that simpler techniques, such as pooling encoders and pre-trained word embeddings, can perform on par with more complex multi-task learning approaches in certain scenarios. This highlights the importance of carefully selecting the right level of complexity for the task at hand. Overly complex models may not always yield proportional performance gains and can introduce unnecessary computational overhead.

To enhance knowledge transfer further, some models integrate both shared and task-specific embeddings. The Shared and Task specific EMbeddings (STEM) paradigm exemplifies this approach by combining common embeddings that capture general features with specialized embeddings that focus on task-specific nuances. This combination allows the model to generalize across tasks while retaining the ability to specialize when necessary.

For instance, in recommendation systems, users may exhibit different preferences across various categories like movies, books, or music. By sharing embeddings that represent general user behavior and incorporating task-specific embeddings for each category, the system can provide more accurate and personalized recommendations.

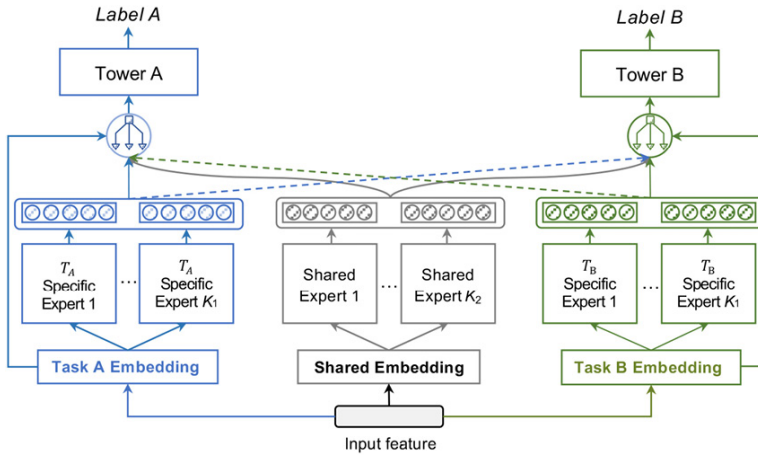


Figure 23. Fig_05.jpg

Fig 5.3.2

Fig 5.3.2[[3]] shows how STEM-Net takes raw input features and transforms them into two types of embeddings. These embeddings are then processed by separate expert networks and later combined in a way that carefully balances common knowledge with task-specific details. This structure enables the model to learn nuanced user preferences across different recommendation tasks, ultimately leading to better overall performance.

These shared symbolic representations are instrumental in promoting knowledge transfer, reducing data requirements, and improving the generalization capabilities of multi-task learning models.

[Task-Specific Symbolic Modules within Neural Frameworks]{.Bold}

While shared representations are essential for knowledge transfer, incorporating specialized symbolic reasoning modules within neural architectures allows models to handle task-specific requirements effectively. This integration ensures that while the model benefits from shared knowledge, it doesn't lose the ability to focus

on the unique aspects of each task.

One approach to achieving this integration is by translating logic programs into neural network structures. By embedding logical reasoning directly into the neural architecture, models can perform deductive reasoning tasks that are challenging for traditional neural networks. For example, converting a set of logical rules into a neural network allows the model to reason about relationships and constraints inherent in the data.

The Modalities Algorithm takes this concept further by translating modal logic programs into ensembles of neural networks. This enables the representation and reasoning of modal logics, which are essential in fields like knowledge representation and reasoning, within a neural framework. Such an approach allows the model to handle concepts like necessity and possibility, expanding its reasoning capabilities.

Another innovative method is the Logic-Integrated Neural Network (LINN), which constructs dynamic neural architectures based on input logical expressions. LINN learns basic logical operations as neural modules and assembles them to perform propositional logical reasoning. This dynamic construction allows the model to adapt its architecture based on the specific logical reasoning required for a task, providing flexibility and efficiency.

For example, in robotic planning, a robot may need to reason about the sequence of actions required to achieve a goal while considering constraints like safety and resource limitations. By incorporating task-specific symbolic modules, the neural network can plan and reason about its actions more effectively.

These task-specific modules ensure that the model maintains high performance on individual tasks while still benefiting from the shared knowledge present in the shared representations.

[Balancing Generalization and Specialization Strategies]{.Bold}

Achieving the optimal balance between generalization and special-

ization is critical in multi-task symbolic representation learning. Models must generalize well across tasks to leverage shared knowledge while also specializing sufficiently to excel in individual tasks. Various techniques have been developed to optimize this balance.

Balancing Generalization and Specialization Network can exemplify an end-to-end network that combines both abilities at instance and dataset levels. It employs a Generalization Network that uses episodic meta-learning to capture generalized knowledge across tasks. Simultaneously, it features a Balanced Specialization Network comprising multiple attentive extractors that focus on task-specific discriminative features.

A self-adjusted diversity loss function optimizes the specialization network by encouraging diversity among the specialized components. This mechanism ensures that each task-specific extractor focuses on different aspects of the data, reducing redundancy and improving overall performance. Additionally, a differentiable dataset-level balance simulates network pruning, optimizing the network's structure for both generalization and specialization.

In the realm of natural language processing, balancing these strategies is crucial for tasks like named entity recognition (NER) and relation extraction (RE). A novel neural architecture might introduce additional task-specific layers for each task, allowing for deeper specialization without compromising the shared representations. This approach has led to state-of-the-art results while maintaining computational efficiency.

Moreover, the concept of balancing generalization and specialization extends beyond traditional machine learning tasks. In cloud computing, dynamic load-balancing strategies use deep learning models combined with reinforcement learning to distribute workloads efficiently. By generalizing across different types of workloads and specializing in resource allocation strategies, these systems achieve optimal performance.

These techniques demonstrate that carefully designed architectures

and optimization strategies can effectively balance the needs for generalization and specialization. By doing so, models can harness shared knowledge while excelling in individual tasks, leading to more robust and versatile AI systems.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.3.3-Symbolic-Representation-Learning-for-Explainability}5.3.3 Sy[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162044}mbolic Representation Learning for Explainability

Understanding the decisions made by AI models is essential, especially in critical applications like healthcare, finance, and autonomous systems. Symbolic representation learning offers a pathway to interpretability by embedding knowledge in forms that are more aligned with human reasoning. This section explores how learning interpretable symbolic representations can enhance model transparency, the methods for generating human-readable explanations using these symbols, and the metrics for evaluating their clarity.

The complexity of deep learning models often makes them black boxes, obscuring the reasoning behind their outputs. To address this, researchers are developing methods to learn symbolic embeddings that maintain the performance of neural networks while enhancing interpretability.

One innovative approach combines symbolic planning with deep reinforcement learning to form a framework known as Symbolic Deep Reinforcement Learning (SDRL). In this setup, the model learns to associate high-level symbolic actions with low-level neural network options. This hierarchical structure allows the model to plan and reason about tasks in a way that is more transparent to humans. For example, in a robotics application, the robot can plan its actions using symbolic representations like “move forward” or “pick up object,” which are linked to complex motor control policies learned through reinforcement learning.

Require: $(I, G, D, \mathbb{F})$ where $G = (quality > 0)$, and an exploration probability ϵ

Initialization: $P_0 \Leftarrow \emptyset$, $\Pi_0 \Leftarrow \emptyset$
for t = 1 ... end of episodes **do**
  $\Pi^* \Leftarrow \Pi_{t-1}$
  take ϵ probability to solve planning problem and obtain a plan $\Pi_t \Leftarrow$ CLINGO.$solve(I, G, D \cup P_{t-1})$
  if $\Pi_t = \emptyset$ **then**
    return Π^*
  end if
  for symbolic transition $\langle s, a, s' \rangle \in \Pi_t$ **do**
    obtain current state $\tilde{s}$
    correspond to subtask g by using $\mathbb{F}$ to obtain initiation set and terminate condition
    while $\beta(\tilde{s}) \neq 1$ and maximal step is not reached **do**
      pick up an action $\tilde{a}$ and obtain transition $(\tilde{s}, \tilde{a}, \tilde{s'}, r_i(\tilde{s'}))$
      store transition in experience replay buffer $\mathcal{D}_g$
      estimate $Q(\tilde{s}, \tilde{a}; \theta, g)$ by minimizing loss function (5) when there are sufficient samples in $\mathcal{D}_g$
      update current state $\tilde{s} \Leftarrow \tilde{s'}$
    end while
    calculate extrinsic reward $r_e(s, g)$
    update $R(s, g)$ and $\rho^g(s)$ using (6).
  end for
  calculate quality of Π_t by (2).
  update planning goal $G \Leftarrow (quality > quality(\Pi_t))$.
  update facts $P_t \Leftarrow \{\rho(s, a) = z : \langle s, a, s' \rangle \in \Pi_t, \rho_t^a(s) = z\}$
end for

Figure 24. Fig_06.jpg

Fig 5.3.3

Fig 5.3.3[[4]] explains the algorithm for SDRL.

Another promising technique involves embedding hierarchical symbolic data into hyperbolic space, such as an n-dimensional Poincaré ball. Hyperbolic embeddings are particularly effective at

capturing hierarchical relationships inherent in symbolic data. For instance, in natural language processing, words can be embedded in a hyperbolic space to reflect their semantic hierarchies. Words like "animal," "mammal," and "dog" naturally form a tree-like structure that hyperbolic space can represent efficiently. This method not only enhances interpretability but also improves the model's ability to generalize from limited data.

[Genetic programming-based symbolic regression (GPSR)] is another avenue for learning interpretable models. By using a Bayesian framework, GPSR can quantify uncertainty and provide probabilistic predictions. This approach generates models that are inherently interpretable because they are constructed from mathematical expressions that describe the data. For example, in scientific data analysis, GPSR can uncover underlying physical laws from experimental data, presenting them in the form of human-readable equations.

These methods contribute to making AI models more transparent by embedding knowledge in symbolic forms that are easier for humans to understand and analyze.

[Generating Explanations from Symbolic Representations]{.Bold}

Having a symbolic representation is only part of the journey toward explainable AI; the next step is generating human-readable explanations from these symbols. This involves translating the internal symbolic knowledge of the model into explanations that make sense to users.

One method to achieve this is by leveraging ontologies as background knowledge to enrich explanations. By mapping low-level model features to higher-level concepts defined in an ontology, explanations become more meaningful. For instance, in a medical diagnosis system, instead of indicating that certain pixel intensities in an image led to a diagnosis, the system can explain that it detected "inflammation patterns consistent with arthritis" based on learned symbolic representations.

Another approach focuses on the interaction between humans and AI systems, emphasizing the importance of symbols in communication rather than internal reasoning. In this context, the AI system doesn't necessarily use symbolic reasoning internally but can generate symbolic explanations that are understandable to humans. For example, a recommendation system might suggest a movie and explain, "You might like this film because it is directed by the same director as your favorite movie," providing a symbolic rationale for its suggestion.

These methods highlight that the value of symbolic representations lies not just in model transparency but also in fostering effective human-AI collaboration. By providing explanations that align with human reasoning, AI systems can build trust and facilitate better decision-making.

[Evaluating Explainability in Symbolic Learning]{.Bold}

To ensure that symbolic representations genuinely enhance explainability, it's essential to have metrics and frameworks for assessing their clarity and usefulness. Evaluating explainability is a multifaceted challenge that involves both objective measurements and subjective user assessments.

One quantitative approach involves fidelity metrics, which assess how accurately an explanation reflects the model's actual decision-making process. High-fidelity explanations closely mirror the internal workings of the model, providing users with insights into how inputs are transformed into outputs.

Another framework, known as the Co-12 properties, identifies key conceptual properties for evaluating explanations, such as correctness, completeness, and comprehensibility. By scoring explanations against these properties, researchers and practitioners can systematically assess the quality of the explanations generated by their models.

In domains like knowledge graph-based recommender systems, specialized metrics like the Max Explainability Score have been

developed. This metric considers factors such as the number of rules used in an explanation, the probabilities of traversal paths in a knowledge graph, and entropy values to quantify the clarity and informativeness of explanations.

Importantly, evaluating explainability also involves user studies to gather feedback on how end-users perceive and understand the explanations provided. This qualitative aspect ensures that the explanations are not only theoretically sound but also practically useful to those who rely on them.

