Charting the Landscape of Neuro-symbolic Reasoners

An Xuelong

4th Year Project Report
Cognitive Science (Humanities)
School of Informatics
University of Edinburgh

2023

Abstract

Pure symbolic or logical approaches to learning are brittle as they are inflexible and do not consider uncertainty. Pure deep learning approaches are equally brittle as they cannot robustly generalize. Neuro-symbolic models try to overcome these two limitations by combining the ability of deep learning approaches as feature engineers/extractors, with the intelligible reasoning capabilities of classical symbolic models.

Interest over this family of models is growing, with a constant influx of novel models and benchmarks to test their robust generalization and reasoning capabilities. However, much of the successes reported by a lot of neuro-symbolic methods over assessed datasets are often too disparate to one another, and it is an ongoing research endeavor to seek a common benchmark suite for which to comprehensively test the plethora of neuro-symbolic models. The prerequisites of devising such benchmark involve understanding how neuro-symbolic methods and datasets relate to one another.

In this thesis, we survey the current panorama of neuro-symbolic model architectures and benchmarks. As a result, we propose a general taxonomy for classifying current and future neuro-symbolic models and reasoning benchmarks. From this, we propose SaSSY-CLEVR, a highly heterogeneous common benchmark suite which can serve as a common testing ground for different neuro-symbolic reasoners to compare and contrast their strenghts and limitations. We ran experiments on one aspect of our reasoning benchmark, namely CLEVR-Hans3 to showcase how can we conduct comprehensive and fair model comparison between neuro-symbolic and deep learning approaches.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*An Xuelong*)

Acknowledgements

We are living in very interesting times where the introduction of large-language models have seemingly dominated the AI discourse and have framed the pursuit of artificial intelligence as a matter of scaling-up transformer-based neural networks. Amidst this frenzy of building ever larger networks, I've had the luck of meeting Antonio and his lab, APRIL, who introduced me to the field of neuro-symbolic AI. I'm deeply inspired by their braveness in navigating through the yet unexplored waters of neurosymbolic AI which is surely riddled by challenges previously unseen in deep learning, but nonetheless worthy to pursuit if we want to build reliable, parsimonious and transparent AI models. I also want to particularly thank my friends such as Lorenzo Loconte from APRIL who helped me understand ProbLog syntax, Alessandro Palmarini, Andreas Grivas, Paolo Cassina and Shino Chen for the insightful discussions on neurosymbolic AI and AI in general. I learnt a lot from simply speaking to you, as well as working with you. Lastly, I can't stress enough the importance of my family's support. Despite being far from home and not be able to be in physical contact with them, their love serves as a backbone for me to look forward on my studies. For a future where we can build reliable AI tools that can help us in our daily lives, the current introductory work serves as the first step towards a long voyage of the pursuit of artificial intelligence.

Table of Contents

Chapter 1

Introduction

1.1 A brief history of Artificial Intelligence

Seventy years have elapsed since the Summer Research Project held at Dartmouth College in which the systematic pursuit of artificially-designed intelligence was proposed, inspired by the aspiration that ”every aspect of learning or any other feature of intelligence can in principle ... be simulated by a machine” [Moor, 2006, McCarthy et al., 1955]. A cycle of winters and springs ensued, characterized by the transient rise and steep falls of both symbolic and neurophysiologically-inspired models. Symbolic models failed due to their brittleness exposed by innability to handle unstructured input given the inflexibility of their rules-based architecture and hard-coded, logic-based representation of knowledge. Furthermore, despite an over-saturation of encoded knowledge, they still lacked commonsense reasoning, and were unable to solve novel tasks. One such canonical example is the ambitious expert system named CYC (short for encyclopedia) created by computer scientists Douglas Lenat and Edward Feigenbaum aimed at encoding all necessary knowledge to a machine in order to instill commonsense understanding [Clark, 1997], which unsurprisingly was unable to achieve this goal. This was later followed by a wave of interest centered around neural networks inspired by the mimicking of the biological structure of the brain, with pioneering work carried out by David E. Rumelhart and James L. McClelland [McClelland et al., 1986]. Focus shifted instead on learning representations from raw, low-level, numerical data in order to statistically and accurately map unstructured input to desired output. Additionally, unlike symbolic models which mainly performed serial computations and stored information localized in data structures, neural networks aimed at emulating the brain’s ability to do parallel computation and store information distributed across their connections [Nilsson, 2009]. However, its attention faded due to the inability to scale up computation as well as lack of data to configure those models [Muthukrishnan et al., 2020]. At its initial stages, given the limited computing power available during the late 20th century, neural networks were restricted to solving problems of limited-scale such as handwritten, single, black-and-white digit recognition.

With the introduction of Graphical Processing Units (GPUs) and the abundance of data brought by the Internet Revolution, the epoch past the 2010’s welcomed an unprece-

dented amount of attention targeted at probabilistic Deep Learning (DL), which greatly eclipsed attention given to symbolic models (see an analysis from [Zhang et al., 2021]). This is justified given the recurrent noteworthy performance in a plethora of benchmarks meant to measure diverse faculties of human intelligence, such as the landmark successes at the canonical benchmark of ImageNet meant to chart out how an AI model can simulate Human Vision through classification (see AlexNet by [Krizhevsky et al., 2012], a vision model widely regarded as a catalyst for such renewed attention), or translation benchmarks to measure Human Language understanding. This generally prompted researchers to shift from symbolic approaches towards statistical ones [Zhang et al., 2021].

The trend of successes that followed and reiterative redefinition of what constitutes the "new state of the art (SoTA) performance" has given form to a hypothesis that deep learning models can become more powerful (i.e. achieve more accurate predictions) through scaling up of the underlying network and the curated dataset used for training it. We refer to it as the "Scaling Hypothesis", a term borrowed from [Branwen, 2020]. That is, arguments backing this view hold that the improvement of prediction accuracy of deep learning architectures are to be framed as issues of curating a large enough dataset, enough processing power and larger architectures in terms of parameters. This would, it is hoped, be enough to robustly solve challenging tasks meant to measure faculties of intelligence, such as cross-linguistic translation, visual question answering or high-resolution text-to-image generation. Multiple empirical sources provide the confidence for such claims. For instance, in an analysis done by [Kaplan et al., 2020] on neural language models, they showed that language modeling performance improved smoothly as the neural model's size, dataset size, and amount of compute used for training increased (see Figure 1.1). This view is further reinforced by the recent advances in the transformer-based deep neural network and its variations, which have the capability to learn patterns of training-data across modalities (text-text, text-to-image, image-text, image-to-image), thus overcoming a diverse array of multimodal benchmarks, e.g., see FLAVA, a hybrid vision-language model which outperformed several previously state-of-the art models on a series of multimodal tasks [Singh et al., 2022]. Furthermore, given an apparent flexibility in adapting it to solving new modalities of tasks without having to fully-retrain the transformer architecture, [Lu et al., 2021] even proposed them as universal computation engines. Generally, it is observed that claims which support the scaling hypothesis revolve around successes on the aforementioned benchmarks, achieved by bigger models.

1.2 Neuro-Sybmolic AI

However, careful observers of the scaling hypothesis have noted that purely deep learning systems are not as trendsetting as seemingly portrayed, mainly due to their brittleness for robust generalization beyond the training set distribution, lack of explainability tied to the lack of computational semantics, and lack of parsimony due to its evident over-reliance to big data, and unacceptable levels of computational power consumption [Garcez and Lamb, 2020, Harmelen and Teije, 2019]. The sources of such problems vary, and ongoing research seek to pinpoint, or even to understand them. Some argue that it is due to the incapability of purely learning systems to disentangle con-

Empirical basis for the scaling-up hypothesis.

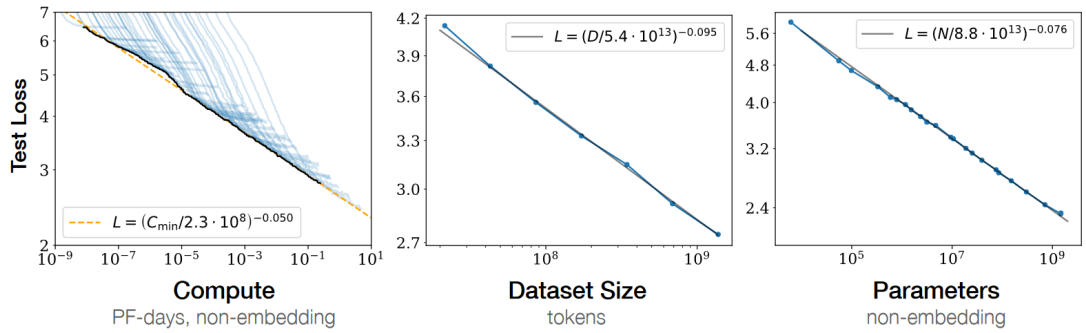


Figure 1.1: Researchers at OpenAI showed that the performance of a language autoregressive model, measured as the cross-entropy loss averaged over a 1024-tokens context, improved linearly given exponential increases in computing power, dataset size and number of parameters in the model's design. Figure extracted from [Kaplan et al., 2020]

cepts such as shape, color or size from the training data and reason about them (see [Higgins et al., 2018] for a more elaborate discussion on disentangled representation of data). Instead, they learn distributed representations of the incoming raw data that are indiscernible given that the weight parameters of the model do not carry any semantic meaning nor reflect the real world in any meaningful way. For example, a deep learning system may have to observe thousands of finely-procured images of a cat in order to recognize it through pattern recognition that would not reliably generalize well to a testing distribution of cat images. For instance, if the training set mainly consisted of pictures of real-life cats, then it is uncertain whether this model can generalize to out-of-distribution (OOD) samples such as cartoon cats, or cats appearing in highly unusual backgrounds. Nonetheless, if they were able to disentangle the concepts of shape and color that are representative of a cat, then, it is argued, not only would it rely on less data, it could robustly generalize to unknown distributions by reasoning over these concepts. Others point out that purely deep learning's weaknesses are immediate consequences of the inherent biases present in some of the canonical benchmarks they are trained on. Specifically, benchmarks for computer vision like ImageNet contain implicit biases such as object-centrism, i.e. images often show the object to be classified conveniently centered, without being rotated, under a well-illuminated background, and without confounding objects [Barbu et al., 2019]. This has led to models trained and having excelling performance on ImageNet to experience accuracy drops of around $40-45\%$ when tested on ObjectNet which control for such biases (Figure 1.2), irrespective of size. Another source of concern is due to the lack of transparency of purely deep learning models both at a functional and structural level. That is, there is no theoretical framework able to explain the underlying functional behaviour of DL models. Thus, there are no mathematical proofs or explanations of why DL models output right and wrong predictions. Although attempts have been made in making a DL model's internal architectures discernible through saliency maps over the input space [Yu et al., 2014], authors such as [Stammer et al., 2021a] have pointed out that such mappings are agnostic to the correctness of a prediction, and thus uninformative