By developing robust evaluation methods, the field can advance toward AI systems that are not only powerful and accurate but also transparent and trustworthy.

The fusion of neural networks with symbolic reasoning marks a significant milestone in the evolution of artificial intelligence. As we push the boundaries of what AI can achieve, one area garnering considerable attention is the development of probabilistic symbolic representations. These advanced techniques enable machines to handle uncertainty more effectively, integrating the strengths of probabilistic logic with neural networks.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.4-Advanced-Techniques-in-Symbolic-Representation-Learning}5.4 Adva[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162049}nced Techniques in Symbolic Representation Learning

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.4.1-Probabilistic-Symbolic-Representations}5.4.1 Pro[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162052}babilistic Symbolic Representations

In real-world scenarios, uncertainty is the norm rather than the exception. [Probabilistic symbolic representations] provide a robust framework for managing this uncertainty, combining the interpretability of symbolic logic with the flexibility of probabilistic models.

For example, in autonomous driving, an AI system might use

probabilistic logic to reason about the likelihood of different traffic scenarios, such as a pedestrian suddenly crossing the street or another vehicle making an unexpected turn.

This method integrates probabilistic logic directly into neural architectures, which enables AI systems to learn from data while also reasoning about probabilities in a structured, logical manner.

Another innovative approach is [probabilistic fuzzy neural networks (PFNNs)], which incorporate fuzzy logic. This deals with reasoning that is approximate rather than fixed and exact, into neural networks. This allows the model to adaptively handle uncertainties in the data. For instance, in environmental monitoring, PFNNs can process imprecise sensor readings to make accurate predictions about pollution levels or climate changes.

[Stochastic neighbor embedding (SNE)] is a method used for dimensionality reduction that models the similarities between high-dimensional data points as probabilities. By embedding these data points into a lower-dimensional space, SNE preserves the probabilistic relationships, allowing for effective visualization and analysis. For example, in natural language processing, words can be embedded in a stochastic space where the distances reflect the probabilities of their co-occurrence, capturing semantic relationships in a meaningful way.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.4.2-Temporal-and-Sequential-Symbolic-Representations}5.4.2 Tem[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162055}poral and Sequential Symbolic Representations

Where timing and sequence matter, AI systems must grasp static facts as well as how those facts evolve over time. [Temporal and sequential symbolic representations] can help neural networks to model, reason, and predict temporal events, enhancing their applicability in fields like natural language processing, robotics, and financial forecasting.

One approach involves translating [temporal logic theories] into

neural network ensembles. For example, in automated planning, an AI system might use temporal logic to ensure that certain tasks are completed before others commence, adhering to constraints like "the foundation must be laid before walls can be built" in construction planning.

Understanding [sequences] is fundamental for tasks like language translation, speech recognition, and DNA sequence analysis. Sequential symbol embedding techniques aim to represent sequences of symbols in continuous vector spaces, capturing both the content and the order of the elements.

Techniques such as [Long Short-Term Memory (LSTM)] networks have been employed to encode multiple sequences into a unified representation. This is particularly useful in applications where multiple streams of data need to be processed simultaneously, such as sensor data fusion in autonomous vehicles.

One effective strategy for [time-series] data is [temporal abstraction], where continuous time-series data is converted into higher-level symbolic representations. This abstraction can enhance the generalization capabilities of neural networks, leading to improved performance in classification tasks.

For instance, in healthcare monitoring, raw physiological signals like heart rate and blood pressure can be abstracted into symbols representing states like "normal," "elevated," or "critical," which simplifies the detection of health anomalies.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.4.3-Compositional-Symbolic-Representations}5.4.3 Com[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162058}positional Symbolic Representations

Understanding and generating complex symbols from simpler ones is akin to how humans comprehend language and abstract concepts. In AI, traditional neural networks often struggle with this, as they lack the inherent structure to represent symbolic reasoning processes.

Compositional symbolic representation deals with the challenge of teaching neural networks to build and manipulate complex symbols from basic elements. [Vectors Approach to Representing Symbols (VARS)] and [Neural-Symbolic Stack Machine (NeSS)] are two such methods.

[Vectors Approach to Representing Symbols (VARS)] enables standard neural architectures to encode symbolic knowledge explicitly at their output layers. It allows networks to achieve combinatorial generalization, meaning they can understand and generate new combinations of known symbols.

For example, a neural network trained to recognize individual words can, through combinatorial generalization, comprehend novel sentences constructed from those words.

[Neural-Symbolic Stack Machine (NeSS)] combines the sequence processing capabilities of neural networks with the recursive power of symbolic stack machines. This allows for the execution of complex language-related tasks without the need for supervision during training on execution traces. It can understand nested structures in language, such as embedded clauses in sentences, which are essential for accurate language comprehension and generation.

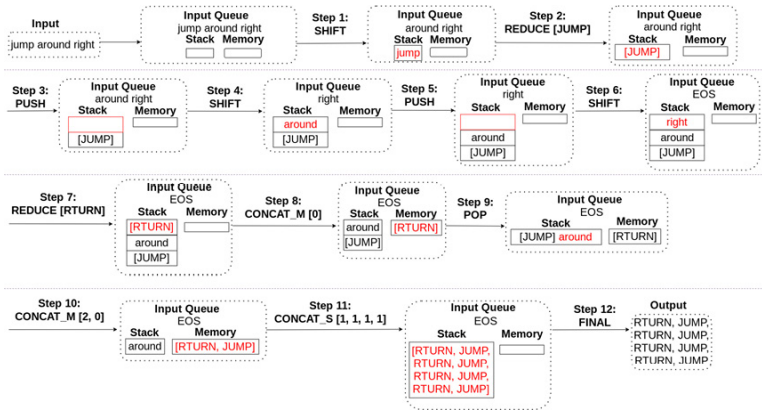


Figure 25. Fig_07.jpg

Fig 5.4.3

Fig 5.4.3[[5]] shows how the stack machine breaks down a navigation command into manageable pieces, processes each piece with its specialized operations, and then reassembles everything into the correct sequence of actions.

[SHIFT][ ]step reads the next token, a word or symbol) from the input stream and appends it to the end of the current top frame of the stack. Think of it as "pulling in" one word from the sentence.

[REDUCE [t_1, t_2, ..., t][☒][],] takes that group and "reduces" or transforms it into a new sequence in the target language. For example, an action sequence. The list [t_1, t_2, ..., t[☒]{.Times}] tells the machine how to combine or interpret those tokens.

[PUSH] operation creates a new frame on top of the existing stack. A new frame is like a fresh workspace that isolates part of the processing. It's used when the machine needs to start processing a nested or independent part of the input.

[POP]removes (or "pops") the top frame off the stack. The data from the popped frame is then appended to the new top frame, effectively merging the processed segment with what has been built

so far.

[CONCAT_M [i_1, i_2, ..., i][⊠][]] operator takes specific items from the current top of the stack and from a separate memory storage (using the provided indices i_1, i_2, ..., i[⊠]), concatenates (or joins) them, and then stores the resulting sequence back into the memory. This is useful for keeping intermediate results that may be used later.

[CONCAT_S [i_1, i_2, ..., i][⊠][]] is similar to CONCAT_M, this step concatenates items chosen by the given indices. However, instead of storing the result in memory, it places the combined sequence directly back onto the top of the stack. This helps to update or refine the current working segment.

During the [FINAL] step, the machine outputs the sequence that is currently on the top of the stack as the final result. In our example, that final sequence is the translated or action sequence corresponding to the input command.

[Hierarchical compositional learning] introduced the concept of multiple levels of abstraction. It mirrors how humans learn complex concepts, by first understanding simpler components and then combining them into higher-order structures. This has shown remarkable success.

In the realm of [music transcription], it can learn unsupervised representations of musical structures. It starts with basic elements like individual notes and progressively build up to recognize chords and entire melodies. It can transcribe music accurately, even capturing the nuanced interplay between different musical elements.

For [Object recognition], hierarchical Bayesian frameworks can learn generic low-level features such as edges and textures. These features are then combined to form high-level concepts like shapes and objects. This enables in learning new concepts from limited examples or where data is scarce.

For [multi-agent reinforcement learning], hierarchical approaches

allow agents to break down complex tasks into simpler sub-tasks across different levels and time scales. For instance, in a robotic assembly task, high-level goals like "assemble the device" can be decomposed into mid-level tasks like "attach component A to B," which are further broken down into low-level actions like "move arm to position X." This enables agents to learn and execute complex behaviors more efficiently.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x5.5-Benchmark-Datasets}5.5 Benchm[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162061}ark Datasets

Benchmark datasets provide standardized environments where models can be tested for their reasoning capabilities, generalization skills, and adaptability across various domains. [Synthetic data] allows for the controlled generation of datasets where specific variables and conditions can be manipulated. This is particularly useful for testing models on rare events or specific reasoning tasks that may not be well-represented in real-world data. Synthetic datasets can be scaled easily, providing ample data for training and testing without the constraints of data availability or privacy concerns. They are also instrumental in creating proxy benchmarks that can simulate big data and AI workloads efficiently, reducing execution time while maintaining performance accuracy.

On the other hand, [real-world data] provides authentic challenges that are critical for evaluating a model's practical applicability. Datasets derived from real financial indicators, for instance, have been used to train models for bankruptcy prediction, offering insights into how symbolic learning methods perform in genuine economic contexts. Similarly, real-world vibration data and modal properties are utilized in structural damage assessment, testing models on data that reflects true environmental and operational conditions.

One notable example of real-world dataset is [CLEVRER][[6]], which focuses on temporal and causal reasoning within visual

scenarios. This dataset presents models with videos containing simple objects and events, accompanied by questions that require descriptive, explanatory, predictive, and counterfactual reasoning. By challenging models to go beyond mere pattern recognition, CLEVRER assesses their ability to comprehend causal structures and make inferences about future or hypothetical events.

[Geometry3K][[7]], consisting of over 3,000 geometry problems annotated in a formal language, evaluates a model's capacity for abstract reasoning and symbolic manipulation within mathematics. Models are tested on their ability to understand problem statements, apply axiomatic knowledge, and derive solutions symbolically, mirroring the cognitive processes involved in human mathematical reasoning.

In computer vision, datasets like Office-31[[8]] and VisDA-2017[[9]] are widely used for evaluating domain adaptation methods. These datasets contain images from different domains, such as office environments and synthetic objects, enabling the assessment of a model's ability to transfer learned knowledge between varying visual contexts. By testing models on their performance across these domains, researchers can gauge the effectiveness of their symbolic learning methods in adapting to new, unseen data.

[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#References-553}References[]{#Neuro-Symbolic-AI-EPUB-8.xhtml#x.162064}

[1] Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., & Liang, P. (2020, November). Concept bottleneck models. In International conference on machine learning (pp. 5338-5348). PMLR.

[2] Marc Roig Vilamala, Tianwei Xing, Harrison Taylor, Luis Garcia, Mani Srivastava, Lance Kaplan, Alun Preece, Angelika Kimmig, and Federico Cerutti (2023). DeepProbCEP: A neuro-symbolic approach for complex event processing in adversarial settings. Expert Systems With Applications, 215, Article 119376.

https://doi.org/10.1016/j.eswa.2022.119376

[3] Su, L., Pan, J., Wang, X., Xiao, X., Quan, S., Chen, X., & Jiang, J. (2024). STEM: Unleashing the Power of Embeddings for Multi-Task Recommendation. Proceedings of the AAAI Conference on Artificial Intelligence, 38(8), 9002-9010. https://doi.org/10.1609/aaai.v38i8.28749

[4] Lyu, D., Gustafson, S., Yang, F., & Liu, B. (2019). SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 2970—2977. https://doi.org/10.1609/aaai.v33i01.33012970

[5] Chen, X., Liang, C., Yu, A. W., Song, D., & Zhou, D. (2020). Compositional Generalization via Neural-Symbolic Stack Machines. In Advances in Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

[6] Yi, Kexin, et al. "Clevrer: Collision events for video representation and reasoning." arXiv preprint arXiv:1910.01442 (2019).

[7] Lu, Pan, et al. "Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning." arXiv preprint arXiv:2105.04165 (2021).