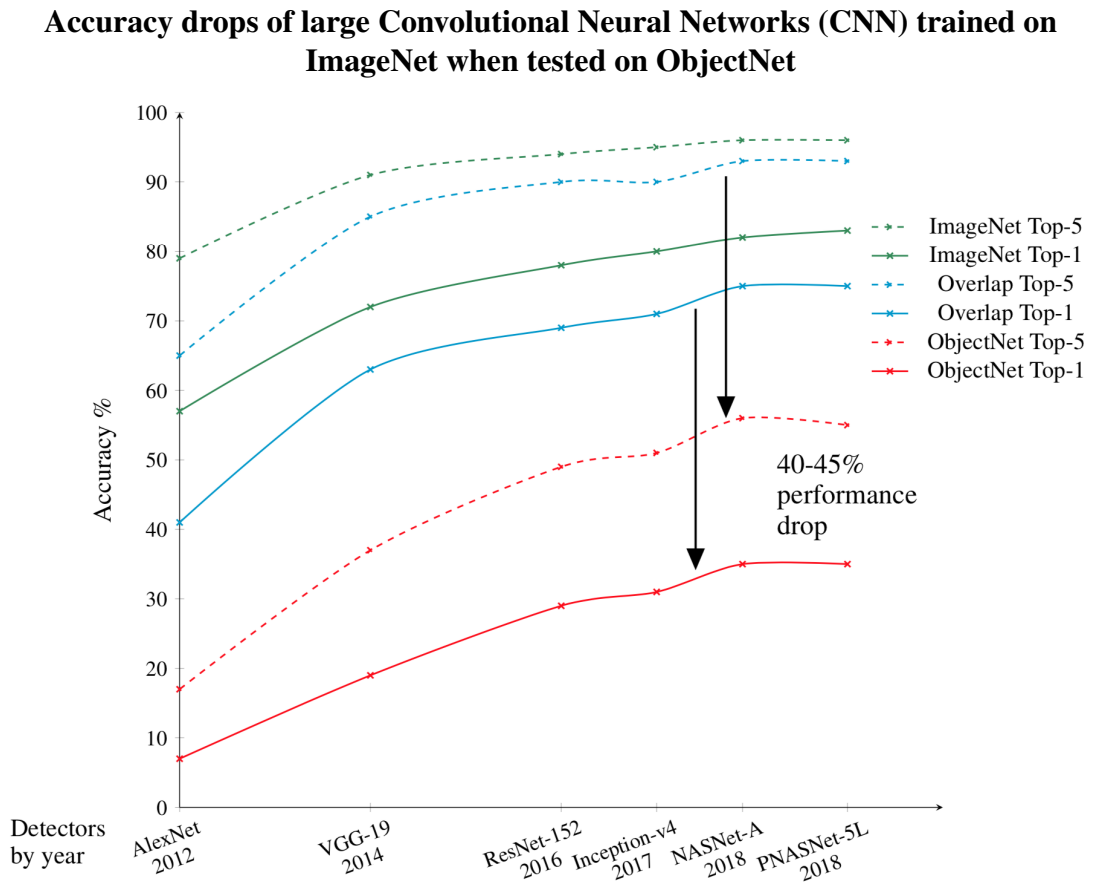


Figure 1.2: Researchers at MIT noted that SoTA CNNs trained on ImageNet across the years experienced classification accuracy drops, as measured by top-1 and top-5 performance, when tested on ObjectNet which controls for object-centric bias. Figure extracted from [Barbu et al., 2019]

of whether the model has grasped why a prediction is right from wrong. Structurally, it is also unknown why changes in the neural architectures can achieve higher prediction accuracy than predecessors in the same benchmarks. For instance, to address gradient vanishing and build deeper neural networks, [He et al., 2016] proposed residual networks (ResNet). It has been empirically shown that adding their residual connections to a neural architecture can also smooth the associated loss function's landscape [Li et al., 2018], optimizing gradient descent convergence. However, the cause behind such phenomenon remain unclear. Similar elusiveness is present when explaining the capabilities of transformer-based neural architectures, as well as why do they have to be assembled as described in [Vaswani et al., 2017]. Owed to this black-box architecture, [Liu et al., 2021] proposed an alternative model, called gated multi-layer perceptrons, which lacks a self-attention mechanism, and can achieve similar accuracy in benchmarks originally addressed by transformers whilst trained on a less amount of parameters. They interestingly point out that they neither know why gMLPs succeed over transformer, nor why transformers are successful.

Given these shortfalls tied to the hypothesis of scaling up deep neural architectures, an

alternative approach which we refer to as the "integration hypothesis"[1] seeks to revisit the school of symbolism to build hybrid models and formulate new benchmarks that can address the aforementioned obstacles.

While symbolic models are well known to be brittle for their incapability to handle uncertainty, they have strengths that can complement the shortfalls of neural networks. Purely symbolic models such as expert-systems can be saturated with hard-coded rules, and still be unable to capture all the forms an input can take or process unstructured data. Nonetheless, they are credited for their transparency and explainability of output. This is due to their explicit logic syntax, rules-based approach to processing data which is theoretically understandable. They are also parsimonious because in contrast to neural networks which are data-hungry and model-size hungry, purely symbolic models perform better through defining a low amount of putative rules because otherwise the search space for solutions grows exponentially [Van Krieken et al., 2022], and computing logical inference becomes intractable. They are also able to easily incorporate human expert knowledge in their design and function [Saker et al., 2021] grounded on the task to be solved.

Given the flexibility of purely deep learning models and intelligible reasoning process of purely symbolic models, the integration hypothesis rests on the promising potential that a hybrid Neuro-Symbolic (NeSy) model can complement the shortfalls of each other, whilst expanding each other's capabilities [Harmelen and Teije, 2019, Garcez and Lamb, 2020] without compromising each others' strengths (Figure 2.1). Such integration can take many forms, such as equipping neural networks with the ability to reason, or using neural networks for informed search over the hypothesis space of a symbolic model as opposed to uninformative search strategies like depth-first search. Argued advantages of such hybrid include 1) being robust to training from small data sets which could be plagued by poor quality data by exploiting prior domain-specific expert knowledge, 2) improve explainability through explicit knowledge representation, 3) constrain and optimize learning in neural networks by incorporating inductive biases tailored to the task, and/or 4) being able to process more diverse data modalities and solve multimodal problems by addressing different input representations [Harmelen and Teije, 2019, Garcez and Lamb, 2020, Saker et al., 2021, Sun et al., 2022] such as structured knowledge graphs and unstructured tensors.

There has been a myriad of models being proposed as a result of this interest in NeSy, along with challenging benchmarks to empirically assess them (which we thoroughly discuss in 3.0.2. However, the successes reported by a plethora of research teams are often disparate with respect to one another as a consequence of substantially different model architectures proposed, and diverse benchmarks used to test them. As [Garcez and Lamb, 2020] stated, *"NeSy AI is in need of standard benchmarks...[to] provide a fair comparative evaluation of different approaches..."*. It is therefore an ongoing research endeavor to build a comprehensive view of the current landscape of NeSy models and available benchmarks. This would be helpful in gauging the capabilities of neuro-symbolic models relative to each other, as well as understand

[1]The choice of the term "integration" is to reflect the idea that the merging of models of these dichotomous schools of thoughts result in a "whole greater than the sum of its parts". Hence, "integration" is best distinguished from a simple summation of deep learning and symbolism.

what different benchmarks are essentially assessing. In the present work, we provide a thorough review of the integration hypothesis by surveying the growing literature on neurosymbolic models and benchmarks to test them.

Over the course this thesis, we aim to chart out both the benchmarks currently available and the capabilities of these models by empirically evaluating them on a devised common benchmark suite. In order to do this, in Chapter 2 we first provide a systematic review of what are the strengths and weaknesses of both symbolic and learning paradigms, motivating the combination of both in hybrid architectures. Afterwards, in Chapter 3 we propose a taxonomy for understanding how the myriad of neuro-symbolic models relate to one another, along with a distillation of the various datasets employed to test them. As a result of this understanding, in Chapter 4 we devise a common benchmark suite able to comprehensively test the capabilities of a neuro-symbolic model, and is also useful for making inter-model comparisons. Given the heterogeneity of our benchmark, in Chapter 5 we focus on running experiments with neuro-symbolic model variants on one aspect our benchmark, outlining guidelines on running fair model comparison and comparing the capabilities of different architectures. We follow this with a discussion of the results obtained, analyzing their current strengths and weaknesses, and how they inspire further research in the neuro-symbolic literature. In Chapter 6 we conclude with a synthesis of our findings, and future prospects of NeSy AI.

Our main contributions can be summarized as three-fold:

1. We provide to the research community an overarching taxonomy in order to classify current and future NeSy models proposed in the literature, allowing us to firsthand gauge the strenghts and limitations of models falling within each category.
2. We propose a common reasoning benchmark suite accounting for the diverse challenges proposed for NeSy architectures so far, and thus useful for running comprehensive experiments amongst NeSy methods.
3. By running experiments on publicly available NeSy models on one aspect of our benchmark suite, we explore on how to conduct fair model comparison between NeSy methods, how to highlight their strengths over purely deep learning baselines and discuss the strengths and weaknesses among NeSy candidates.

Chapter 2

Purely Symbolic and Deep Learning Models

In order to thoroughly understand the motivation behind the marriage of the school of symbolism and connectionism, we provide a brief, functional description of models from both schools.

Symbolism

One of the earliest speculations of the 1956 Dartmouth Summer Project is that *"a large part of human thought consists of manipulating words according to rules of reasoning and rules of conjecture"*, an idea that shaped AI for decades [Moor, 2006]. The basis of the symbolic school of thought is logic, a system which represents knowledge through the use of symbols, and the manipulation of such symbols is a process referred to as "inference".[1] Propositional logic is the most basic form by which statements about reality can be represented. Given a *proposition* $\boldsymbol{P}$, it can be evaluated to either True or False. Propositions can be connected through logical connectives $\rightarrow, \neg, \wedge$ and $\vee$, which correspond to either *if-then*, *negation of*, *and* and *or* to form a compound proposition describing reality. Because propositions take zero arguments, they are of *arity* zero.

Relational logic builds on top of this formal system by allowing propositions to take in arguments, then acquiring the name of *atoms* or *predicates*, allowing the representation of *relations* if the predicate is binary, or *properties* if it is unary. These atomic formulas or predicates can either take a variable or term (written in uppercase) $smoker(X)$ and/or it can be *grounded* to a constant, making the predicate be known as an entity/fact/ground-atom, e.g., $smoker(shirley)$ (written in lowercase). Both denote *X is a smoker* and *Shirley is a smoker* respectively. A *clause/rule* is a disjunction of literals and is often written as $h \leftarrow b_1 \wedge b_2 \wedge \ldots b_k$, where h is the *head* of the rule, and it implies the literals b_i which constitute the *body*. This can be rewritten as a *Horn clause* as $h \vee \neg b_1 \vee \neg b_k \ldots$, or when written in a programming language it is h :- $b_1, b_2 \ldots, b_k$

[1] In this thesis, we do not engage in a terminology dispute of what a symbol is. At a pragmatic level, symbols refer to the usage of logical predicates, while subsymbols refer to the usage of tensors of numbers when solving a task

[Šourek et al., 2015, Manhaeve et al., 2021]. A *logic program* is a set of atoms and rules describing how atoms relate to one another. First order logic (FOL) builds on top of relational logic by allowing the use of universal or existential quantifiers $\forall, \exists$. However, in practice, such as in logic programming, variables appearing in rules are assumed to be universally quantified [Manhaeve et al., 2021], and thus relational logic and FOL often refer to the same logic system.

Fuzzy logic relaxes the logical semantics so that predicates can be evaluated to a continuous spectrum in [0, 1], thus becoming differentiable. This is at the cost that logical equivalences don't hold as originally did and logical predicates lose their semantic meaning [Van Krieken et al., 2022]. Probabilistic logic allows ground-atoms to have probability distributions attached to them. They're often written as a annotated disjunction $0.8 :: h$:- $b_1, b_2 \ldots b_k$, which in contrast to fuzzy logic retain their original semantic meaning. The trade-off, however, is that NeSy architectures which incorporate probabilistic logic programs are difficult to scale-up as the symbolic component inherits the necessity to search over an exponentially growing hypothesis space specified by the program, a #P-hard problem known as Weighted Model Counting [Chavira and Darwiche, 2008], with ongoing work seeking to improve the tractability of probabilistic logical inference [Van Krieken et al., 2022].

For a purely symbolic program to solve a task, we must first prescribe a domain-specific language (DSL) which will be used to write a logic program which serves as an abstraction to the problem at hand. For example, consider the symbolic model in Figure 2.1a. It specifies a logic program P through FOL to represent a small world denoting a group of friends and how are the dynamics of friendship, smoking and emphysema. It can be given a query q of inferring whether it is true that Kerry has emphysema. The symbolic model would first check whether the query q is an explicitly stated fact in the program. If not, it tries to prove the truth of the query by grounding the variables to the constants, a process called *unification*. To further find applicable rules, the model attempts to unify q with the heads of all available rules. If unification succeeds, variable substitutions occur in the atoms as stated in the body of the rule. Those atoms becomes a subgoal, and each subgoal is recursively proven using this same process [Weber et al., 2019].

From this brief description of the pipeline of a symbolic model, we can note the challenges tied to running symbolic models. As the amount of rules increase, the search space in order to prove a query grows exponentially, therefore symbolic models have been weak at scaling up. Subsequently, it is also a very intricate endeavor the process of designing and incorporating the necessary domain-specific knowledge tailored at the task to be solved. This makes them inflexible to transfer among substantially different tasks. The upside to these issues is that the logic program is transparent, and compelled to be parsimonious to be tractable.

2.0.1 Deep learning

Deep learning is a paradigm shift from the symbolic school of thought. The learning school of thought employs vectors (also referred to as *sub-symbols* in the literature) to represent knowledge, such as one-hot vectors for words in language tasks, pixel

matrices in computer vision tasks for brightness and weight matrices embedded in the neural architecture to refer to latent, distributed knowledge representations. A deep neural network (DNN) is a biologically inspired model made of directed, acyclic stacked layers of perceptrons where the weights across layers are tune-able parameters according to incoming batches of training data [Goodfellow et al., 2016].

Specifically, given a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{X}$ consists of input features and $\mathbf{y}$ is the desired output, a DNN is a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ that maps vectors from input space to target space [Šourek et al., 2015]. In an *autoencoding* task, the target space is the original input, and the function is a "bottleneck" to learn how to reconstruct the input. This function is composed via series of layers l, each parametrized by a weight matrix W^l. Each prior layer is related to the next one via a differentiable activation function τ, such as a rectified linear unit (ReLU) for intermediate layers and a final activation for the output layer σ. Put together, this results in:

$$\hat{\mathbf{y}} = \sigma(W^l \tau^l (W^{l-1} \dots W^2 \tau^1 (\mathbf{X} W^1)))$$

, where $\hat{y} \in \mathbb{R}^n$ is the predicted output, τ could be the ReLU activation function, and σ could be the softmax function if the task is multi-class classification. Each $W^l \in \mathcal{W}$. Prior knowledge, commonly called *inductive biases*, shapes the design of the function. If the input data consists of image and the function must account for translational, scaling or rotational invariance, then f is a *convolutional neural networks* (CNN). If the data is sequential and has long term dependencies, f can be either a *recurrent neural network* (RNN) or *transformer-based neural network*. If the data must learn about relationship between entities, f can be a *graph neural network* (GNN) [Goodfellow et al., 2016].

Depending on the assumptions about the training data distribution and the nature of the task to be solved, e.g., classification or prediction, an error function $\mathcal{L} : \{\hat{\mathbf{y}}, \mathbf{y}\} \rightarrow \mathbb{R}$ is defined, such as the binary cross-entropy loss for binary classification. From there, the disparity between predicted output and ground-truth is used as a signal that is backpropagated from the last layer to earlier ones in order to adapt weights through stochastic gradient descent to minimize the prediction error of the network. The optimal set of weights is $\mathcal{W}^* = arg\min_{\mathcal{W}^*} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ computed from the training data. A validation set is used to find optimal hyperparameters, such as number of layers, perceptrons per layer, batch size of training data, epoch, with particular focus on learning rate α that controls convergence of gradient descent or other architectural concerns like dropout rate that controls overfitting. The overall neural network learns to approximate the probability distribution of the training data after observing it for multiple batches across several epochs [Šourek et al., 2015], as depicted in Algorithm 1.

The performance of the DNN is evaluated on a never-seen test set. The current literature in deep learning is interested in generalization, the ability to perform well on the test set without memorizing the training data distribution. Challenges arise when the testing distribution does not reflect the training distribution, a phenomenon known as generalization to *out-of distribution* data.

A probabilistic treatment can be given to the above deep architecture by assuming weights $\mathbf{W}_i \sim p(\mathbf{W}_i)$, input features $\mathbf{X}_i \sim p(\mathbf{X}_i)$ and labels $\mathbf{y} \sim p(\mathbf{Y})$ are random vari-

Algorithm 1 Stochastic gradient descent for training neural networks and choosing optimal hyperparameter

Input: Training dataset $\mathcal{D}_{train} = \{\mathbf{X}_{train}, \mathbf{y}_{train}\}$, Validation dataset $\mathcal{D}_{val} = \{\mathbf{X}_{val}, \mathbf{y}_{val}\}$
hyperparameters: number of layers, epoch, minibatch size, α, among others.
Output: optimal $\mathcal{W}^*$
Initialize $\mathcal{W}$
for hyperparam α in grid **do**
  for epoch **do**
    for minibatch in $\mathcal{D}_{train}$ **do**
      forward pass: $\hat{\mathbf{y}} = f(X_{minibatch}; \mathcal{W})$
      stochastic gradient descent: $\mathcal{W} \leftarrow \mathcal{W} - \alpha \nabla_{\mathcal{W}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
    end for
  end for
  validation forward pass: $\hat{\mathbf{y}} = f(X_{validation}; \mathcal{W})$
  compute validation loss: $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}_{val})$
end for
Output optimal $\mathcal{W}^*$ with lowest validation loss

ables with underlying distributions describing them. The goal becomes estimating the data generating process $p(\mathcal{D})$. With a simple function f like linear regression, which can be thought of as a DNN with one layer, the distribution of the weights $\mathcal{W}$ is estimated via Bayes Theorem:

$$p(\mathcal{W}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{W})p(\mathcal{W})}{p(\mathcal{D})}$$

, where $p(\mathcal{D}|\mathcal{W})$ is the likelihood of observing data given the current parameters, $p(\mathcal{W})$ is the prior over parameters that encode domain expertise on the plausible range of values of the parameters before seeing any data, and $p(\mathcal{D})$ is the model evidence or normalization constant, which is the expectation of the likelihood under the prior, and serves to ensure the posterior is a valid probability distribution.

As an example of deep learning, consider now Figure 2.1b, where we can try to solve the same problem of querying whether Kerry has emphysema by employing a neural network. Its usage opens the opportunity to represent the entity as Kerry through a tensor of numeric features. This input space allows for a representation that can account for uncertainty, e.g., feature x_1 can be the frequency of cough which could be recorded from a wearable sensor by Kerry, where the sensor can capture noise from the environment. Given enough training data (samples of patients with emphysema or controls), and framing this problem as a binary classification task, the model could learn to discriminate with high accuracy patients with emphysema from those who do not in the training set. However, the output results are often untrustworthy given that the underlying process mapping the input to the output is commonly regarded as a black-box. Furthermore, there is no theoretical guarantee that there would be a small generalization gap if the model is to predict over a test set which data distribution may not reflect the training data distribution. For example, an implicit bias in the training

Complementareity of symbolic models and neural networks exemplified through a task of prediction of having emphysema

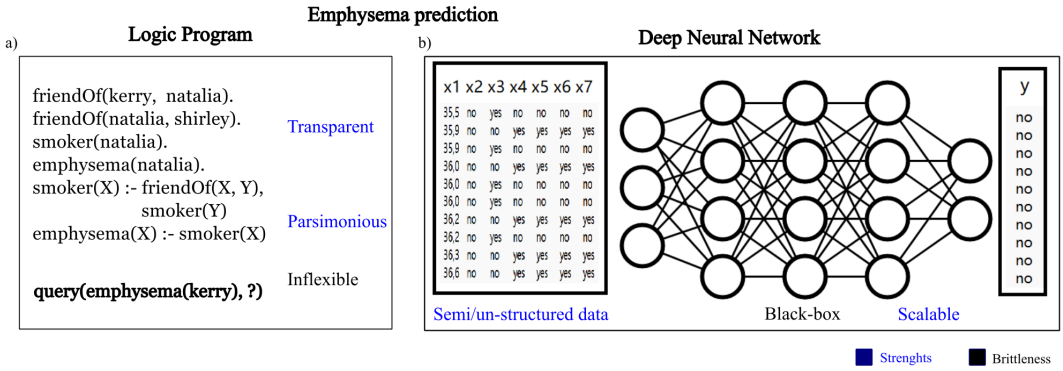


Figure 2.1: Strengths and weaknesses of both models. Notice that in $a)$, the logic program is transparent but written exclusively for the task of querying whether the atom "Kerry has emphysema" holds. In $b)$, the neural network is able to learn a desired input-output mapping, where input is represented more thoroughly by measuring a wider spectrum of features to characterize an individual. It is scalable and can capture complex patterns in the input distribution, but this is at the cost of being a black-box and lack of parsimony.

distribution could be, by chance, due to most training samples being measured from young adolescents who had access to wearable sensors, while the testing distribution also accounted for older adults with low-quality sensors.

Given this brief example, we note that due to this paradigm shift in modeling with respect to symbolic models, challenges tied to fitting neural networks are different to those of their counterparts. For example, given that neural networks embed knowledge in their parameters in a distributed manner, they become difficult to interpret and discern, even though they are easily scalable. It is outside the control of the researcher to control what the DNN is learning, and there is no mathematical guarantee to ensure the model is not picking-up spurious correlations in the data or statistical artifacts. Furthermore, tangent to their black-box architectures includes their unreliability in generalizing to out-of-distribution data, which gets exacerbated as it is also difficult to diagnose the source of an invalid output. For example, in the above scenario, the DNN could classify a newborn baby (of age 0) who never coughed or presented any symptom, as having emphysema. Such prediction would be paired with no insight into why.

There *are* attempts at mitigating such black-box behavior by back-tracing the sequence of activations that correlate with certain input-output pairs, or performing dimensionality reduction techniques on the weights $\mathcal{W}$ to find semantically meaningful representations [Barredo Arrieta et al., 2020]. While we acknowledge their importance, the focus of our work explores NeSy AI as alternative to address the black-box nature of deep learning and expand its capabilities, where current explainability methods can work in tandem with NeSy AI.

Chapter 3

Unifying Framework over Neuro-Symbolic Models and Benchmarks

3.0.1 Taxonomy of Neuro-Symbolic Models

The idea of integrating symbolic models with neural networks isn't novel. Views date as far as prior to the 21st century, such as how [Minsky, 1990] argued that useful AI systems should be "built by combining diverse components, some connectionists, and others symbolic". In recent years, attention revolving hybrid architectures rapidly gained traction as a response to the discussed brittleness of deep neural networks. This is evidenced by how the topic of NeSy AI has been discussed in major world conferences and by influential researchers [Harmelen and Teije, 2019, Garcez and Lamb, 2020]. A myriad of architectures has been proposed, with examples including neural-driven derivation of rules [Marra and Kuželka, 2020, Marra et al., 2020], neural driven knowledge graph search as opposed to uninformative searches through breadth-first or depth first search and setting constraints through expert rules in order to guarantee the soundness of outputs of multi-label classification and prediction models [Xu et al., 2018a, Ahmed et al., 2022a]. There is also work on integrating probabilistic logic programming to provide a structured representation for deep generative modelling [Misino et al., 2022]. Generally, on one hand, such constraints can address the theoretical concerns of whether a model can produce sound outputs given training data. On the other hand, the clear semantics of a NeSy model addresses concerns of transparency and trust, which contrasts with black-box deep learning models.