[8] Saenko, Kate, et al. "Adapting visual category models to new domains." Computer Vision—ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11. Springer Berlin Heidelberg, 2010.

[9] Peng, Xingchao, et al. "Visda: The visual domain adaptation challenge." arXiv preprint arXiv:1710.06924 (2017).

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml}

Basic-Text-Frame []{#Neuro-Symbolic-AI-EPUB-9.xhtml#Chapter-6-}Chapter 6:[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162067}

Neuro-Symbolic Reasoning Models

Figure 26. 162788.png

A neuro-symbolic reasoning model is a whole AI model that uses both neural network parts and symbolic reasoning parts to solve complex problems through a unified model. These models do both the "seeing patterns" tasks, and the "following logical rules" tasks. Such models can further improve with training data and domain knowledge. It performs reasoning steps that mimic human-like logical reasoning that require symbolic logic, within a neural or hybrid framework.

These models apply integrated neural-symbolic representations to solve problems like theorem proving, question answering etc. over knowledge bases. For example, a neuro-symbolic reasoning model could read a story, using a neural network, and then use logical rules to answer a question about the story's plot using symbolic reasoning.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.1-Differentiable-Reasoning}6.1 Differ[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162070}entiable Reasoning

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.1.1-Neural-Networks-for-Logical-Inference}6.1.1 Neur[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162073}al Networks for Logical Inference

Traditional logic gates like AND, OR, and NOT work with binary values (0 or 1). In contrast, neural networks operate on continuous values (numbers that can take on any value in a range). This mismatch makes it difficult to directly integrate standard logic operations into neural network computations.

To overcome this, researchers have developed [continuous, differentiable activation functions] that mimic the behavior of logical operations. These functions allow neural networks to perform logical inference, drawing conclusions based on logical relationships, while remaining fully trainable through gradient-based methods.

[Squashing Functions for Logical Operations]{.Bold}

[Squashing functions] are a specific type of continuous activation function designed to replicate logical operations within a neural network.

Unlike typical activation functions that simply add non-linearity, squashing functions enable the network to execute logical reasoning directly. For example, imagine a neural network that classifies images based on multiple criteria. By using differentiable logic gates combined with squashing functions, the network can explicitly model relationships between features. It can learn rules such as, "an object is a vehicle if it has wheels AND an engine."

This explicit modeling leads to more transparent decision-making because you can see how the network is combining features to reach its conclusions. In short, squashing functions improve both the reasoning capability and the explainability of the network.

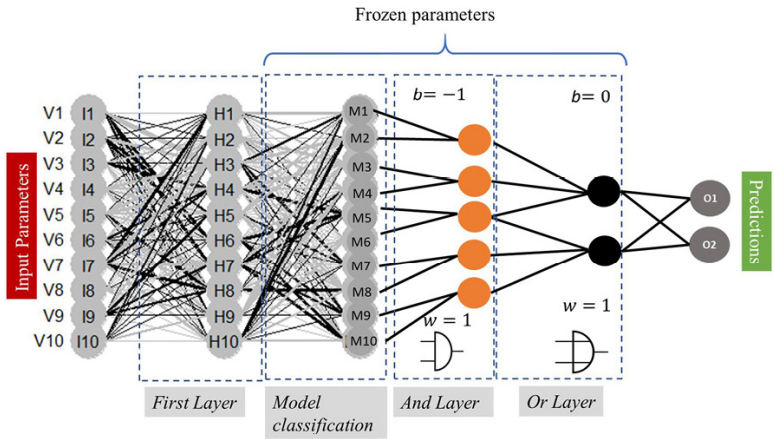


Figure 27. Fig_08.jpg

Fig 6.1.1

Figure 6.1.1 is a simplified diagram of the logical operator layer in

the neural network model[[1]]. V stands for the raw input features, H represents the trainable weights in the first hidden layer, l is the linear combination produced by this layer. M denotes a set of nodes or neurons that are designed to perform logical operations. Unlike the first layer, these nodes are not trained. Their weights and biases are fixed to implement logical functions (like AND and OR) using a special squashing function.

It shows us how the intermediate nodes (Mi) are first processed by two AND layers and then by an OR layer. In each of these layers, squashing functions are used as the activation functions to smoothly but sharply enforce the logic of AND and OR operations. They provide the "soft" thresholding needed to emulate digital logic while still allowing the model to be trained efficiently.

[Soft Unification and Variable Binding]{.Bold}

In logical expressions, variable binding is the process of assigning specific values to placeholders (or variables). Traditional logic requires exact matches. Either two items are exactly the same, or they are not.

However, neural networks work with probabilities and degrees of similarity, not strict binary matches. [Soft unification] relaxes the strict rules of classical logic by allowing for partial or probabilistic matches. This means that instead of requiring an exact match, the network can consider items that are similar to a certain degree.

For example, in natural language processing, understanding a sentence like "Alice gave Bob a book" involves recognizing relationships between entities (e.g., that a book was transferred from Alice to Bob). Soft unification enables the network to bind these variables (like "Alice", "Bob", and "book") even when the matches are not perfect, thereby improving tasks such as question answering and information extraction.

[Logical Neural Networks (LNNs)]{.Bold}

[Logical Neural Networks (LNNs)] are a method that combines

classical Boolean logic with neural network architectures. These networks incorporate trainable parameters but maintain logical structures. This allows them to learn interpretable first-order logic rules using standard training techniques like backpropagation.

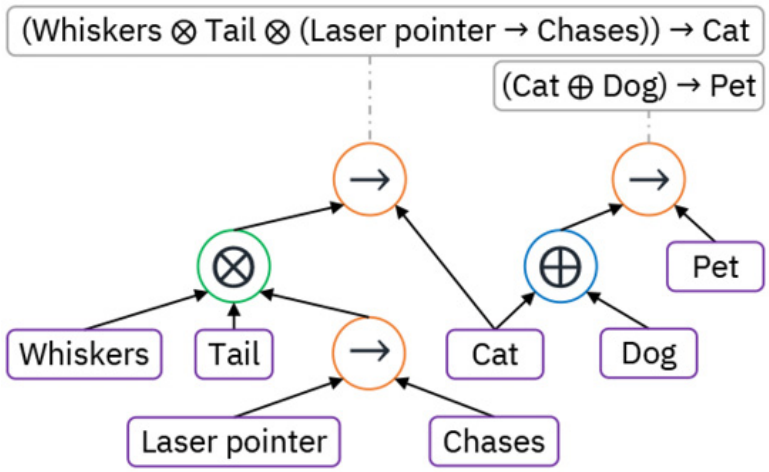


Figure 28. Fig_09.jpg

Fig 6.1.2

Figure 6.1.2 illustrates how a logical formula is translated into a neural network[[2]]. Each node in the diagram represents a part of a logical expression. The overall structure of the network mirrors the "syntax tree" of the logical formula. In other words, the way the neurons are connected reflects the way the logical expression is built up from smaller pieces.

Bounds	Unknown	True	False	Contradiction
Upper	$[\alpha, 1]$	$[\alpha, 1]$	$[0, 1-\alpha]$	Lower > Upper
Lower	$[0, 1-\alpha]$	$[\alpha, 1]$	$[0, 1-\alpha]$	

Figure 29. Fig_10.jpg

Fig 6.1.3

Rather than using a simple binary (yes/no) answer, LNNs work with degrees of truth. Figure 6.1.3 illustrates how each neuron in a logical neural network doesn't just give a single "true" or "false" answer. Instead, every neuron outputs two numbers that serve as a lower and an upper bound for the truth value. These bounds determine whether a logical statement is definitely true, definitely false, or somewhere in between (ambiguous or uncertain).

[Logic-Integrated Neural Network (LINN)] is another related architecture, which can learn basic logical operations as separate neural modules. They are structured to perform propositional logical reasoning, making them effective for tasks that rely on understanding and applying logical rules, such as recommendation systems or decision-making processes.

[Weakly Supervised Neuro-Symbolic Learning (WS-NeSyL)]{.Bold}

[Weakly Supervised Neural Symbolic Learning (WS-NeSyL)] is a framework designed to learn logical rules without needing extensive labeled data. It uses a "back-search" algorithm, which is similar to navigating a maze. You make a series of choices, and if you hit a dead end, you backtrack to the last decision point and try a different path. This process explores various reasoning paths and helps the neural network learn probabilistic representations of logic.

For instance, in healthcare analytics, it can learn logical rules from patient data, such as symptoms and test results to help diagnose diseases. By combining data-driven learning with logical reason-

ing, the system can provide diagnoses that are not only accurate but also explainable, leading to better patient outcomes.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.1.2-Continuous-Relaxation-of-Logical-Operators}6.1.2 Continu[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162076}ous Relaxation of Logical Operators

Traditional logic gates (AND, OR, NOT) work with strict 0s and 1s. However, neural networks work with numbers that can be any value within a range (continuous values). This makes it hard to directly apply classical logic in a neural network. Researchers modify these strict logical operations so they can work with continuous values. This process is called continuous relaxation, and it lets neural networks combine symbolic (rule-based) reasoning with learning from data.

[Fuzzy Logic and T-Norms]{.Bold}

[Fuzzy logic] allows truth values to be any number between "completely true" (1) and "completely false" (0), instead of just 0 or 1. This system can express partial truth.

[T-Norms] are mathematical operations that generalize the logical AND to work with continuous values. In fuzzy logic, they serve as "synaptic operators" in fuzzy neurons, determining how input values combine before the neuron activates. T-Conorms, also known as s-norms work like the OR operator, shaping the final output of the neuron (they act as "somatic operators").

Imagine a neuron evaluating the statement "The weather is warm AND humid." In traditional logic, you would say "warm" and "humid" are either true or false. In fuzzy logic, they might be partially true, say, warm is 0.7 and humid is 0.6. A t-norm (for example, multiplication) would combine these values: $0.7 \times 0.6 = 0.42$, giving an overall measure of truth for the statement.

[Probabilistic Soft Logic (PSL)]{.Bold}

[Probabilistic Soft Logic (PSL)] comes into play in real-world

scenarios, where often uncertainty is involved. In many real-world situations, uncertainty is a major factor. Instead of just handling partial truths, it's important to account for uncertainty in a probabilistic way. Probabilistic Soft Logic (PSL) blends soft logic with probabilistic reasoning.

It represents logical relationships using a mathematical model called hinge-loss Markov random fields (HL-MRFs). This model converts first-order logic into a probabilistic framework, making it efficient and scalable, even when dealing with millions or billions of variables and rules.

For example, in natural language processing tasks like Semantic Textual Similarity (STS), PSL can capture the subtle relationships between words and phrases, reflecting both the logical structure of language and the inherent uncertainty in meaning.

[DeepPSL and Bayesian Optimization]{.Bold}

[DeepPSL] integrates deep learning techniques with probabilistic soft logic. This creates an end-to-end trainable system that benefits from neural networks' ability to learn rich features from data, while also performing logical reasoning through PSL.

Methods like Bayesian optimization for weight learning (BOWL) are used to fine-tune the parameters of these models based on performance in the specific application, ensuring they are both theoretically robust and practically effective.

[Embedding Logical Structures in Vector Space]{.Bold}

Another approach to making logic work in neural networks is by embedding logical expressions, entities, and relationships into continuous vector spaces. Here, everything is represented as a vector (a list of numbers), and logical operations can be performed on these vectors in a differentiable way. In this method, logical rules are used as constraints during the training of these embeddings.

For example, if there's a rule like "All cats are mammals," the model learns vector representations so that the vector for "cat" is close

to the vector for "mammal." The relation "is a" is modeled to capture this logical connection. The network can then use these embeddings to perform logical inferences, such as concluding that "If Felix is a cat, then Felix is a mammal."

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.1.3-Applications-of-Differentiable-Reasoning}6.1.3 Applica[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162081}tions of Differentiable Reasoning

[ReasoNets] use a special controller along with a shared memory. Instead of following fixed, pre-set rules for reasoning, they learn how to take reasoning steps directly from the data. This means they figure out the best way to reason on their own. Imagine an AI system analyzing a social network. It knows "Alice is a friend of Bob" and "Bob works with Carol," but it doesn't have a direct link saying "Alice is connected to Carol." Using differentiable reasoning, the system can infer the missing connection by understanding the underlying relationships, effectively filling in the gap in the knowledge base.

[Natural Language Inference (NLI)]{.Bold}

[Natural language inference (NLI)] is about determining the logical relationship between sentences. For example, it checks whether one sentence supports (entails), contradicts, or is neutral with respect to another sentence. This technology is used in various applications, from virtual assistants to translation services. Traditional neural networks often struggle with logical reasoning tasks.

By combining differentiable reasoning with neural logic models, these systems can process both the meaning of words (semantic features) and the logical relationships between them at the same time.

Consider the statements "All birds can fly" and "Penguins cannot fly." A differentiable reasoning model understands that while the general rule is that birds can fly, there are exceptions. Penguins, for instance, cannot fly. This ability to manage exceptions helps the model resolve apparent contradictions.

[Commonsense Reasoning]{.Bold}

[Commonsense reasoning] is the ability to make assumptions about everyday situations, something humans do naturally. For example, knowing that if someone is carrying an umbrella, it might be raining.

By integrating large amounts of world knowledge with logical inference, differentiable reasoning equips neural networks to draw logical conclusions even in unclear or ambiguous situations. When large language models are trained using these techniques, they show a remarkable ability to understand and reason about everyday situations, much like humans.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.2-Neural-Theorem-Provers}6.2 Neural Th[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162084}eorem Provers

Neural theorem provers are systems that blend symbolic logic with neural networks with an aim to perform logical reasoning. It simulates the process of logical reasoning in a way that "smooths out" the hard, yes-or-no decisions of classical logic and makes them compatible with modern machine learning techniques. This allows the system to learn from data and adjust its internal representations accordingly. This process can also be used for proving theorems.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.2.1-End-to-End-Training-for-Theorem-Proving}6.2.1 End-to-[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162087}End Training for Theorem Proving

Traditional theorem proving methods rely on working with exact symbols and rules, otherwise known as symbolic manipulation, and discrete logic, which are strict true/false values. This approach follows fixed logical rules to prove statements.

Neural networks, on the other hand, use gradient-based optimization to learn from data. They work with continuous values (numbers that can vary smoothly) rather than strict binary values. To combine these two worlds, researchers developed "differentiable

prover architectures." This means they designed systems that can perform logical reasoning in a way that fits within the continuous, trainable framework of neural networks.

[Neural Theorem Provers (NTPs)]{.Bold}

[Neural Theorem Provers (NTPs)] perform first-order logical inference by drawing conclusions based on rules about objects and their relationships, using continuous vector representations. They embed symbols like predicates, constants, and rules into a shared vector space.

This means every element is represented as a vector, i.e. a list of numbers, which the network can work with. They use a version of the backward chaining algorithm, a method that starts with a goal and works backward to find supporting facts but in a differentiable way. This makes it possible to use gradient-based training methods such as backpropagation.

[Greedy Neural Theorem Provers (GNTPs)]{.Bold}

[Greedy Neural Theorem Provers (GNTPs)] are needed when dealing with large datasets, as the traditional approach can become computationally expensive. GNTPs address this by dynamically constructing computation graphs and focusing on the most promising paths to build a proof. This "greedy" strategy helps manage computational complexity and makes the system scalable.

[Forward and Backward Chaining in Practice]{.Bold}

Let's understand [Forward chaining] and [backward chaining] for AI systems. Consider an AI-powered diagnostic system in healthcare. With forward chaining, the system starts with observed symptoms (facts) and applies medical knowledge (rules) to infer possible diseases (conclusions). Conversely, with backward chaining, the system could begin with a suspected diagnosis and work backward to check if the symptoms support it. Modeling these strategies within neural frameworks allows for efficient and accurate diagnostic reasoning.

[Learning Proof Strategies from Data]{.Bold}

Neural networks can learn proof strategies directly from data. By training on large datasets containing proofs neural networks can be taught proof strategies. By doing so, they learn patterns and strategies (heuristics) that lead to successful proofs. In automated theorem proving, neural networks can learn which lemmas (small proven statements) or axioms (basic truths) are most useful in certain contexts.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.2.2-Scaling-Theorem-Proving-with-Neural-Networks}6.2.2 Scaling[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162090} Theorem Proving with Neural Networks

In huge knowledge bases, not every piece of information is equally important. Some axioms (basic rules or facts) are more useful for reasoning than others. Automatic Extraction of Axioms streamlines the reasoning process by identifying and focusing on the most relevant information in large knowledge bases. Researchers analyze the structure of a knowledge base to identify and extract these high-impact axioms.

This process "prunes" or cuts away less relevant parts of the search space. By focusing on the densely connected areas, where important predicates and relationships reside, neural theorem provers can speed up processing significantly while still covering most of the necessary information.

[Integrating Multiple Knowledge Bases]{.Bold}

[Aligning and integrating multiple knowledge bases] is another strategy. As the name suggests, Instead of relying on one massive dataset, multiple, smaller knowledge bases are combined. Techniques like partition-and-blocking are used to divide data into manageable parts, and overlapping entities (common pieces of information) are identified across these bases.

This integration allows the theorem prover to use a richer and

broader set of information. For example, in natural language understanding, linking a lexical database like WordNet with an encyclopedic source like DBpedia provides better semantic context. The result is more nuanced reasoning, as the system can draw on diverse sources when constructing proofs or inferring relationships.

[Parallelizing Proof Search]{.Bold}

Instead of testing one reasoning path at a time (sequentially), the system can explore multiple paths simultaneously. This is known as parallel processing. Multiple proof strategies or heuristics are run at the same time across different processors or machines.

One processor might handle algebraic manipulations while another focuses on geometric interpretations. Their results are later combined to form a complete proof. This approach significantly reduces the time needed to find a valid proof and increases the chances of discovering innovative or non-obvious solutions.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.3-Probabilistic-Logic-Networks}6.3 Probabili[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162093}stic Logic Networks

[Probabilistic logic networks] are frameworks that combine traditional logic with ability to handle uncertainty, using probabilities. Instead of treating statements as strictly true or false, PLNs attach probabilities to facts and rules to reflect how likely they are to be true.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.3.1-Combining-Probability-with-Logic-in-Neural-Models}6.3.1 Combini[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162096}ng Probability with Logic in Neural Models

Integration of probability and logic empowers AI systems to handle uncertainty while reasoning with the precision of logical inference. One of the approaches is the concept of [Markov Logic Networks (MLNs)]. In an MLN, each logical formula is associated with a weight, reflecting the strength or confidence in that particular piece

of knowledge. The network captures both the logical structure and the probabilistic dependencies among variables. However, this is not scalable for large datasets. MLNs, when integrated with neural networks, particularly Graph Neural Networks, can manage the complexity more efficiently. For example, the [ExpressGNN] model employs GNNs to perform variational inference within MLNs.

[Bayesian Logic Programs (BLPs)], blend Bayesian networks, which deal with uncertainty by assigning probabilities, with logical clauses from first-order logic, which provide clear logical structure. They separate the logical relationships from the uncertainties. For example, a medical diagnosis system might use logic to represent a rule like, "If a patient has a sore throat and a fever, they might have strep throat." Then, the probabilistic part assigns the likelihood of each disease, with confidence intervals, to help clinicians decide on treatment.

In NLP, these models help understand ambiguous sentences. They consider both the grammatical structure (the logic) and the chance that a certain interpretation is correct (the probability). This dual approach leads to better comprehension and more accurate language models.

In environments where sensor data can be noisy or incomplete, combining probabilistic logic with neural networks allows self-driving cars to make safer decisions by accounting for uncertainties in the surroundings.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.3.2-Learning-in-Probabilistic-Logic-Networks}6.3.2 Learnin[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162099}g in Probabilistic Logic Networks

In a probabilistic logic network, every rule or piece of evidence gets a weight or parameter. These numbers help the network decide how much trust to put in a strict logical rule versus uncertain, real-world data. In other words, they guide the network's overall reasoning process.

[Supervised Learning Probabilistic Neural Networks (SLPNNs)] adjust various factors, such as synaptic weights, which are connections between neurons, and kernel radii, which influence how data is grouped, and data weights, based on supervised learning. This fine-tuning allows the network to improve its performance over time. [Deep Probabilistic Logic (DPL)] on the other hand combines deep learning with probabilistic logic. It uses a technique called variational Expectation-Maximization (EM), which works in two alternating steps. The first step involves refining the deep neural network based on the current data and the second step involves Updating the weights of the logical rules based on what the network learned. This approach lets the network learn directly from data even if not every variable has explicit labels.

[Structure Learning]{.Bold}

[Structure learning] figures out how different variables (or pieces of data) in a network are connected and how they influence each other. For Bayesian networks (which represent probabilistic relationships), structure learning searches through different graph configurations to find the best one that explains the observed data.

Some of the methods are [score-based methods], which evaluate how well each possible structure fits the data, and [constraint-based methods] which use statistical tests to determine which variables are dependent or independent. [Probabilistic relational models] extend Bayesian networks to handle structured and relational data. They use structure learning to manage the huge number of possible connections in relational domains.

[Inference and Scalability]{.Bold}

When a logic network becomes very complex, performing exact calculations for every inference is too slow. Instead, [approximate inference methods], like variational inference, simplify complex probability distributions into easier-to-compute ones. Variational inference, for example, models nonlinear relationships between variables while still allowing probabilistic reasoning. [Conditional

Random Fields (CRFs)] on the other hand, breaks down the problem into smaller, independent parts, which can then be computed in parallel.

[DeepProbLog] framework integrates probabilistic logic programming with neural networks. It uses specialized search algorithms (like DPLA*) to perform approximate inference much faster, sometimes orders of magnitude quicker, without sacrificing accuracy. This speedup makes it practical to use sophisticated probabilistic models even in settings with limited computational resources.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.4-Advanced-Neuro-Symbolic-Reasoning-Models}6.4 Advanced []{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162102}Neuro-Symbolic Reasoning Models

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.4.1-Graph-Neural-Networks-for-Logical-Reasoning}6.4.1 Graph N[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162105}eural Networks for Logical Reasoning

Logical reasoning is about understanding how different things connect and interact. Traditional neural networks often struggle at it, because they treat each piece of information separately, without naturally focusing on the relationships between them.

[Graph Neural Networks (GNNs)] are designed to work with data that looks like a network, where items (or nodes) are linked by relationships (or edges). This makes GNNs well-suited for tasks that require understanding these connections.

GNNs work by having each node share information with its connected neighbors, as this information is passed around, nodes gradually build up a picture of both their immediate surroundings and the larger structure of the network. In practical terms, this means that in a social network, GNNs can look at friendships and shared interests to figure out community groups.

Many GNNs also use an attention mechanism that helps the model focus on the most important connections in the network.

Knowledge graphs are structured maps where pieces of information are connected by relationships, and because these maps are often incomplete, GNNs can be used to predict and fill in the missing links by learning patterns from the available data.

Advanced GNN architectures, such as relational graph attention networks, improve on this by assigning different weights to various connections, allowing the model to prioritize more influential relationships for more accurate predictions.

When it comes to answering complex questions, GNNs can convert both the query and the related graph data into a format they can work with. Logical reasoning is then simulated by propagating information through the network to piece together an answer.

Some models even integrate techniques like fuzzy logic or sequential query encoding to handle uncertainty and simplify complex queries, ensuring that the system can deal with ambiguous real-world data while still providing accurate and reliable results.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.4.2-Neuro-Symbolic-Dynamic-Reasoning}6.4.2 Neuro-S[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162108}ymbolic Dynamic Reasoning

In everyday life, environments are dynamic, data flows continuously, and circumstances can shift in an instant. Traditional AI systems, which rely on fixed datasets and predefined rules, often struggle when they encounter new, unexpected situations.

[Dynamic reasoning] is the capability of an AI system to understand, adjust, and react to these changes as they happen on the fly. Without dynamic reasoning, AI is limited to what it was originally programmed or trained to do, and it can't easily deal with new information or scenarios.

Imagine a self-driving car navigating a busy city. It needs to process a constant stream of information, from traffic signals and pedestrian movements to weather changes, and make decisions in a split second. A static AI model, using only fixed rules, wouldn't

be able to cope with such variability, whereas dynamic reasoning enables the car to continually update its strategy for safe and efficient driving.