Henry Kautz, at the Third AAAI Conference, was one of first researchers who proposed a highly abstract, quinary classification system to categorize different NeSy architectures proposed [Saker et al., 2021]. Within it, a *sequential* architecture is of the form *symbolic Neuro symbolic*, mainly referring to models which pre-process non-numerical input into a matrix embedding to a neural network, outputting a non-numerical result. A nested architecture refers to either a Symbolic[Neuro] or Neuro[Symbolic] model where the parenthesis is used to denote the dominant component in the model. For example, Symbolic[Neuro] would be used to classify models where there is a dom-

inant reasoning system where the neural component simply guides internal decisions, such as AlphaGo. A *cooperative* architecture of the form Neuro→Symbolic consists of a neural network which outputs an intermediate result that is passed to a symbolic reasoner which solves a complementary task. For example, in visual question answering, the neural component would be in charge of object detection and the symbolic component would reason over this state space. A *compiled* system of the form Neuro : (symbolic), which refers to an approach where symbolic rules are "compiled" away during training of the neural network. Here, knowledge could be compiled into either the network's weights or into the loss function. As a canonical example of this class, they cite the work of [Lample and Charton, 2019] where they build a model able to recognize the compositional structure of mathematical expressions to solve differentiation [Hamilton et al., 2022].

[De Raedt et al., 2020] propose a more intricate system by categorizing models based on seven dimensions, mainly focusing on the structural aspects of assembling NeSy models. First, a NeSy model could be distinguished as either being represented as a directed graphical model (allowing for causal interpretation as in Bayesian Networks), or an undirected one as a Markov Network. Second, they examine what are the dominant components of the model, whether it's a probabilistic model, neural network or a rules-based model. Thirdly, the semantics of the logic integrated into NeSy models can be identified into either propositional, relational, first-order or logic programming. Functionally, they identify whether the inference engine of the NeSy model could either be model-theoretic or proof-theoretic. This refers to the difference between models which use forward chaining for forming new clauses given rules until some terminal condition is reached and those which employ backward chaining to find the set of applicable rules that can be used to prove a clause. They also propose identifying whether the model is mainly learning the weight parameters of the neural network, or learning the logical clauses of the symbolic program. Finally, a NeSy model could be divided into those which use logical clauses to represent input, those which use vectors or both. While this taxonomy provided by [De Raedt et al., 2020] is more thorough compared to Kautz's, it is more controversial as it may result in more contentious decisions, given that it is more challenging to determine whether a NeSy model should be assigned to a particular class, and whether the division among classes is sound. For example, there is a lot of overlap when it comes to determining the semantics of a NeSy model as knowing it uses first-order logic enables us to know that it represent entities using clauses (logical predicates), which is symbols in dimension 6. Additionally, when being implemented in code, first order logic is a form of relational logic as all variables are assumed to be universally quantified. Furthermore, all NeSy models employ a neural network and engages in parameter learning, making this dimension redundant.

We try to combine the generic quality of Kautz's framework with the granular characteristics outlined by [De Raedt et al., 2020]. In doing so, we aim in building a classification system of neuro-symbolic architectures which strikes the balance between abstraction and specificity. Drawing inspiration from Zoubing Gahramani's cube for categorizing probabilistic models, we propose the following general framework to understand how neuro-symbolic models have been assembled. Our framework not only tries to embody a wide spectrum of models, but also allows the reader to judge first

Prior work on building a taxonomy of NeSy models

	Dimension 1 (D)irected (U)ndirected	Dimension 2 (G)rounding (P)roofs	Dimension 3 (L)ogic (P)robability (N)eural	Dimension 4 (L)ogic (P)robability (F)uzzy	Dimension 5 (P)arameter (S)tructure	Dimension 6 (S)ymbols (Sub)symbols	Dimension 7 (P)ropositional (R)elational (FOL) (LP)
∂ILP [Evans and Grefenstette, 2018]	D	P	L+N	L	P	S	R
DeepProbLog [Manhaeve *et al.*, 2018]	D	P	L+P+N	P	P	S+Sub	LP
DiffLog [Si *et al.*, 2019]	D	P	L+N	L	P+S	S	R
LRNN [Šourek *et al.*, 2018]	D	P	L+N	F	P+S	S+Sub	LP
LTN [Donadello *et al.*, 2017]	U	G	L+N	F	P	Sub	FOL
NeuralLP [Yang *et al.*, 2017]	D	G	L+N	L	P	S	R
NLM [Dong *et al.*, 2019]	D	G	L+N	L	P+S	S	R
NLProlog [Weber *et al.*, 2019]	D	P	L+P+N	P	P+S	S+Sub	LP
NMLN [Marra and Kuželka, 2019]	U	G	L+P+N	P	P+S	S+Sub	FOL
NTP [Rocktäschel and Riedel, 2017]	D	P	L+N	L	P+S	S+Sub	R
RNM [Marra *et al.*, 2020]	U	G	L+P+N	P	P	S+Sub	FOL
SL [Xu *et al.*, 2018]	U	G	L+P+N	P	P	S+Sub	P
SBR [Diligenti *et al.*, 2017]	U	G	L+N	F	P	Sub	FOL
Tensorlog [Cohen *et al.*, 2017]	D	P	L+N	P	P	S+Sub	R

Figure 3.1: Highly granular classification system which can account for a plethora of NeSy models. Whilst highly informative, it is equally contentious given disparities in how authors can interpret what each dimension refer to. For example, dimension 3 and 5 are highly correlated and redundant with respect to one another as NeSy models with a neural network are bound to have tuneable parameters. Table extracted from [De Raedt et al., 2020]

hand the capabilities of some NeSy architectures.

Our framework has a tri-dimensional coordinate system which describes how displacement along an axis changes a significant component of the represented NeSy model. In the leftmost section of the X-axis are models which perform heavy reasoning over clauses, such as probabilistic inference, and light learning, such as parameter estimation [Yu et al., 2022]. The dominant component is encircled with '{}', e.g., for NeSy architectures with a dominant reasoning component, they are written as $Ne\{Sy\}$. As we move rightward along this axis, we encounter NeSy models which consist of a primary neural component and mainly engages in parameter learning, i.e., $\{Ne\}Sy$.

We leave "Sy" as a placeholder, which along the Z axis will take on values to represent the type of logic employed by the model. As we descend, the expressiveness of the logic increases, so we either substitute with propositional, fuzzy, first order or a higher order logic. Lastly, the Y-axis denotes whether the neural and symbolic components are connected via a nested architecture, compiled system or another way. We do not consider Henry Kautz's categories of "sequential" or "cooperative" since we argue there is a lot of overlap with the other two. Namely, at a functional level, both compiled and nested architectures can be thought of as cooperative or sequential since both the symbolic and neural components exchange outputs during training and testing. As the literature on NeSy develops, there will be new ways of merging both neural and symbolic systems. Our framework would thus be capable to anticipate architectural variants that mix certain types of logic or prioritizes learning over reasoning or viceversa.

We proceed in describing the capabilities and examples belonging to each vertex. NeSy

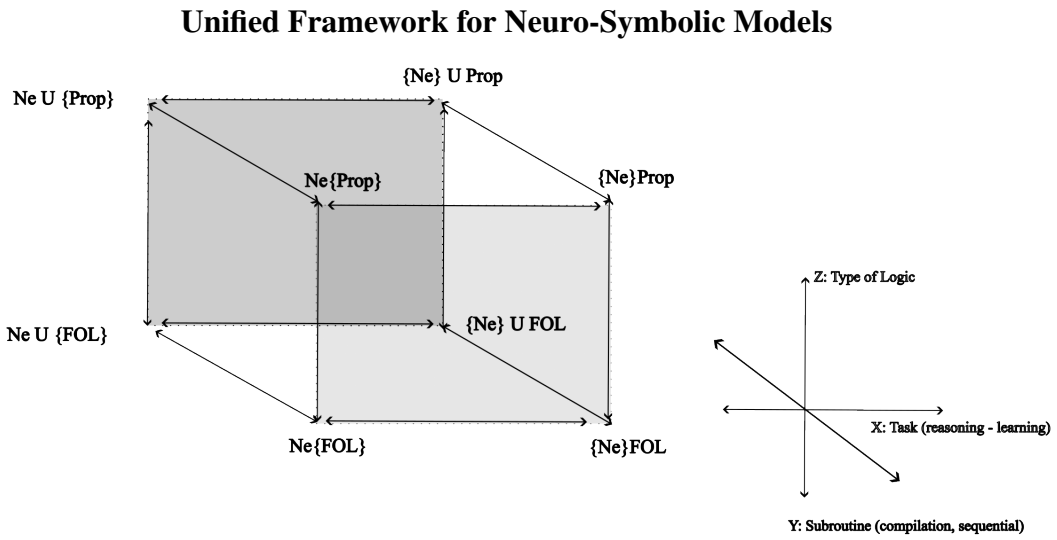


Figure 3.2: Non-exhaustive Schematic taxonomy of neuro-symbolic models

models can shift around these vertices depending on the particular task to be solved, such as whether it involves low-level perception and thus require a deep neural network like CNNs, or whether it is querying over a knowledge graph and a factor graph representation using Logic Markov Networks is enough.

- $\{[Ne]\}Sy$: It is used to refer to models which mainly perform function approximation and the logical module is mainly reserved to manipulate the probabilities of the network to solve a task. For example, in [Yi et al., 2018], the neural component is a CNN which learns to output a representation of the input image, which resembles a SQL-table where rows are the objects appearing in the scene and columns are the attributes describing each object. The symbolic component is a SQL-query executor that queries this representation. It is worth noting that in such architectures, the logical component itself is not differentiable, but the NeSy architecture as a whole is end-end differentiable. Henry Kautz would also argue that deep reinforcement learning models would also fall within this category, exemplifying AlphaGo, which is composed of a deep neural network and a non-differentiable Monte-Carlo search tree. The outputs of the neural component are used to passed to the search tree to obtain an output. Whilst the symbolic component of a NeSy model has been in the past also associated with Good-Old-Fashioned-AI (GOFAI) techniques like tree search, transversal algorithms or game theory, in our work we restrict ourselves to associating the symbolic component to the manipulation of logical predicates.

- $Ne\{[Sy]\}$: Here, we refer to NeSy architectures which have a dominant reasoning component consisting of a probabilistic logic program. Given a prescribed logic program consisting of probabilistic clauses, the probability distribution of each of them are estimated by neural networks. Examples include DeepProbLog [Manhaeve et al., 2021] or Scallop [Li et al., 2023]. Usually, in this kind of nested architecture the symbolic program's logical predicates are non-differentiable and retain their semantics, which is helpful for transparency. The

trade-off, however, is that they also inherit the difficulty of scaling up as pertaining to classical symbolic logic programs. As mentioned previously, there is ongoing effort to overcome this bottleneck through approximate inference [Van Krieken et al., 2022]. Other examples of this nested architecture involve Neural Markov Logic Networks [Marra and Kuželka, 2021], where standard feed-forward networks are employed to estimate the probability distributions of a factor graph. The parameter estimation of such networks are constrained by FOL prior knowledge. They also inherently suffer from the symbol grounding problem, i.e., on figuring out which symbols in the logical program corresponds to what desired concept. Such issue is not commonly discussed given that most benchmarks in which nested architectures are tested upon only rely on images composed of one object, such as an image containing only one digit such as in MNIST addition.