Dynamic knowledge bases are continuously updated by automatically extracting data from multiple sources like web content, sensor inputs, and real-time user interactions. The system then uses probabilistic reasoning to assess the reliability of this incoming data, allowing it to make well-informed decisions even when some information is incomplete or uncertain.

[Differentiable Neural Computers (DNCs)] merge neural networks with external memory resources similar to a computer's RAM, enabling them to store and manipulate complex data structures. This hybrid approach dramatically boosts the AI's capacity for advanced reasoning and problem-solving, making it better equipped to handle intricate, real-world challenges.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.4.3-Neuro-Symbolic-Models-for-Temporal-Reasoning}6.4.3 Neuro-S[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162113}ymbolic Models for Temporal Reasoning

[Temporal data] is simply data that changes over time. Think of it as any information where the order or timing of events matters. Recognizing patterns in how things change can improve predictions in many fields, from forecasting weather to managing traffic.

[Temporal logic] provides rules to say things like "event A happens before event B." This helps make sense of the sequence and timing of events. Some neural networks are designed to directly handle time. They include features like [time delays] and [recurrent structure]. Time delay allows the network to learn from events that occur at different intervals. Recurrent structures help in remembering previous events to predict future ones. These networks can automatically figure out the best way to capture the timing and sequence of events without needing a preset "time window."

[Adaptive Neuro-Fuzzy Inference System (ANFIS)] is a popular

method, which combines neural networks and fuzzy logic. Neural network helps in learning complex patterns, whereas fuzzy logic helps in handling uncertainty and imprecision, such as ambiguous human behavior or market sentiment. This blend makes predictions more robust, even in noisy environments like the stock market or weather forecasting.

Figure 6.4.3[[3]] presents the overall layout of an ANFIS system as a five-layer neural network that implements a fuzzy inference system. It converts numerical inputs into fuzzy values, applies fuzzy if-then rules, with both fuzzy matching and linear output functions, and finally defuzzifies the result to produce a precise output. This layered structure allows the system to learn and fine-tune both the fuzzy membership functions, and the linear rules (consequent parameters) to model complex nonlinear relationships effectively.

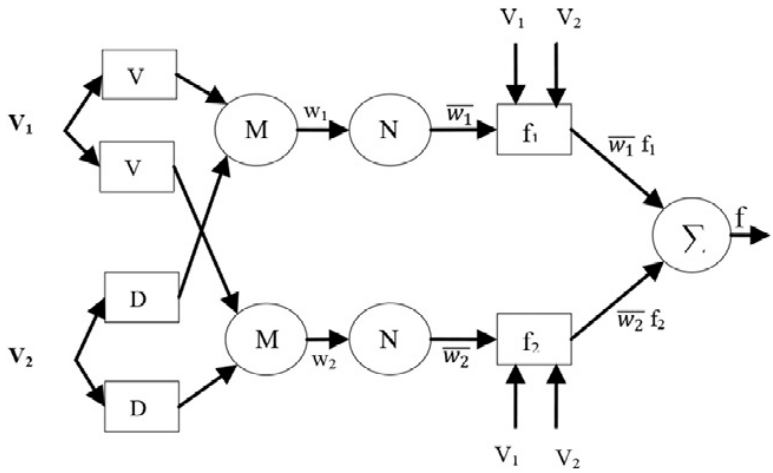


Figure 30. Fig_11.jpg

Fig 6.4.3

The first layer, shown as a square, is the fuzzy layer. Each node here

is adaptive and calculates the membership grade for each input. In other words, it transforms exact numbers into fuzzy values, using membership functions. The “V” nodes compute the degree to which the input value belongs to a fuzzy set. The “D” nodes do the same for another input dimension

The second layer is the product layer. Here, each node multiplies the membership values coming from the first layer. This multiplication acts as a fuzzy “AND” operation, giving a rule’s firing strength. It tells us how strongly the inputs match the fuzzy conditions. Each node, labeled as M multiplies the membership grade from the V side with that from the D side.

The third layer is the normalization layer. This layer takes the firing strengths from the second layer and normalizes them by dividing each firing strength by the sum of all firing strengths. This step ensures that all the rule activations add up to one, making them comparable. Each node, labeled as N divides its firing strength by the sum of all firing strengths from Layer 2, yielding a normalized weight. So, if w_1 and w_2 are the raw firing strengths, the normalized values become $w_1/(w_1+w_2)$ and $w_2/(w_1+w_2)$.

The fourth layer is the de-fuzzy layer. Each node here multiplies the normalized firing strength by a first-order polynomial. These polynomials represent the rules’ outputs, or consequences, in a Sugeno fuzzy model. For Rule 1, the output is $f_1 = p_1 \cdot v + q_1 \cdot d + r_1$, and for Rule 2, it is $f_2 = p_2 \cdot v + q_2 \cdot d + r_2$. Here, p_1, q_1, r_1 (and similarly p_2, q_2, r_2) are the consequent parameters that are tuned during training.

The fifth and final layer is the total output layer. This layer aggregates the outputs from the fourth layer, resulting in a crisp, final output for the system.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.4.4-Causal-Reasoning-with-Neuro-Symbolic-Models}6.4.4 Causal R[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162116}easoning with Neuro-Symbolic Models

Traditional neural networks are great at spotting patterns in data. However, they typically notice patterns when two events happen together (correlation) rather than understanding if one event actually causes the other (causation).

To address this, [causal reasoning] models blend neural networks with logical reasoning, allowing AI to not only learn from data but also understand why things happen. Instead of just finding patterns, these systems use clear rules to explain relationships between events. Imagine a system that diagnoses diseases.

Traditional models might simply link high blood sugar to diabetes because they often occur together. A causal reasoning model, however, understands that high blood sugar actually leads to symptoms like increased thirst and frequent urination, making the diagnosis more accurate and explainable.

[Counterfactual reasoning] can explore what might happen if certain parts of a situation were different through "what if" questions. For example, a self driving car might simulate different routes to explore "What if I turn left here instead of going straight?" in order to decide the safest path. Counterfactual reasoning is most helpful for scenarios where limited data is available. It does so by imagining and testing different scenarios, making it more adaptable in dynamic situations.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.5-Optimization-and-Training-Techniques-and-Datasets}6.5 Optimizati[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162119}on and Training Techniques and Datasets

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.5.1-Gradient-Based-Optimization-in-Logical-Models}6.5.1 Gradient[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162122}-Based Optimization in Logical Models

Neural networks learn through a process known as gradient-based optimization, which relies on adjusting weights using gradients. This process relies on computing derivatives via backpropagation.

However, logical operations like AND, OR, and NOT aren't smooth and don't change gradually. This [non-smoothness] makes it hard to compute gradients directly through these operations.

Logical operations can be [smoothened] through [continuous relaxation]. This is introduced by approximating hard logical functions into smooth, differentiable functions, so that gradients can be calculated. For example, the AND operation can be approximated by multiplying values, only if all inputs are close to 1 does the product approach 1. Similarly, the OR operation can be mimicked by using a smooth maximum or a probabilistic sum.

Sometimes, even with approximations, some components remain non-differentiable. There are multiple strategies to mitigate this. [Surrogate functions] replace non-differentiable parts. Similar to the L1 norm used for sparsity, with smooth versions that behave similarly, so gradients can still be used. [Relaxation techniques] convert binary decisions into continuous values between 0 and 1, allowing the model to compute gradients even for problems originally defined with discrete choices. There are [alternative gradient methods], such as forward gradient method which computes directional derivatives without needing a full backward pass, which can simplify computations. Similarly, subgradient methods extend the idea of gradients to functions that aren't smooth by using generalized gradients.

As neuro-symbolic models get more complex, they might learn the training data too well and fail on new data. Regularization techniques can help mitigate this situation. For example, [Dynamic regularization] adjusts the strength of regularization based on performance. It increases if the model overfits and decreases it if the model is too simple. [Spectral dropout] works in the frequency domain, using Fourier transforms to remove weak, noisy components from the model, which can speed up training and reduce overfitting. [L1 regularization] encourages sparsity, with fewer nonzero weights in the model, which not only helps prevent overfitting but also makes the model easier to understand. [Logical

components] can be regularized by applying regularization to the symbolic parts, to keep the reasoning process simple and to avoid overfitting to irrelevant patterns.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.5.2-Reinforcement-Learning-for-Reasoning}6.5.2 Reinforc[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162125}ement Learning for Reasoning

Traditional AI systems often follow [fixed, pre-programmed reasoning steps]. These steps are usually designed by experts or extracted from static data, which can make the system rigid and hard to update.

[Reinforcement learning (RL)] lets an AI learn the best steps to reach a conclusion by trial and error, instead of following a fixed route. The AI interacts with its environment, experimenting with different reasoning paths and learning which ones lead to correct answers. By learning the best paths through complex networks, the AI can process information faster and more accurately. Also, since the learned reasoning paths can be inspected, it's easier for humans to understand how the AI reached its conclusions. This is important in fields like healthcare and finance.

Logical consistency can be ensured by [designing reward functions]. The AI receives rewards for making correct decisions. The reward system must promote not just any solution, but ones that make sense logically. For example, in language tasks, the model might earn rewards for correctly identifying logical mistakes, reinforcing valid reasoning.

Logical consistency can also be achieved through [case-based reasoning] by making the AI learn from previous examples. Current reasoning path is compared to successful past cases. This helps the model get feedback, which in turn helps the model improve its logic over time.

Consider a system that recommends books. Using RL, the AI can explore a network of books, authors, genres, and reader preferences to discover patterns. It might learn, for example, that readers who

enjoy classic science fiction also tend to like modern speculative fiction, and then make recommendations based on these discovered pathways.

Combining RL with [graph neural networks (GNNs)] helps the AI better understand the structure of the knowledge graph. This fusion allows the model to more effectively narrow down which connections are most important. It can boost performance in tasks like question answering and information retrieval.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.5.3-Transfer-Learning-and-Meta-Learning}6.5.3 Transfer[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162128} Learning and Meta-Learning

[Transfer Learning] is about reusing knowledge from one task or domain to help solve a related problem in another domain. Imagine you've learned how to fix one type of machine. You can use that experience to understand and repair a different, but similar, machine.

A model trained to recognize parts in one machine can adapt to recognize parts in another machine with only a little extra training. This saves time and data compared to starting from scratch.

In a neuro-symbolic system, these pre-trained neural modules can be used for extracting general features for initial embedding. These features can be mapped to symbolic representations that capture high-level concepts. Both components adapt to new tasks, allowing the system to generalize its reasoning skills with minimal extra training data.

[Meta-Learning] helps models quickly adapt to new tasks by training them on multiple tasks so they pick up common strategies. Instead of training a model for just one task, you train it on many different tasks so it learns a general strategy for learning.

When a new task or new user comes along, like recommending books for someone with very little history, the model can quickly adjust using the "learning strategy" it developed from previous

tasks.

Meta-learning in neuro-symbolic AI lets the system develop a flexible, general-purpose reasoning ability. This enables it to quickly adapt to new tasks. It can be for example understanding a novel language pattern, solving a unique problem, or making sense of a new type of data. It does so by leveraging the common reasoning strategies it has learned from previous experiences.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.6-Evaluation-and-Benchmarking-of-Reasoning-Models}6.6 Evaluation[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162131} and Benchmarking of Reasoning Models

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.6.1-Standard-Datasets-and-Benchmarks}6.6.1 Standard[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162134} Datasets and Benchmarks

AI models are pushed to prove their logical reasoning abilities through various benchmarks and datasets.

[bAbI][[4]] and [logic puzzles] are structured datasets that test basic inference and relationship tracking. bAbI is a collection of 20 tasks designed to test different types of logical reasoning like deduction, induction, and tracking relationships in a story. It challenges the model to understand sequences and relationships, not just recognize patterns.

For example, a short story might say, "Mary went to the kitchen. John went to the garden. Mary picked up the apple. Where is the apple?" The AI must follow the story steps to answer correctly. Logic puzzles has puzzles such as Sudoku or grid puzzles require the model to follow strict logical rules to solve them.