- $\{Ne\}$: + Sy: We include models which mainly perform learning by incorporating inductive biases, i.e., knowledge of the engineer embedded which influence how they process input and adjust parameters. These inductive biases do not follow the syntax of logical predicates, but are nonetheless implicit to the model architecture, and hence we denote them as compiled. The symbolic component is also differentiable but play a minor role such as specifying soft or hard constraints regarding plausible range of output values. Examples include Semantic Loss Functions used for multi-label classification and Semantic Probabilistic Layers used for predicting the shortest path over a input map. We note that because these models lack a substantial symbolic component, they are unable to process symbolic input like logic programs.

- Ne : $\{Sy\}$: We incorporate here models which reason over a logic program, but instead uses differentiable logical predicates, such as by adopting fuzzy logic. We include models like Logic Tensor Networks [Donadello et al., 2017] and Lifted Relational Networks [Šourek et al., 2015], the latter which uses "real logic" to refer to differentiable FOL, that is, Fuzzy FOL. Depending on the model architecture and chosen abstraction of the problem, compiling logic can also help shape informative search over graphs structures as opposed to uninformative search methods which are non-tractable as the graph scales up. This also helps scaling the model during inference at the expense of losing the semantics of logic predicates.

While our framework can be interpreted as a discretized space where at each vertex particular kinds of models reside, our framework is better understood as allowing models to be positioned in a continuum along the edge. For instance, models like DeepProbLog can reside between $\{[Ne]\}Sy$ and $Ne\{[Sy]\}$. This is because the user can define only a probabilistic logic program for it to perform approximate inference, such as solving program induction for sorting arrays. But, DeepProbLog also engages in parameter estimation such in its MNIST Addition Task where it learns which digits sum to certain output number without needing the corresponding digits to be labeled during training [Manhaeve et al., 2021]. Furthermore, we point out that our framework could also be used to compose unknown architectures by extrapolating the corresponding axes.

3.0.2 Unified Framework for Benchmarks

Tangent to the plethora of models proposed is the panorama of benchmarks used to test each of those models. We surveyed the vast literature and as expected, it is common that successes reported by different research teams for the models proposed are disparate to one another due to either unrelated benchmarks [Hamilton et al., 2022], as well as intrinsic differences in the tasks completed by each model proposed. We provide a more detailed survey in the Appendix A.

A very common benchmark we have surveyed and publicly available is CLEVR by [Johnson et al., 2017] (Figure 3.3), which is a visual question and answering (VQA) task which diagnoses whether a model can reason over a set of objects (see Table A.2). CLEVR is designed to be challenging by controlling for certain aspects of object-centrism by introducing several confounding objects in an image, each of varying rotation, position, background illumination, color, shape, material and size. Given a input scene which consists of maximum 10 objects with varying attributes, a model is asked questions such as "how many red spheres are there?" and "what is the shape of the object next to the blue cube?". Such questions fall within 4 categories:

- **Query attribute**, which asks for an object's attribute, such as 'what color is the sphere?'
- **Counting**, such as asking 'how many red cubes are there?'
- **Existence**, exemplified by asking "are there any cubes to the right of the red sphere?"
- **Compare integer**, which asks questions like "are there fewer cubes than red things?".

[Mao et al., 2019] report a neuro-symbolic concept learner, *NeHOL*[1] where they report state-of-the-art prediction accuracy despite holding out 90% of the training data, a form of out-of-distribution generalization from small data. There are several variations of CLEVR, such as CLEVR-Hans3 proposed by [Stammer et al., 2021a] and tested by their proposed NeSy model, as well as αInductive Logic Programming (ILP) by [Shindo et al., 2023], a neurosymbolic forward reasoner. Another variant is CLEVRER by [Yi et al., 2020] where instead of images, videos of colliding objects are shown. Similar to before, a model is challenged to answer different types of queries over these events, with the addition that counterfactual queries can be asked about collisions which are not shown in the sequence of frames.

Other NeSy models have been tested on reasoning over knowledge graphs, completing tasks such as multi-hop reasoning (see Figure 3.5) for knowledge base completion by employing datasets like MedHop, WikiHop and babi [Weber et al., 2019, Marra and Kuželka, 2021]. In such cases, we note how evaluation metrics slightly differ to metrics such as prediction/classification accuracy as we also include satisfiability, and completion time. The former refers to whether the model's arrived fact doesn't contradict existing knowledge

[1]Their NeSy model synthesizes programs applied over an embedding of the input to answer a query on the image. Here we describe programs as a form of HOL, where we point the reader to [Van Emden and Kowalski, 1976] for a discussion on this

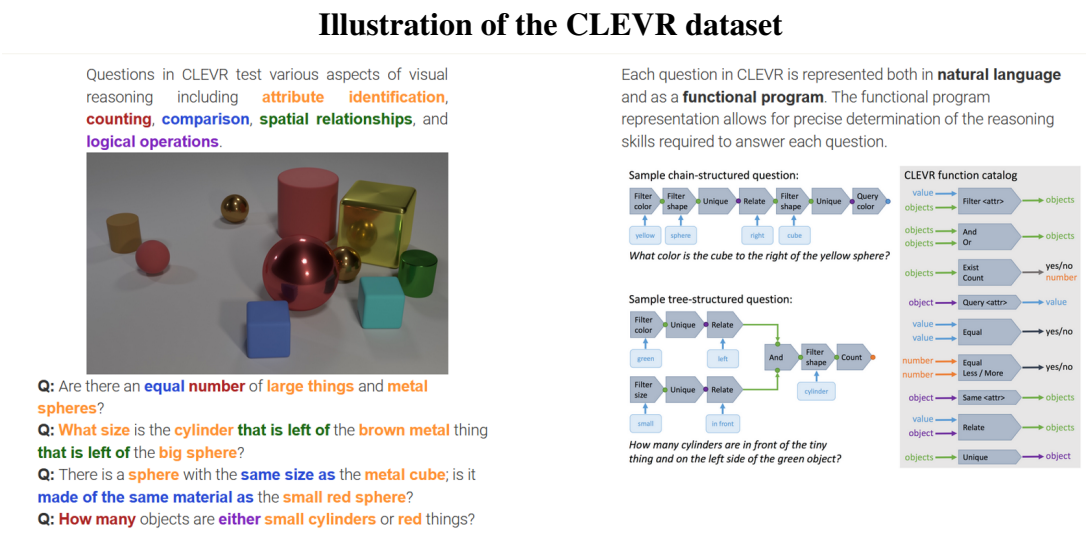


Figure 3.3: The CLEVR dataset is a VQA dataset which given an image and a query, a model is challenged to reason over the image in order to answer the query. This dataset serves as an alternative to ImageNet as images from it control for object-centric bias. CLEVR is thoroughly annotated, and each image is paired with a knowledge graph detailing in a semi-structured format what are the qualities, where does each object appear, and what is the query-answer for it. A functional program on how to compute the query is also supplied.

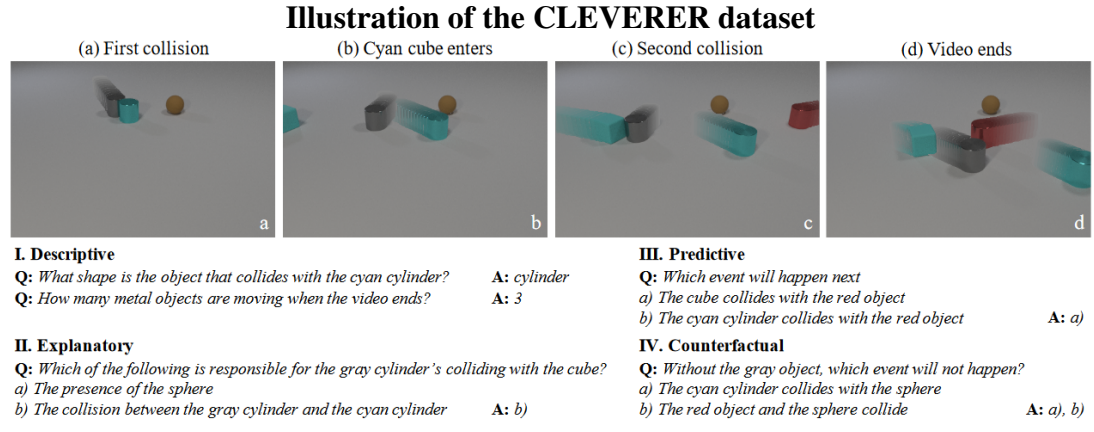


Figure 3.4: CLEVRER is a variant of CLEVR where frames of images are fed to the model in order to answer a wider variety of queries, including counterfactual queries over collision events which have not happened. Figure extracted from [Yi et al., 2020].

Example of a training datapoint in a Knowledge Graph for multi-hop reasoning

The Hanging Gardens, in **[Mumbai]**, also known as Pherozeshah Mehta Gardens, are terraced gardens … They provide sunset views over the **[Arabian Sea]** …

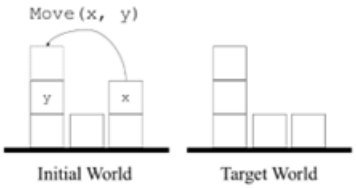
Mumbai (also known as Bombay, the official name until 1995) is the capital city of the Indian state of Maharashtra. It is the most populous city in **India** …

The **Arabian Sea** is a region of the northern Indian Ocean bounded on the north by **Pakistan** and **Iran**, on the west by northeastern **Somalia** and the Arabian Peninsula, and on the east by **India** …

Q: (Hanging gardens of Mumbai, country, ?)
Options: {Iran, **India**, Pakistan, Somalia, …}

Figure 3.5: During training, a model parses a knowledge graph which consists of a natural language prompt, a structured question and a given answer. In this example, the model is supplied knowledge that the Hanging Gardens are located in Mumbai, while Mumbai is located in India. We can therefore ask whether the Hanging Gardens are located in India because although this wasn't explicitly mentioned in the provided information, it can be reasoned over.

Small world reasoning task



$\texttt{On}(x, y)$	True if x is on y
$\texttt{IsGround}(x)$	True if x is the ground
$\texttt{Clear}(x)$	True if there is no block on x
$\texttt{Moveable}(x)$	$\neg\texttt{IsGround}(x) \wedge \texttt{Clear}(x)$
$\texttt{Placeable}(x)$	$\texttt{IsGround}(x) \vee \texttt{Clear}(x)$

Figure 3.6: Given a structured small world description which consists of logical predicates, a model is tasked with finding a sequence of steps that map intial state to target state.

contained in the knowledge graph, while the latter is a measure of the time complexity of the inference procedure.

Different research teams also propose models where they test task-completion over a prescribed logic program, and empirically evaluate their models over defined problems such as the Block's World Problem [Dong et al., 2019] (Figure 3.6), sorting arrays [Manhaeve et al., 2021] or determining the evenness of a digit [Evans and Grefenstette, 2018]. Performance metrics in this case are similar to those of knowledge graph reasoning: satisfiability and completion time. What is interesting from models completing these types of tasks is that the prescribed logic program to define a problem can be expressive enough to encompass problems like linear, logistic regression as well as unsupervised clustering, as shown by [Donadello et al., 2017].

Another novel benchmark is Procedurally Generated Matrices (PGMs) [Barrett et al., 2018], which is an abstract reasoning task where a model is tasked choosing an image, from multiple choices of candidates, that completes an implicit pattern present in a preceding sequence of panels (Figure 3.7). Here, the term 'abstract' refers to the absence of a query and lack of an explicit mention of the rule the sequence of generated images is following. Variants inspired by PGMs include RAVEN's progressive matrices.