[LogicInference][[5]] dataset presents problems in both formal notation and natural language, focusing purely on logical reasoning without relying on statistical language patterns. Datasets like [FOLIO][[6]] mix natural language with formal logic, challenging AI to handle both language understanding and strict logical deduction.

Theorem Proving Benchmarks evaluate the ability to generate step-by-step proofs and handle complex mathematical logic. It forces the AI to demonstrate deep understanding and the ability to construct logical arguments rather than simply matching patterns.

[INT][[7]] dataset focuses on proving mathematical inequalities, challenging the model to apply mathematical concepts correctly. [LeanDojo][[8]] contains thousands of theorems and proofs, emphasizing the selection of the right premises and axioms, much like how mathematicians work.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x6.6.2-Evaluation-Metrics-and-Comparative-Analysis}6.6.2 Evaluati[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162137}on Metrics and Comparative Analysis

Evaluating neuro-symbolic reasoning models entails checking how accurately they predict or classify data, and how well they follow logical rules. For tasks like classification or prediction, common metrics include precision, recall, F1-score, mean squared error, and mean absolute error. These metrics tell us how close the model's outputs are to the correct answers. For tasks like speech recognition, models are tested using measures such as accuracy, Word Error Rate (WER), and how well they work on new, unseen data.

Standard metrics don't fully capture whether a model is reasoning logically. To address this, researchers use special measures, often called logical fidelity metrics[[9]] to check logical entailment of generated statements from semi-structured tables. These metrics assess the model's ability to maintain logical consistency while generating fluent and coherent text.

Multiple fidelity metrics that assess different aspects of logical reasoning, such as attribute fidelity, bivariate fidelity, population fidelity, and application fidelity[[10]]. By using a diverse set of metrics, researchers can gain a more holistic understanding of a model's performance in terms of logical consistency and reasoning capabilities.

[]{#Neuro-Symbolic-AI-EPUB-9.xhtml#References-577}References

[1][]{#Neuro-Symbolic-AI-EPUB-9.xhtml#x.162140} Ochoa, J.G.D., Csiszár, O. & Schimper, T. Medical recommender systems based on continuous-valued logic and multi-criteria decision operators, using interpretable neural networks. BMC Med Inform Decis Mak 21, 186 (2021). https://doi.org/10.1186/s12911-021-01553-3

[2] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. 2020. Logical Neural Networks. https://doi.org/10.48550/ARXIV.2006.13155

[3] Walia, N., Singh, H., & Sharma, A. (2015). ANFIS: Adaptive neuro-fuzzy inference system-a survey. International Journal of Computer Applications, 123(13).

[4] Weston, J., Bordes, A., Chopra, S., et al. (2016). "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks." Facebook AI Research.

[5] Ontañón, S., Ainslie, J., Cvicek, V., & Fisher, Z.K. (2022). LogicInference: A New Dataset for Teaching Logical Inference to seq2seq Models. ArXiv, abs/2203.15099.

[6] Han, Simeng, et al. FOLIO: Natural Language Reasoning with First-Order Logic, EMNLP 2024, https://aclanthology.org/2024.emnlp-main.1229/

[7] Yuhuai Wu, Albert Jiang, Jimmy Ba, Roger Baker Grosse, An Inequality Benchmark for Evaluating Generalization in Theorem Proving, 9th International Conference on Learning Representations, ICLR 2021

[8] Yang K, Swope A, Gu A, et al. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. In: Proceedings of the Neural Information Processing Systems Conference; 2023

[9] Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical Natural Language Generation from Open-Domain Tables. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7929—7942, Online. Association for Computational Linguistics.

[10] Dankar, F. K., Ibrahim, M. K., & Ismail, L. (2022). A Multi-Dimensional Evaluation of Synthetic Data Generators. IEEE Access, 10, 11147—11158. https://doi.org/10.1109/access.2022.3144765

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml}

Basic-Text-Frame []{#Neuro-Symbolic-AI-EPUB-10.xhtml#Chapter-7-}Chapter 7:

App[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162145}lications of Neuro-Symbolic AI

Figure 31. 162850.png

Neuro-symbolic AI combines the strengths of both neural networks and symbolic reasoning, offering a more balanced and intelligent approach to problem-solving. Neural networks are great at recognizing patterns, handling noisy data, and making sense of images or speech, but they struggle to explain their reasoning or handle new rules easily. Symbolic reasoning, on the other hand, is good at logical, rule-based thinking and can provide clear explanations, but it usually can't handle messy, real-world input as smoothly. By blending these two methods, neuro-symbolic AI can learn efficiently from examples, understand context, follow logical steps, and clearly explain its decisions, leading to smarter, more reliable, and more trustworthy AI systems.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.1-Natural-Language-Processing}7.1 Natural La[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162148}nguage Processing

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.1.1-Question-Answering-Systems}7.1.1 Question[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162151} Answering Systems

[Overview]{.Bold}

[Question answering systems] are computer programs designed to understand questions written in everyday language and then provide accurate answers. Traditionally, these systems had two major approaches, namely symbolic systems and neural systems. [Symbolic systems] use structured databases called knowledge graphs that store facts and the relationships between them. These systems are great at logical reasoning (for example, figuring out connections like "who invented something?") but can struggle with the subtlety of natural language. [Neural systems] use neural networks that learn from large amounts of unstructured text (like articles or conversations). They are excellent at understanding the patterns and nuances in language but aren't as good at performing clear logical steps or reasoning over structured facts.

[Example]{.Bold}

For example, When we ask, "Which scientist developed the theory of relativity?" the system uses neural networks to pick out key words like "scientist" and "theory of relativity.". It then consults the structured knowledge graph to find the relationship between the theory and its developer. Finally, it gives the answer: "Albert Einstein.".

[Relational Graph Convolutional Network (R-GCN)]{.Bold}

[Relational Graph Convolutional Network (R-GCN)] is often used for this purpose. R-GCNs are designed to work with data that has many different relationships, just like those found in knowledge graphs. They allow the system to perform multi-step (or multi-hop) reasoning for finding answers that aren't directly linked to the question by connecting several pieces of information together.

[Memory Networks and Dynamic Memory Network

(DMN)]{.Bold}

Many real-world questions require the system to pull answers from unstructured text like articles or reports, not just from pre-built databases. Here, [memory networks] come into play, especially a type known as Key-Value Memory Networks. It reads documents and stores important pieces of information as key-value pairs. Keys represent important words or concepts and values hold the surrounding context or details. When a question Is asked, the system uses the question to search its memory for the right keys. It then retrieves the corresponding values to piece together the answer.

For example, if given a document about climate change and asked, "What are the primary causes of global warming?" the network will look for keys like "causes" and "global warming." and retrieve information about factors such as greenhouse gas emissions and deforestation. This is very effective for handling long documents or complex questions because the system can keep track of various details and their relationships.

Adding symbolic reasoning to this approach further helps the system stay logically consistent and ensures that the information retrieved directly answers the question. Dynamic Memory Network (DMN), for example is a state-of-the-art neural network architecture that processes input sequences and questions, forms episodic memories, and generates relevant answers. It uses an iterative attention process to condition its focus on inputs and previous iterations.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.1.2-Language-Understanding-with-Logic-and-Deep-Learning}7.1.2 Language[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162154} Understanding with Logic and Deep Learning

One of the biggest challenges in natural language processing (NLP) is teaching computers to understand how human language works. Human languages follow specific grammatical rules and have a lay-

ered (or hierarchical) structure. Traditional neural network models are great at spotting patterns in data but often have difficulty with the deeper, nested structures of language, like how words form phrases and phrases form sentences. To overcome this, researchers have been trying to build these grammar rules directly into neural network models.

[Hierarchical Neural Models]{.Bold}

Words combine to form phrases and phrases come together to form sentences. [Hierarchical neural models] are designed to mimic this exact structure. By organizing the network to reflect the natural hierarchy of language, these models can better understand the relationships between different parts of a sentence. In machine translation, a hierarchical phrase-based model doesn't just translate word by word. Instead, it considers entire phrases and their correct order, resulting in translations that sound more natural and maintain the original language's structure.

[Learning Language Structure]{.Bold}

There are two main ways neural networks can learn about language structure. These are implicit learning and explicit learning. Some models are trained on large amounts of text from different languages. They gradually develop an internal understanding of language structures that work across languages. This shows that neural networks can learn grammar rules from data even without being explicitly taught. On the other hand, other models use syntactic parsing, where the network is given a "map" of a sentence's structure (often represented as a parse tree). This extra guidance helps the model focus on the grammatical relationships in a sentence. A good example of why explicit guidance matters can be seen in sentiment analysis.

[Sentiment Analysis Example]{.Bold}

Consider these two sentences.

"I didn't like the movie."

"I didn't dislike the movie."

The meaning changes dramatically with just a small difference in structure. A model that understands syntax is more likely to capture such subtle differences.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.2-Computer-Vision}7.2 Computer V[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162157}ision

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.2.1-Scene-Understanding}7.2.1 Scene Un[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162160}derstanding

Scene understanding is about teaching machines not only to recognize objects in an image but also to grasp the relationships and context that make the scene meaningful, much like how humans look at a picture and immediately understand what's going on.

[Machine Perception and Scene Graphs]{.Bold}

Machines combine what they "see" (using cameras and image processing) with what those objects represent or mean. This way, a model can both detect a "car" and understand that it might be "parked in front of" a "house". Imagine a diagram where nodes represent objects, like a car, a house, or a tree, and edges describe the relationships, such as "parked in front of". Scene graphs help models capture the structure of a scene rather than just listing objects.

[Surveillance]{.Bold}

In surveillance, a neuro-symbolic system might see a person running (via neural perception) and then infer that they could be fleeing from something (via symbolic reasoning), potentially triggering an alert.

[Visual Reasoning Techniques]{.Bold}

Visual reasoning can be performed through [deep neural network] based pipeline for object detection and [multi-modal pairwise

relationship prediction] to generate scene graphs. It can then leverage common sense knowledge from heterogeneous knowledge graphs to enrich these scene graphs, resulting in improved downstream reasoning. Visual reasoning can also be performed through [program induction]. Think of program induction as teaching a machine to "think step-by-step" like following a recipe or a series of instructions. Instead of solving a problem all at once, the model breaks it down into a sequence of simpler operations. Multi-step reasoning can be performed through [stacked attention networks (SANs)]. These models use multiple layers of focus. For example, to answer "What color is the car parked next to the tree?", in the first step, it will locate the tree. In the second step, it will identify the car next to the tree. And finally, it will determine the car's color.

[Applications]{.Bold}

Scene understanding can be applied for self-driving cars, so that it can detect other vehicles, pedestrians, and traffic signals, enabling safe navigation. It can help robots to better interact with their environment by understanding spatial layouts, object positions, and relationships, thereby helping them perform tasks like object manipulation and navigation. For AR/VR, accurate scene understanding can help digital elements to be overlaid on the real world in a seamless and natural manner.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.2.2-Visual-Question-Answering}7.2.2 Visual Q[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162163}uestion Answering

Visual Question Answering systems are like smart assistants that can look at an image and answer questions about it. They combine two main abilities for visual understanding, which helps in recognizing objects, colors, and actions in an image and language comprehension, which helps us understand what the question is asking.

[How VQA Works]{.Bold}

Imagine you see a picture of a park and someone asks, "How many people are playing frisbee?" A VQA system will look at the image and spot people. It will notice which people are playing frisbee. Finally, it will count them to give you the right answer. However, VQA systems can sometimes rely too much on patterns in the questions rather than the actual image content.

[Technological Advances]{.Bold}

There are deep learning models that combine attention mechanisms, multi-modal fusion techniques, and knowledge integration. The [Question-Guided Hybrid Convolution (QGHC)] network, for instance, uses question-guided kernels to convolute with visual features, capturing textual and visual relationships early in the process. Another advanced technique is the [MuRel] [network], which introduces a relational reasoning primitive to model interactions between question and image regions, progressively refining visual and question interactions.