The RAVEN dataset [Zhang et al., 2019] is a multiple choice, abstract-pattern completion problem with a similar setting as PGMs. Each training instance consists of panels, each panel's image has attributes has objects described in terms of type (triangle, square, pentagon, hexagon and circle), size (enumerated from 1 to 6), and color (enumerated from 1 to 10). Objects follow 4 types of rules:

- **Constant:** The attribute value does not change per row.
- **Progression:** The attribute value monotonically increases or decreases in a row by a value of 1 or 2.
- **Arithmetic:** The attribute values of the first two panels are either added or subtracted, yielding the attribute value of the third panel in the row.
- **Distribute three:** This rule involves the fact that three different values of an attribute appear in the three panels of every row (with distinct permutations of the values in different rows)

[Hersche et al., 2023] propose a neurosymbolic architecture that vectorize rules of logic and embeds them into a neural architecture to achieve new state-of-the art completion accuracy compared to purely deep learning alternatives. Their work, however, assumes the model knows a priori what are the implicit sequences followed by these PGMs.

[Chollet, 2018] propose the Abstract Reasoning Corpus that tests for analogy-making, a form of identifying implicit patterns in provided panel of images. The difference with PGMs lie in that a pattern is not chosen from a pool of candidates, but rather has to be generated by the model. Most importantly, there is no a priori specification of what rules or composition of them underlies the analogy. At the time of writing this thesis, we haven't identified an existing NeSy model achieving remarkable results in solving the ARC challenge, although there are attempts. For example, [Banburski et al., 2020] leverages Dreamcoder[Ellis et al., 2021], the program synthesis model, to only solve a portion of ARC tasks, such as those with a common theme of symmetric patterns. Despite challenging for the current landscape of deep learning and NeSy models, it has been studied that humans can easily achieve a pattern-completion accuracy exceeding 80%[Moskvichev et al., 2023]. Furthermore, the same authors propose a variant of ARC called ConceptARC to help the machine learning community diagnose a solution to the original challenge and assess the limitations of purely deep learning systems like GPT-4, which is reported to be unable to solve it.

Above is a brief description of some of the benchmarks that have been used to test different neurosymbolic reasoners, with a more detailed overview in A.2. It is evident that successes reported by different research teams can not be placed on a wider context due to essentially different challenges being solved, and different evaluation metrics pertaining to a specific dataset used to assess a NeSy model (see Figure 3.9). Rarely there is an overlap of different NeSy models being tested on a same benchmark, and as such it is difficult to chart out the capabilities of NeSy models as well as means to optimize them. Comparisons amongst models is challenging, which in turn prompted authors like [Garcez and Lamb, 2020] to conclude that "NeSy is in need of standard benchmarks and associated comprehensibility tests which could provide a fair comparative evaluation of different approaches".

Sample image of a procedurally generated matrix

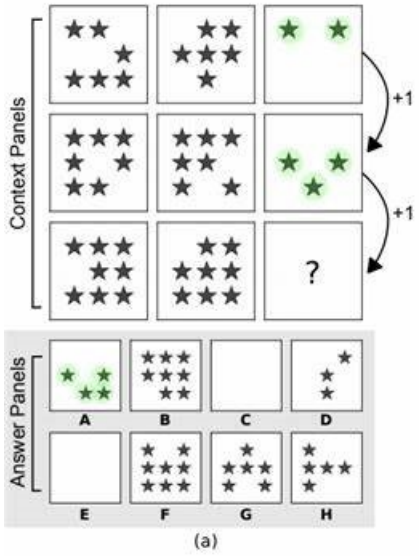


Figure 3.7: Given a panel of images, a model is tasked to complete the sequence. In the example shown, the number of starts increase by one across each column. The position of the starts are irrelevant when deciding what is next in the sequence. Figure cropped from [Barrett et al., 2018]

Our survey suggests that in order to design a common benchmark, we have to account for the different challenges and evaluation metrics that are used in the above datasets. This can be done in a straightforward way by assembling essentially distinct datasets into a benchmark suite. The benefits of devising this standard benchmark is that it allows current and future researchers to be able to systematically compare different methods under this common benchmark on whether they can achieve goals such as out-of-distribution generalization from small training data, interpretability of reasoning process [Hamilton et al., 2022], and computational time and memory complexities. In order to devise the common benchmark suite, we first designed an abstract framework useful for understanding what most, if not all, datasets are testing (Figure 3.10).

This abstraction aids us during our survey of the vast literature of datasets as we catalogue proposed datasets in terms of input/output requirements and sizes, computational challenges, and reasoning task (see Table A.2). As a result, we identify and propose the following quinary classification system to describe known reasoning tasks useful to test neurosymbolic models:

- **Object-centric Relational Reasoning** (often referred to as *visual question and answering*). According to Figure 3.10, or the blue arrow in Figure A.1, this would correspond to a curated dataset consisting of low-level input, logical predicates and/or natural language question having to be mapped to a fact or a natural language answer. In this reasoning task, a model is tasked to perform relational reasoning over unstructured data consisting of images. Challenges arise given the appearance of multiple confounding concepts in the image, such as

Sample image of a traning instance of the ARC challenge

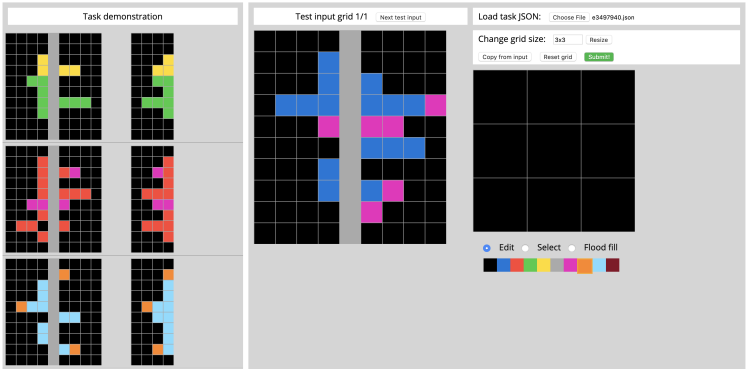


Figure 3.8: A guided-user interface depicting a training instance that consists of maybe 3-5 pairs of panels. Preceding pairs of panels are task demonstrations where the transition from left to right follows an implicit pattern, i.e., an ”analogy”. A model is tasked to complete the pattern for the last pair. In the example shown, the right image is reflected to the left. Image extracted from [Chollet, 2018]

Disparity of successes in the NeSy literature

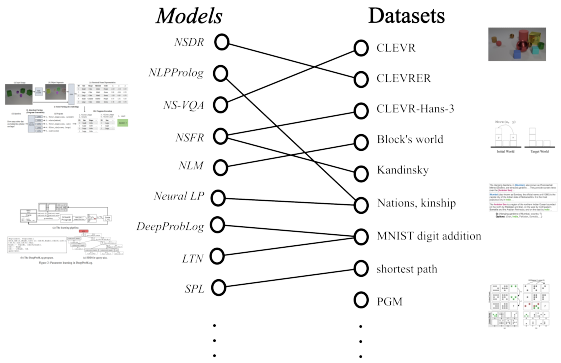


Figure 3.9: Depiction of disparate successes of NeSy architectures. Notice the rare overlap among models and datasets, making comprehensive comparisons challenging.

in CLEVR and its variations, thus assessing the model's ability of learning a disentangled representation of such signal.

- **Multi-hop Reasoning** over a knowledge base or graph, also known as knowledge-base completion or query answering (QA) over incomplete knowledge graphs (KGs). We adopt the definition of KG as a semi-structured dataset consisting of a collection of entities and relationships among them. In Figure 3.10, or black-fill arrow in Figure A.1, this would correspond to a curated knowledge base (structured or semi-structured with natural language data) having to be mapped to a natural language answer or logical fact. A NeSy model, mainly its symbolic component, is challenged to perform transitive reasoning to discover new knowledge given the KB to answer a user-given query. Example datasets include MedHop, WikiHop and Freebase.

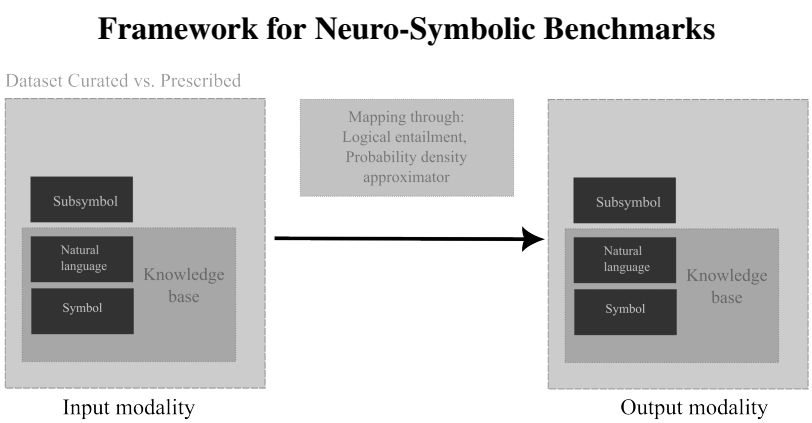


Figure 3.10: General framework for benchmarks of neuro-symbolic models. Essentially, benchmarks test whether a model can successfully map an input to a desired output, with differences arising from the modalities of input/output and the process of transforming it, whether it's through parameterized functional approximation, logical inference or both. Annotated version is found in Appendix Figure A.1

- **Task-driven Reasoning**: In Figure 3.10 or red arrow in Figure A.1, this would correspond to a small-scale, prescribed knowledge base (a logic program) having to be mapped to a knowledge base through logical inference. These kinds of tasks often involve researches to define a small world. NeSy models are then tasked to reach certain specified goal condition whilst ensuring satisfiability of the final proof. Challenges tied to this form of reasoning include prevention of time overflow when searching for rules, as well as the soundness of the final proof. It is noteworthy the flexibility of prescribing the logic program has allowed researchers to convert traditional learning tasks such as regression, classification and unsupervised clustering into reasoning ones [Donadello et al., 2017]. Other examples include the classical Block's World problem and sorting arrays.
- **Object-centric Abstract Reasoning**: In Figure 3.10, or green arrow in Figure A.1, this would correspond to a mapping between a curated dataset of images to images. Under this branch, we incorporate reasoning datasets like PGMs and its variants such as RAVEN's progressive matrices. We also refer to ARC[Chollet, 2018] and variants like ConceptARC [Moskvichev et al., 2023].
- **Object-centric Counterfactual Reasoning**: In Figure 3.10, or black-dotted arrow in Figure A.1, this would correspond to the same mapping procedure as Object-centric Relational Reasoning. However, counterfactual queries pose hypothetical questions about objects or events which haven't happened, i.e., "what if" scenarios. In contrast, relational queries ask about objects which are present in the image. A canonical example is from the CLEVRER, where counterfactual queries can take the form "Without a gray object, which (collision) event will not happen?"[Yi et al., 2020].

We note that although we partition reasoning tasks into five categories, boundaries are not definite and it is often the case that a benchmark combine different tasks of reasoning tasks. For example, the model DeepProbLog by [Manhaeve et al., 2021] both

prescribe a program for task-driven reasoning to define a addition over digits, where digits images from the MNIST dataset, altogether a form of object-centric relational reasoning. Table 3.2 describes how our framework can embody a wide spectrum of benchmarks and detail their challenging aspects, which serves as more compact summary for Table A.2.

Given our classification system and framework, we chart out how our benchmark suite should be designed and aim to build it in a way to test each reasoning task, with their respective challenges. As noted previously, because NeSy AI is the assembly of different components, it inherits challenges unique to each one of them. On the neural side, it is common for benchmarks to assess classification or prediction accuracy pre- and post-training, evaluating the model's ability to maximize training accuracy and minimize generalization error. On the symbolic side, however, challenges arise during the inference phase, i.e., whether the arrived proof actually satisfies the query and whether it can be done in a tractable amount of time[2]. A common benchmark, hence, must account for different evaluation metrics to assess both the neural and symbolic component.

[2]In technical terms, symbolic models which perform logic programming and proves a query, or probabilistic logic programming where the probability of a query is calculated, performing exact inference becomes a non-polynomial (NP task). It is therefore a challenge for symbolic models to perform approximate inference in order to reduce computational complexity

NeSy models	Neural components	Symbolic components	Categorization
DreamCoder	Program recognition module	Program synthesis	{[Ne]}HOL or {[Ne]}{[HOL]}
NeSy-Visual Question Answering (VQA)	Mask RCNN for image recognition + LSTM to parse questions	SQL-like Query executor	{[Ne]}HOL
αILP	Pretrained Slot Attention	Differentiable forward reasoner	Ne{[FOL]}
DeepProbLog	User-specified neural network	ProbLog using probabilistic circuits for scalable inference	Ne{[FOL]} or {[Ne]}{[FOL]}
Neural Markov Logic Networks (NMLM)	Standard feedforward neural networks to represent factor graph	Probabilistic inference	Ne{[FOL]}
Logic Tensor Networks	User-specified neural network	User-specified logic program in Real Logic	Ne : {Fuzzy FOL}
Relational Neural Machine	Neural Markov Logic Networks	Weighted probabilistic inference	Ne : {Fuzzy FOL}
Semantic Probabilistic Layer	CNN	Approximate inference using probabilistic circuit	{Ne} : Fuzzy Prop

Table 3.1: Classification of several NeSy architectures based on our taxonomy, serving as a compact summary for Table in Appendix. We notice that nested NeSy archictures mainly perform exact or approximate inference by employing probabilistic logic, where neural networks estimate such probabilities. NeSy architectures which compile logical predicates using fuzzy logic also perform probabilistic inference, as well as constrained learning. This serves as a summary for Appendix Table A.1

Catalogue of Benchmarks based on Five Major Reasoning Tasks

Nature of Task	Input-Output	Challenging Aspects	Examples
Object-Centric Relational Reasoning	Images and Query - Answer	Confounding concepts; out-of-distribution generalization; interpretability	CLEVR, Kandinsky Patterns, CLEVR-Hans
Multi-Hop Reasoning	KB and Query - Answer	Satisfiability; search efficiency	ChEMBL, Wiki-Hop, MedHop, babi
Task-Driven Reasoning	Logic program and Query - Answer	Satisfiability; search efficiency	Block's World, Sorting Arrays, Coin Ball
Object-Centric Abstract Reasoning	Image - Image	Patterns in images are implicit; interpretability	PGM
Counterfactual Reasoning	Image/KB and Query - Answer	Objects in the query are absent in dataset; interpretability	CLEVRER

Table 3.2: Proposed taxonomy to classify reasoning tasks which serve as a preamble to designing the common benchmark suite

Chapter 4

Common Benchmark Suite Design

4.1 SaSSY CLEVR Benchmark

While there is no common benchmark for testing NeSy models, there has been several ones proposed for deep learning architectures spanning over computer vision and Human language modelling. Recent years have seen interest over multimodal benchmarks, i.e., those which can test a model's capability to perform both visual perception and language understanding, with tasks such as VQA.

Inspired by this, we assemble a common benchmark suite for NeSy models by collecting existing benchmarks which test different types of reasoning and merge them. For example, a similar approach has been taken by [Singh et al., 2022] where in order to validate the strengths of a foundational vision-language model (FLAVA) they propose, they compare available deep learning candidates on a common compendium of different computer-vision and language benchmarks. This include challenges spanning from VQA, sentiment analysis, natural language inference, semantic similarity, among others. We take inspiration from this and build our benchmark suite by assembling together representative datasets of each kind of reasoning task we identified in Table 3.2.

The design of this benchmark suite mainly elaborates on CLEVR, as it is an overarching benchmark encompassing object-centric and knowledge graph reasoning by default. We name it the SaSSY-CLEVR (short for Symbolic and Sub-Symbolic CLEVR) common benchmark suite. We suggest the baseline used in this common benchmark to be a purely deep learning model in order to identify the improvements that NeSy methods can achieve. We note that all reasoning tasks expect the tested model to produce interpretable results. We proceed in detailing how each of the reasoning task is tested in our benchmark suite:

4.1.1 Object-centric reasoning

We recycle CLEVR-Hans3 proposed by Stammer et al. (2021). CLEVR-Hans3 is a variant of CLEVR in that there is no provided query and it is a ternary classification task. CLEVR-Hans3 is split into a training, validation and testing partition consisting

of 9000, 2250 and 2250 samples respectively. Class assignment is evenly balanced in each split, e.g., in the training set, each class is represented by 3000 images. A class 1 label is assigned when an image contains at least a "large cube and large cylinder", while class 2 images contain at least "a small metal cube and small sphere" and in class 3 there is at least "a large clue sphere and a small yellow sphere". These classes are mutually exclusive and images are carefully generated so as to ensure non-overlap. By default, CLEVR-Hans3 tests a model's robustness to generalize under visual confounders, i.e., during training and validation, only images with large gray cubes and large cylinders are shown as class 1, while during testing large cubes of varying color are introduced 4.1. Here, the color 'gray' is a confounding concept which a DNN may learn as a statistical artifact to decide class assignment, even though it is not a determinant.

4.1.2 Knowledge-Graph reasoning

CLEVR's scene annotations are essentially knowledge graphs encoding the relationships between objects, their attributes and spatial relations with respect to each other. Therefore, we employ open-source software[1] to generate such scene graph annotations for CLEVR-Hans3 and test for knowledge graph reasoning in this augmented dataset. The scene annotations describes in detail each object appearing in the scene along with a natural language question and answer. There are 700 thousand natural language questions for training and 150 thousand questions for validation and testing. There are maximum 10 entities per question, each entity with varying amount of properties corresponding to its 3D coordinates, 2 possible materials and sizes, 3 shapes and 8 different colors. Relations can encode which object is next to one another or counting how many objects are there. The evaluation metric is measured by the correctness of answers given to test set queries and supplementary analysis can include inference time for answering. A model that can answer the test queries correctly in the shortest time is considered highly-performing.

4.1.3 Task-driven reasoning

We elaborate on the Block's World Problem where the underlying task is finding a program to map the initial state to a goal state. Similar to [Dong et al., 2019], we model the Block's World as a reinforcement learning problem. Different to the aforementioned authors, we integrate CLEVR-Hans3 to make the problem more challenging by introducing a a diverse state space $\mathcal{S}$ and more actions in $\mathcal{A}$.

Each state is a matrix $\mathbf{S} \in \mathbb{R}^{O \times D}$. These are object-centric representations of each image that can be automatically retrieved in CLEVR-Hans3's source code. Each state represents the maximum $O = 10$ objects appearing in the scene, each S_o described by $D = 19$ attributes that fall within 5 categories, and are indexed as follows: $D_{0:3} =$ 3D coordinates and presence of the object, $D_{4:6} = 3$ possible shapes (sphere, cube, cylinder), $D_{7:8} = 2$ sizes (large, small), $D_{9:10} = 2$ materials (rubber, metal) and $D_{11:18} =$ 8 colours (cyan, blue, yellow, purple, red, green, grey and brown). We randomly

[1] https://github.com/facebookresearch/clevr-dataset-gen

retrieve 2 images of the training set (Figure 4.2) and set the former as initial state, and latter as goal state. The agent has the following actions available to transform the initial state as desired:

- **move:** takes a state, object ID, and (x, y, z) coordinates as input. Outputs a new state with the specified object with updated coordinates.
- **change color:** takes a state, object ID, and color as input. Outputs a new state with the specified object with updated color.
- **change shape:** takes a state, object ID, and shape as input. Outputs a new state with the specified object with updated shape.
- **change size:** takes a state, object ID, and size as input. Outputs a new state with the specified object with updated size.
- **change material:** takes a state, object ID, and material as input. Outputs a new state with the specified object with updated material.
- **add:** takes a state and object ID. Outputs a new state with a new object.
  - A new object is instantiated at the coordinates (0.5875, 0.5875, 0.5875) as a large, rubber, cyan sphere by default.
  - Such object can only be instantiated at an empty slot.
- **remove:** takes a state and object ID. Outputs a new state with the object removed.

Thus, this reinforcement problem has an overall goal of estimating the optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that can output the sequence of (valid) actions that map the the initial state to final state. The reward space $\mathcal{R}$ consists of positive reward, e.g. $+1$, for reaching the final state, negative (e.g. -10 for illegal actions like adding an object in the position of an existing one and 0 otherwise. To avoid the trivial solution whereby all objects are removed and objects in the final state are added, the minimum description length (MDL) of the solution program is accounted as part of the reward. Evaluation metrics include satisfiability of the found program, and supplementary analysis can include inference time over the number of predicates. Interpretability is assessed by the rules found by the model.

The above setting is subject to change in future work. For example, the problem's complexity can increase by integrating probability distributions to the actions to, such as changing color of an object only having a 90% probability of success. More features such as the relative position between objects can be added to increase the dimensionality of $\mathcal{S}$.

4.1.4 Object-centric abstract reasoning

We draw inspiration from PGMs and design IQ-style patterns of images generated with OpenCV following simple rules of counting, positioning and sizing. Each training instance consists of 4 panels of images: a sequence of 3 preceding panels showing objects moving with an implicit pattern followed by an empty panel.

Influenced by the Abstract Reasoning Corpus [Chollet, 2018], and unlike prior work like PGMs which focuses on providing multiple choice answers, our task is designed for the model to generate the panel that follows the abstract pattern observed in previously. This choice is to avoid commonly reported problems in statsitical learning where models choose the correct answer often based on exploiting unanticipated artifacts in the training dataset. For example, a correct panel is chosen not because it completes a pattern, but rather because by coincidence most correct answers during training had a large number of objects appearing in it (for a more elaborate argument please see [Zhang et al., 2023]).

We consider each image as having at most 6 objects, each taking at one of 2 possible shapes (squares and circles), and 6 possible colors (red, blue, green, orange, yellow, gray). This follows the domain setting as the Sort-of-CLEVR dataset by [Santoro et al., 2017] (except that we do not generate images where objects are randomly placed along with relational or non-relational queries). Implict patterns followed by images synthetized include:

- **arithmetic progression:** each subsequent panel has an additional object. It could be one or two more objects.
- **(counter-)clockwise rotation:** an object appears following a (counter-)clockwise movement.
- **alternating size:** an object alternates between small and large sizes for each subsequent panel.
- **composition of implicit patterns:** to increase the diversity of patterns, we compose them to generate novel variants. For example, an arithmetic, clockwise rotation consists of objects appearing in clockwise manner, each panel having said object's number 4.3.

Furthermore, we draw inspiration from CLEVR-Hans3 [Stammer et al., 2021a] by designing the abstract reasoning task to be more challenging by introducing cofounders during testing. For example, given the example in Figure 4.3, the model learns to complete the pattern seeing only blue squares. During testing, the confounder is introduced and the center square can have varying color for the 3 preceding panels, e.g. green. If the model understood the pattern, then it should disentangle color from the clockwise-increase pattern.

We generate 3000 training instances, 750 validation, and 750 testing instances. These are generated with different patterns aforementioned (and composition of them) along with confounders. Each image has an underlying matrix representation as in [Chollet, 2018]. Our evaluation metric is an average of the pixel-wise similarity of the generated image to the ground-truth.