[Applications and Impact]{.Bold}

Visual question answering can help people with visual impairments by describing scenes or answering questions about their surroundings. It can be used for creating interactive tools where students can ask questions about images or diagrams. It can help doctors by analyzing medical images and answering questions about potential issues. It can also enhancing online shopping experience by letting customers ask detailed questions about product images, like features, colors, or dimensions.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.2.3-Object-Recognition-with-Ontologies}7.2.3 Object R[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162166}ecognition with Ontologies

Object recognition deals with teaching computers to detect and identify items in images. Traditionally, AI systems looked at an image and simply said, "That's a dog" or "That's a car." But as images get more complicated, it becomes important not just to recognize a single object, but also to understand how objects

relate to each other and fit into larger categories. Ontologies are a structured way to represent knowledge. It can store entities, characteristics of the entities and how these are connected. It can help the system disambiguate similar objects and can generalize well in situations where training data is limited.

[Hierarchical Organization in Fine-Grained Tasks]{.Bold}

For example, in fine-grained tasks like identifying species of birds, the ontology might organize concepts in levels, such as Family, Genus, and Species. Each level adds more detail, and different CNNs can be trained to capture features at each granularity. These features are then combined to make a final decision.

[Understanding Object Interactions in Complex Scenes]{.Bold}

In images with several objects or complex scenes, ontologies can describe not only what objects are present but also how they interact. For example, in a sports scene, an ontology might include concepts like “football,” “goal,” and “green area” and use the relationships between them to decide if the image is about soccer.

[Application in Video Behavior Recognition]{.Bold}

For tasks such as video behavior recognition, the ontology defines objects and their attributes (like position, appearance, and components). It also specifies relationships (like spatial or temporal relations). For example, a series of scenes in a video may be analyzed to infer a high-level event (e.g., a person leaving their luggage) by checking if the sequence of spatial and relational conditions between objects is met.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.3-Recommendation-Systems}7.3 Recommenda[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162169}tion Systems

Recommendation systems suggest products, movies, music, or any content that fits a user’s unique tastes. Traditional approaches, such as [collaborative filtering] look at patterns of user behavior, whereas [content-based filtering] focuses on item features.

Recommendation systems struggle when there isn't enough data, otherwise known as "sparse data" or when dealing with new users and items, known as the "cold start" problem. Knowledge graphs allow systems to see not just a direct link between a user and an item, but a whole network of related concepts. By combining knowledge graphs with recommendation systems, we can achieve higher levels of accuracy and explainability.

[Collaborative Knowledge Base Embedding Framework]{.Bold}

Collaborative Knowledge Base Embedding (CKE) framework can learn representations from user-item interactions, as well as semantic features from the knowledge graph. By embedding both types of data into the same space, the system can uncover deep connections. For example, in an online bookstore, if a user likes mystery novels, the knowledge graph might reveal that they would also enjoy thrillers featuring strong female protagonists.

[Advanced Recommendation Techniques]{.Bold}

BERT4Rec uses bidirectional self-attention to analyze the sequence of user behaviors, capturing how preferences evolve over time. When enriched with knowledge graph data, it better anticipates future needs. Neural Attentive Recommendation Machine (NARM) focuses on both long-term preferences and short-term intentions by using attention mechanisms to pick out the key elements of a user's current session.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.4-Robotics}7.4 Robotics

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.4.1-Task-Planning-and-Execution}7[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162172}.4.1 Task Pla[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162177}nning and Execution

[Symbolic planning] is comparable with writing a detailed "to-do" list or a recipe. It takes a big, high-level goal, like "prepare a meal", and breaks it down into clear, logical steps. For example, if a robot

is asked to make a meal, the planner will decide on a series of steps. Starting with gathering all the necessary ingredients, followed by chopping vegetables, cooking the ingredients, and finally plating the finished dish.

[Role of Neural Networks in Task Execution]{.Bold}

[Neural networks] come into play once the plan is ready. They are excellent at handling the "hands-on" parts of the task, like moving robot arms, sensing the environment, and adjusting actions on the fly. For example, while chopping vegetables, the neural network controls the robotic arm's precise movements. And if something unexpected happens, say, an ingredient is missing, the network can quickly adjust the actions.

[Bilevel Planning Systems]{.Bold}

Bilevel planning systems have symbolic planning that abstracts the task and neural networks execute the fine details. It focuses on learning both the symbolic (predicates and skills) and neural (continuous control policies) components from demonstration data. For humanoid assistive robots, that require strict regulations, such as privacy and security, neuro symbolic approaches can help generate plans in a fast manner. It can also result in quick execution, while relying on symbolic methods, or even large language models. This can ensure that safety and policy constraints are met.

[Classical Planning and Deep Learning Integration]{.Bold}

[Classical planning] uses symbolic representations such as logical predicates, and well-defined rules to plan a sequence of actions. It's robust and interpretable but requires a detailed, often hand-crafted, model of the world. As this becomes cumbersome, [deep latent space], a deep learning network approach can learn to compress complex, high-dimensional sensory inputs into a lower-dimensional "latent" space, which retains only the most essential features needed for planning.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.4.2-Perception-and-

Action-Integration}7.4.2 Percepti[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162180}on and Action Integration

Neuro-symbolic AI for robotics integrates perception (what the robot sees and senses) with action (how the robot moves and interacts) by using neural networks to extract detailed, useful features from raw sensor data, and employing symbolic reasoning to interpret these features in a context-rich, logical way. Combining these two approaches allows a robot to understand its environment in depth and plan actions intelligently (like navigating, grasping, or assembling objects). It can adapt in real-time by continuously updating its perceptions and actions based on feedback.

For example, for navigating a cluttered room, neural network can process sensory data to detect obstacles and free pathways, whereas the symbolic planner can reason about these detected features to choose the best route to reach a destination while avoiding obstacles.

[Liquid State Machines and Real-Time Adaptation]{.Bold}

There are special type of neural networks, such as [liquid state machines (LSMs)] which works on continuous streams of sensory data in real-time and maintain a "memory" of recent inputs in their dynamic internal states. This temporary memory captures essential information that a symbolic system can use to make rapid decisions in changing environments. There are algorithms that can enable robots to learn by watching humans perform tasks. This is called [imitation learning].

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.4.3-Human-Robot-Interaction}7.4.3 Human-Ro[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162183}bot Interaction

Human-robot interaction (HRI) is all about teaching robots to understand and respond to human language naturally. Traditionally, robots follow strict, pre-programmed commands with a limited vocabulary, which makes them less useful when situations change. Neuro-symbolic AI leverages neural networks, which learn from

examples and excel at spotting patterns in raw data (like processing the sounds and words in human speech). The Symbolic AI uses clear, logical rules to make sense of information and plan actions.

[Object Manipulation and Communication]{.Bold}

In object manipulation, the robot translates natural language commands into a formal program that guides it step-by-step. In human-robot interaction, the robot can communicate through non-verbal cues as well, as it can show emotions through facial expressions, gestures, and gaze. It can build trust with users through non-verbal cues, verbal explanations, and visual aids.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.5-Artificial-General-Intelligence—AGI-}7.5 Artificial[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162186} General Intelligence (AGI)

By merging neural networks with symbolic AI, we move closer to creating systems that can reason and solve complex problems by combining learned data with logical rules. It can generalize from fewer examples and apply learned rules and concepts across different tasks and domains.

Neural networks in deep reinforcement learning learn by trial and error from massive amounts of data. They're really good at recognizing patterns and learning strategies through rewards and punishments. However, they often don't naturally perform logical reasoning or handle complex, long-term planning very well.

[Hierarchical Neuro-Symbolic Approach]{.Bold}

[Hierarchical] [neuro-symbolic]approach, otherwise known as [vertical integration] can help organize control parameters into a hierarchy of formal languages. At [lower levels], it processes input and learns patterns using neural networks. In spiking neural networks, the timing of spikes can capture dynamic information.

The key idea is to transform the sub-symbolic outputs (like neural activations) into symbolic representations. This "symbol grounding" process can be done through clustering, attention mechanisms,

or dedicated modules that map continuous data into discrete symbols (such as object categories, relationships, or logical predicates).

At [higher levels], once the data is abstracted into symbols, a symbolic reasoning engine (like a logic-based planner or a program interpreter) can perform tasks such as planning, decision-making, or language understanding. This layer can use explicit rules to reason about the relationships among symbols and generate interpretable, step-by-step outputs.

[Horizontal Integration]{.Bold}

[Horizontal Integration], otherwise known as [modular approach] builds a system where different modules handle different tasks through layered hierarchy. For example, one module might specialize in visual perception, another in natural language understanding, and yet another in motor control.

A central "reasoning hub", which is sometimes implemented as a central [spiking reasoning network] or a symbolic planner, collects and integrates the outputs of these modules. This hub can then merge the symbolic information, such as recognized objects or parsed language commands, with sub-symbolic insights (like timing and pattern details) to generate coherent, high-level decisions.

The system can use feedback loops where the symbolic reasoning not only generates actions but also refines the sub-symbolic representations. For example, if the symbolic module detects an inconsistency or ambiguity, it can signal the neural networks to adjust their focus or re-learn certain patterns.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.6-Legal-and-Ethical-Reasoning}7.6 Legal and []{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162189}Ethical Reasoning

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.6.1-Symbolic-AI-in-Legal-Reasoning}7.6.1 Symbolic[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162192} AI in Legal Reasoning

The legal field is extremely complex. It's filled with a maze of

rules, regulations, and past case decisions (precedents) that must be analyzed carefully. As the amount of legal information keeps growing, advanced tools can help legal professionals navigate and finalize the data they need in a quick and efficient manner.

[Understanding Legal Texts]{.Bold}

Symbolic AI can read and understand legal texts by breaking them down into logical rules and facts. It can take a set of case facts, apply the relevant legal rules, and suggest possible outcomes. For instance, in a contractual dispute, it might reference specific contract laws and precedents to outline potential resolutions. Because symbolic AI works with clear rules, it provides a transparent, step-by-step reasoning process. This makes it easier for legal professionals to understand why a certain decision was made and ensures that the same rules are applied uniformly.

[Automated Legal Decision-Making]{.Bold}

It can help with [automated legal decision-making]. It can do so by encoding legal knowledge into a formal structure, the system can analyze case details and automatically infer the outcome based on established laws. For example, if a dispute arises over a contract, the system can review the facts, compare them with similar past cases, and suggest a logical resolution.

[Document Analysis and Case Preparation]{.Bold}

For [document analysis] and [case preparation], AI tools can rapidly sift through massive volumes of legal documents to find relevant information, saving lawyers countless hours. It can also help predict case outcomes by comparing the current case with historical precedents, thus assisting attorneys in formulating stronger legal arguments.

[Regulatory Compliance]{.Bold}

For businesses that mandate [regulatory compliance], symbolic AI can continuously monitor changes in regulations, ensuring that

policies remain compliant. It can help companies proactively address potential legal issues before they become costly problems.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.6.2-Ethical-Reasoning-in-AI-Systems}7.6.2 Ethical []{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162195}Reasoning in AI Systems

AI systems today are being asked to make decisions that have real-world, ethical consequences. Whether it's an autonomous vehicle deciding how to react in an emergency or an AI system in healthcare recommending treatments, these decisions can deeply affect people's lives. The legal and moral rules that guide human behavior are complex and involve detailed regulations, precedents, and philosophical principles.

[Framework for Ethical Decision-Making]{.Bold}

In order for AI systems to act in accordance with human values, we must first find ways to encode ethical guidelines and moral principles into the decision-making processes of AI systems. Symbolic AI provides a framework for representing complex ethical rules in a formal, logical structure.

We can encode moral principles by [programming the AI with clear rules] derived from ethical philosophies such as utilitarianism (maximizing overall well-being), deontology (following strict rules like "do no harm"), or virtue ethics (emphasizing good character). [Logical reasoning] can be enhanced with these rules and the AI can analyze a given situation.

[A Practical Example: The Self-Driving Car]{.Bold}

For example, imagine a self-driving car that must decide whether to swerve to avoid an obstacle. The neural network interprets the situation by processing sensor data, and the symbolic reasoning engine applies ethical rules (like minimizing harm) to choose the safest maneuver.

[Transparency and Accountability]{.Bold}

As the rules and reasoning steps are explicitly defined, we have greater transparency on how the AI reached its decision. The same set of rules is applied every time, reducing the chances of biased or arbitrary decisions and more consistency in decision making. As the systems are more explainable, legal and ethical experts can trace the decision-making process, which is critical for accountability and trust.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.7-Multi-Agent-Systems}7.7 Multi-Agen[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162198}t Systems

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.7.1-Coordination-and-Communication}7.7.1 Coordina[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162201}tion and Communication

[The Importance of Clear Communication]{.Bold}

In any system with many agents (like robots or autonomous vehicles), successful teamwork depends on clear communication. Each agent must share its plans, intentions, and current status so that everyone works together without conflict or wasted effort.

[Defining a Shared Language]{.Bold}

Symbolic AI can define a [common language] that will make all agents agree on a set of symbols and rules. This shared language ensures that when one agent sends a message, others understand it exactly as intended. Different communication acts such as informing, requesting help, negotiating, or promising can be performed by agents with the help of symbolic AI. This can help ensure that the interaction remains organized and predictable. In a disaster response scenario, for example, rescue robots can use symbolic messages to report findings, share maps, and coordinate movements. This structured approach minimizes misunderstandings and avoids duplication of work.

[Adapting to Unpredictability]{.Bold}

While a shared language can help communication amongst agents,

real-world situations are [unpredictable]. To handle unpredictability, agents can learn from experience. For instance, a team of drones monitoring an environment can use neural networks to learn from weather patterns or sensor data and adjust their flight paths accordingly. By observing one another, agents can predict what their teammates might do next, helping them adapt to the evolving situation and avoid conflicts or overlaps in their tasks. If a particular way of communicating or negotiating works well, the system reinforces that behavior. If it doesn't, the system can adjust, leading to more efficient cooperation in the long run. This can help improve coordination over time.

[Managing Conflicts Through Symbolic Reasoning]{.Bold}

In any multi-agent system, [conflicts and competing goals] are inevitable. Symbolic reasoning can mandate each agent to explicitly state its goals and constraints. I.e. what it needs to do and what it cannot do. During a negotiation, agents exchange proposals and work towards an agreement that benefits everyone. Agents can be enabled to evaluate proposals by using logical rules to assess different options, weigh trade-offs, and decide on the best course of action. If an agent rejects a proposal, it can explain its reasoning (e.g., "I can't commit to this deadline because my resources are already allocated"), which builds trust and clarity among agents.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.7.2-Distributed-Problem-Solving}7.7.2 Distribu[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162204}ted Problem Solving

When many agents (like robots, vehicles, or software programs) need to work together to solve complex problems, they must coordinate and communicate effectively. This is known as distributed problem solving. No single agent can handle the whole challenge, so they share information, reason together, and make decisions as a team.

[Collaborative Reasoning]{.Bold}

[Collaborative reasoning] can help mitigate this by pooling exper-

tise from each agent, as they gather local data (e.g., obstacles, hazards, or environmental conditions) and then share that knowledge with the group. By combining these pieces, a global picture can be built of a comprehensive understanding of the situation. This can enable agents to coordinate their actions and solve problems more robustly.

[Decentralized Decision Making]{.Bold}

[Decision making] can be [decentralised] with the help of local autonomy. Instead of having one central controller making all decisions, each agent makes decisions based on its local data. Also, even though decisions are made locally, agents still communicate to ensure their actions are coordinated and aligned with the overall goal.

For example, in the case of smart power grid, devices like household appliances and industrial machines act as independent agents. They monitor local conditions (like energy demand) and make decisions to balance supply and demand without waiting for instructions from a central system.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.8-AI-Driven-Drug-Discovery}7.8 AI-Driven []{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162209}Drug Discovery

Designing new drugs means working with [molecules], which are very complex three-dimensional structures made up of atoms and bonds. To understand how a molecule might behave with respect to its reactivity, stability, and how it interacts with biological systems, we need a detailed way to represent its structure.

[Molecular Representation]{.Bold}

Symbolic AI with the help of rule-based representations can model [molecules as graphs] where each atom is a node and each chemical bond is an edge. This format makes it easy to analyze how atoms connect, which parts of the molecule are similar to each other, and how they might interact. Another popular way to represent

molecules is using [linear notations like SMILES] strings. These are like short, coded sentences that describe the molecule's structure in a linear format. Symbolic AI systems can read these strings, convert them into graphs, and compare different molecules.

[Substructure Identification]{.Bold}

Once a molecule is represented symbolically, [AI can search] for specific patterns or ["substructures"] (like a particular functional group) that are known to affect a drug's properties. This is key for predicting whether a molecule might work as a drug.

[Predicting Chemical Properties]{.Bold}

Symbolic AI can also help predict important chemical properties. Using [established chemical principles, rules, and heuristics] symbolic AI system can infer properties like solubility, stability, and bioavailability. For instance, it might determine that a molecule with a hydroxyl (—OH) group is likely to be more water-soluble. By [integrating with neural networks], the system can learn from experimental data and continuously update its rules to improve accuracy. This neuro-symbolic approach has the adaptability of neural networks and the clarity of symbolic reasoning.

[Applications in Drug Discovery]{.Bold}

It can be applied at multiple stages and processes of drug discovery. One key property in [drug design] is ["drug-likeness",] which evaluates whether a molecule has characteristics similar to known drugs. Symbolic AI can analyze features like molecular weight, lipophilicity, and the number of hydrogen bond donors/acceptors, often using guidelines like Lipinski's Rule of Five.

[Lead Optimization]{.Bold}

For [drug discovery], it can help us with [lead optimization] by exploring chemical space. Once a promising molecule (a lead compound) is identified, symbolic AI can systematically generate and evaluate variants (analogs) by applying transformations, such as adding or substituting chemical groups in order to improve its

efficacy and safety. Instead of physically testing thousands of molecules in a lab, symbolic AI can [rapidly screen vast libraries of compounds virtually], by filtering out those unlikely to be effective based on their symbolic representations. This means focusing resources on the most promising candidates.

[Enzyme Inhibition and Candidate Screening]{.Bold}

Imagine searching for a new drug that can inhibit an enzyme related to a disease. The AI system can analyze the enzyme's active site, determine what molecular features are needed for effective inhibition, and then match these features against a database of molecules. This process can highlight potential inhibitors that might be missed by traditional methods. Neuro-symbolic AI can also help with [lead optimization] and suggest modifications (like adding or altering functional groups) for promising candidates to improve drug properties, such as efficacy and safety.

[Drug Repurposing]{.Bold}

We can use neural networks to detect patterns that hint a drug approved for one condition might work for another, by leveraging data sources such as genomic profiles, health records, and scientific literature. This can help us find new uses for existing drugs, otherwise known as [drug repurposing]. Symbolic reasoning can then help confirm the hypotheses generated by neural networks by checking them against known biological pathways. For example, if Drug A targets Protein X and Protein X is also involved in Disease Y, the system might suggest repurposing Drug A for Disease Y.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.9-Healthcare-Applications}7.9 Healthcare[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162212} Applications

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.9.1-Clinical-Decision-Support}7.9.1 Clinical[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162215} Decision Support

Neural networks can analyze huge amounts of image data to detect

anomalies, say, finding lung nodules that might indicate early-stage lung cancer. They can learn from thousands of examples, identifying patterns that might be too subtle for a human eye. Once the neural network extracts features from the images, symbolic reasoning can translate these into explicit, logical terms (e.g., "irregular shape" or "abnormal tissue density"). This makes it clear why a certain area is flagged as concerning.

[Interpretation of Laboratory Tests]{.Bold}

For interpreting laboratory tests an AI system can examine blood test results along with patient symptoms. By using predefined medical rules, it can suggest possible diagnoses like anemia or an infection. The system doesn't just give a diagnosis, it can explain its reasoning step by step. This can help clinicians understand the decision and trust the AI's recommendation.

[Personalized Treatment Plans]{.Bold}

Every patient is different. Neuro-symbolic AI can take in patient-specific data, such as genetic information, lifestyle, and medical history, and propose personalized treatments. Neural network can identify important patterns in the patient data, while symbolic reasoning can ensure that the treatment plan aligns with established medical guidelines and ethical principles. For instance, in cancer care, the system might recommend a specific chemotherapy regimen based on the tumor's genetic profile.

[Risk Assessment in Cardiology]{.Bold}

In critical areas like cardiology, the system can assess the risk of events (e.g., heart attacks) by analyzing factors such as cholesterol levels, blood pressure, and family history. Symbolic reasoning can provide a clear explanation in such situations. For example, "High cholesterol and a family history of heart disease contribute to an increased risk of heart attack." This helps clinicians decide on preventive measures.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.9.2-Medical-

Knowledge-Representation}7.9.2 Medical []{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162218}Knowledge Representation

Ontologies are foundational for medical knowledge representation. Ontologies are like detailed blueprints or dictionaries that define how medical concepts relate to each other. They provide a standardized vocabulary so that everyone, from doctors to computer systems, use the same terms in the same way.

[Standardization and Integration]{.Bold}

For example, the term "myocardial infarction" (heart attack) is uniformly understood, even if different hospitals might originally label it differently. Ontologies make it easier to integrate data from different sources (like patient records), improve information searches, and support decision-making systems by ensuring consistency and clarity.

[Application in Electronic Health Records]{.Bold}

For example, electronic health records (EHR's) contain a mix of structured and unstructured data (like free-text doctor's notes). Neural networks can process these diverse data types, and then symbolic reasoning maps the extracted information onto the standardized terms defined by medical ontologies. Imagine an AI system reviewing a patient's EHR. It detects a mention of a "blood thinner" in a doctor's note and, using ontological mapping, standardizes this to "warfarin." It then uses established rules to assess potential drug interactions and alerts clinicians if there are risks.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.10-Automotive-Industry}7.10 Automotiv[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162221}e Industry

The automotive industry is undergoing a transformative revolution driven by advancements in artificial intelligence and machine learning. Among these innovations, neuro-symbolic AI stands out as a promising approach that combines the learning capabilities of neural networks with the interpretability of symbolic

reasoning. This fusion offers significant potential to enhance autonomous driving systems, optimize vehicle maintenance, and improve human-machine interactions within vehicles.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.10.1-Enhancing-Autonomous-Driving-with-Neuro-Symbolic-AI}7.10.1 Enhanci[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162224}ng Autonomous Driving with Neuro-Symbolic AI

Autonomous vehicles use sensors like cameras, LiDAR, and radar to capture a huge amount of information about their surroundings in order to understand their environment. Neural networks process this raw data and can recognize key elements such as pedestrians, other vehicles, and road signs.

[Contextual Understanding Through Symbolic Reasoning]{.Bold}

However, simply recognizing objects isn't enough. The vehicle also needs to understand what those objects mean in context. For example, when a neural network identifies a stop sign, symbolic reasoning takes over to apply the rule that the car must come to a complete stop. By merging these two approaches, the vehicle doesn't just "see" the stop sign, it understands the rule behind it and acts accordingly.

[Adaptability in Unpredictable Environments]{.Bold}

The real world is unpredictable. For instance, road construction might force a detour. Neural networks learn from experience to detect unusual situations and adapt. Symbolic reasoning can enhance safety. For example, even if the route is not standard, symbolic reasoning helps the car decide how to navigate safely while still obeying traffic rules.

[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x7.10.2-Ethical-Considerations-and-Safety-Regulations}7.10.2 Ethical[]{#Neuro-Symbolic-AI-EPUB-10.xhtml#x.162227} Considerations and Safety Regulations

[Ethical Decision-Making in Critical Scenarios]{.Bold}

In critical scenarios such as an unavoidable accident, the car's AI must choose an action that could have moral implications, such as deciding whether to protect its passengers or avoid harming pedestrians. Using symbolic reasoning, ethical guidelines (for example, "preserve human life" or "minimize harm") are explicitly programmed into the system as embedded ethics. This ensures that the car's decisions follow accepted ethical and legal standards.

[Regulatory Compliance and Transparency]{.Bold}

Regulatory bodies require that autonomous vehicles meet strict safety standards. With transparent reasoning, manufacturers can demonstrate to regulators that the vehicle's decisions are safe and reliable. This clear explanation is essential not only for certification but also for building public confidence in autonomous systems.