4.1.5 Counter-factual reasoning

CLEVR's queries do not originally test for counterfactual reasoning, referring to queries which ask for situations that are not explicitly shown in the scene nor its scene graph annotations, e.g., "how many red objects are there if the sphere is removed?". Rather,

Query type	**Original**	**Counterfactual**
Query attribute	What color is the thing right of the red sphere?	What color is the thing right of the red sphere if the blue cube is removed?
Counting	How many red cubes are there?	How many objects will there be if the blue metal cube is removed?
Existence	Are there any cubes to the right of the red thing?	Will there be any cubes to the right of the red thing if the blue cube is removed?
Compare integer	Are there fewer cubes than red things?	Will there be fewer cubes than red things if the red cube is removed?

Table 4.1: Depiction of counterfactual queries corresponding to each query type from CLEVR's question generator.

it focuses only on relational or declarative reasoning, e.g., "how many red objects are there?". To compensate for such, we propose imputing counter-factual queries to the existing knowledge graphs of CLEVR-Hans3 retrieved from Section 4.1.2. For every query type that is generated with the question generator, there is a counterfactual counterpart as depicted in Table

As previously done, here we employ the question generator by [Wong et al., 2021][2] and apply it to CLEVR-Hans3. Their expanded question templates satisfy our requirements:

- **remove:** queries that ask about scenes if a subset of objects is removed, e.g, "if you removed the green cubes, how many cubes would be left?".
- **transform:** queries ask about hypothetical scenes where a subset of objects is modified, e.g, "if all of the large yellow rubber things became gray spheres, how many gray spheres would there be?"

For a NeSy model to be able to answer such queries accurately entails it not only understands the existing relationships of objects present, but also reason about hypothetical situations, which may exponentially increase the state space to account for. The evaluation metric is similar to that of reasoning over knowledge graphs, namely correctness of the answer and tractability of obtaining it.

[2] https://github.com/ellisk42/ec/tree/icml_2021_supplement/data/clevr/questions

Table 4.2: Comparison among datasets

	Object-centric	Knowledge Graph	Task-Driven	Abstract	Counterfactual
ARC [Chollet, 2018]	×			×	×
PGM[Barrett et al., 2018]	×			×	
Block's World[Dong et al., 2019]			×		
CLEVR[Johnson et al., 2017]	×	×			
CLEVERER[Yi et al., 2020]	×	×			×
Extended CLEVR [Wong et al., 2021]	×	×			×
Sort-of-CLEVR [Santoro et al., 2017]	×				
CLEVR-Hans [Stammer et al., 2021a]	×				
SASSY-CLEVR (Ours)	×	×	×	×	×

Table 4.3: A cross represents whether the benchmark accounts for this type of reasoning as argued in Table 3.2. We incorporate here the datasets that serve as main inspiration for SASSY-CLEVR

4.2 Implications of Successful Completion of our Benchmark

As reiterated previously, proposing this standard benchmarks serves to fill in a concerning loophole within the research community of NeSy reasoners, which is being able to comprehensively contrast different methods in order to obtain a accurate grasp of their potential, their weaknesses and how to optimize them.

Because we suggest at least using a deep neural network as a baseline, we expect that completing our common benchmark using a NeSy approach also serves to highlight the strengths of the NeSy methods assessed. On one hand, completing SaSSY-CLEVR is not solely an endeavor of achieving high scores in the evaluation metrics of each reasoning task. It is also equally important outlining the reasoning procedure that leads to such high accuracy, such as outputting the logic program which helped map the input to the desired output. Being able to complete all the reasoning tasks also entails that the NeSy method can process multiple input modalities such as knowledge graphs and logic programs on top of unstructured data like image tensors, or word tokens. In Table 4.3, we compare our benchmark suite with existing benchmarks in order to highlight the multimodal, heterogeneity in reasoning tasks our benchmark addresses. We note that our SaSSY-CLEVR simply makes incremental updates to existing benchmarks by assembling different datasets together. For example, we simply adapt CLEVERER's counterfactual queries to knowledge graphs of CLEVR. We also simply adapt Sort-of-CLEVR so that it can test abstract reasoning, rather than object-centric relational reasoning as the original authors intended.

Our benchmark mainly focuses on reasoning tasks, i.e, mostly testing NeSy models which have a dominant symbolic component and perform probabilistic inference where neural networks act as probability estimators for the predicates. For exaample, our benchmark does not account for learning tasks involving constrained optimization. Examples falling into this category involve finding the shortest path given an input image of a map, or a multilabel classification task known as preference learning. Other classical learning tasks include linear or logistic regression along with unsupervised clustering. We reiterate, however, that given a thoroughly specified logic program

as done in Logic Tensor Networks by [Donadello et al., 2017], these learning tasks can be framed as task-driven reasoning tasks. Thus, a model that can complete the task-driven reasoning portion of SASSY-CLEVR can also complete these traditional learning problems.

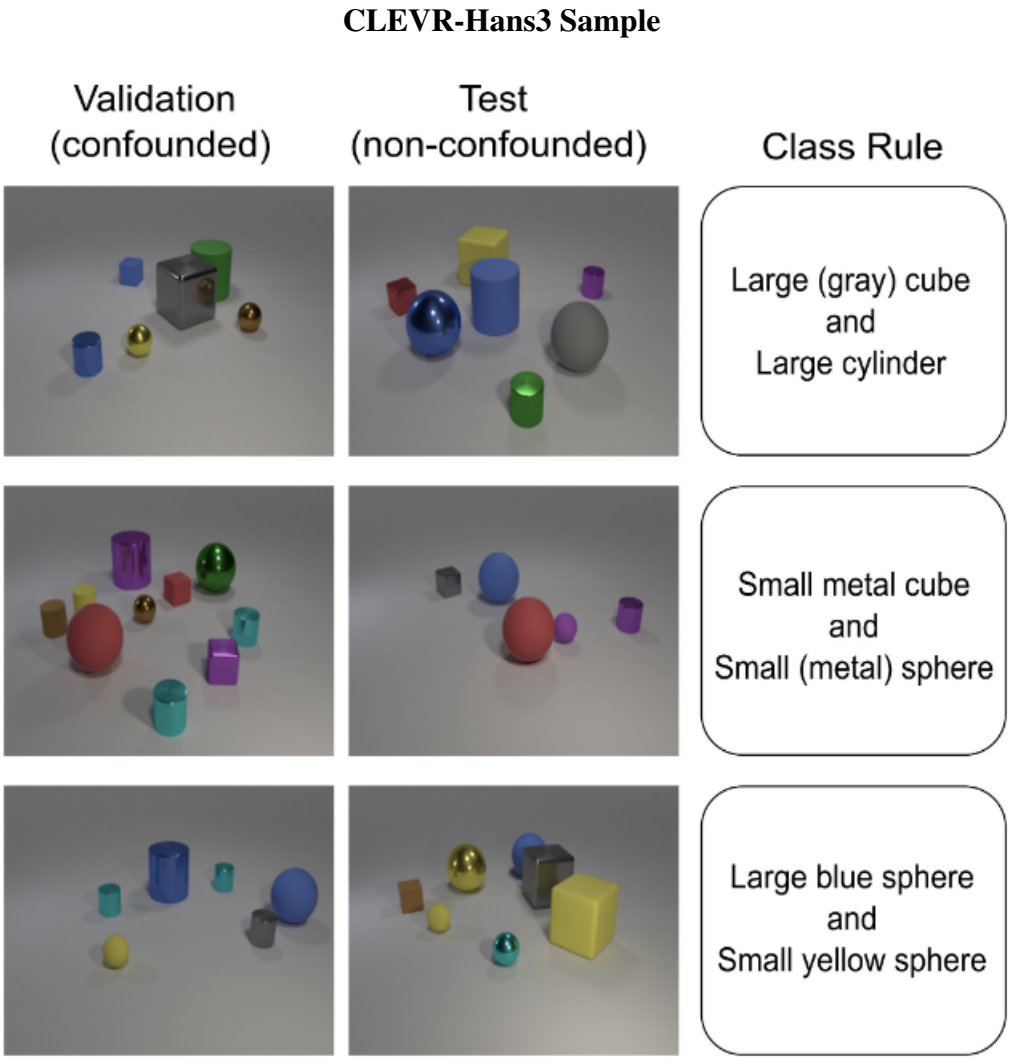


Figure 4.1: Illustration of CLEVR-Hans3. Class 1 and 2 contain confounders enclosed in parentheses, i.e., during training and validation phases, only images with large gray cubes and small metal spheres will be shown with their respective positive labels. During testing, these confounders are relaxed. Each image is coupled with a knowledge graph detailing aspects of each object. Figure extracted from [Stammer et al., 2021a]. The evaluation metric is measured by classification accuracy on the test set.

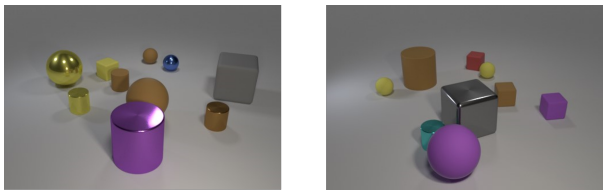


Figure 4.2: Two image samples from the training set are randomly sampled. Their obejct-centric representation is computed to obtain the start and goal state.

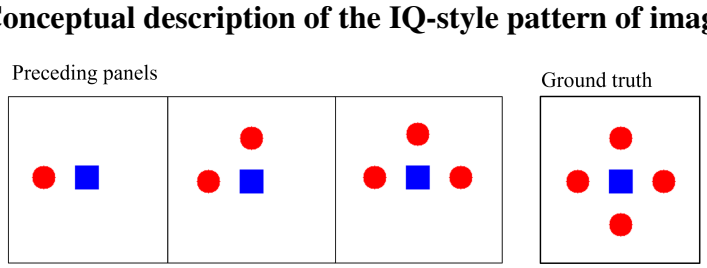


Figure 4.3: A sequence of three images following an implicit pattern is fed into the model and it has to predict the image that most likely completes the pattern. In this case, the pattern is a clockwise increase of red circles

Chapter 5

Experiments

5.1 Methodology

Our benchmark is challenging due to its heterogeneity of tasks addressed. Owed to this, for this thesis we only focus on the task of Object-Centric Relational Reasoning, i.e., CLEVR-Hans3. We mainly elaborate on prior work already proposing architectures solving CLEVR-Hans3, namely a NeSy architecture called Concept Learner proposed by [Stammer et al., 2021a] and a model which performs Inductive Logic Programming (ILP) αILP by [Shindo et al., 2023]. Both models share a common neural component: a Slot Attention pretrained on CLEVR's scene graph annotations to output an object-centric representation of an input image. This is a tensor $\mathbf{Z} \in \mathbf{R}^{B \times O \times D}$, where B is the batch size, O is 10 the maximum amount of objects that can appear in a scene, and D is 19 corresponding to the attributes that describe each object as mentioned in subsection 4.1.3. Hence, a single slot represents an object identified in the image, and the associated 19-dimensional vector refers to an unnormalised probability distribution over the attributes of this object. In both NeSy architectures, the parameters of the Slot Attention are fixed.

In the Concept Learner [Stammer et al., 2021a], this object-centric representation is passed to a Set Transformer which during training learns which attributes to attend to in order to derive the class label (Figure 5.1). In αILP [Shindo et al., 2023], this representation is converted into probabilistic facts used to assess how useful is each of the clauses, found through top-k beam search, in predicting the class label. Clauses improve iteratively over training epochs (Figure 5.2). Candidate clauses are formed from the logic program consisting of the following binary probabilistic logic predicates:

- $in/(2,[obj,image])$: probability that one of the 10 possibles objects is in the image
- $shape/(2,[obj,shape])$: probability that the object is of possible shapes cube, cylinder or sphere
- $color/(2,[obj,color])$: probability that the object is one of the colors cyan, blue, yellow, purple, red, green, gray, brown

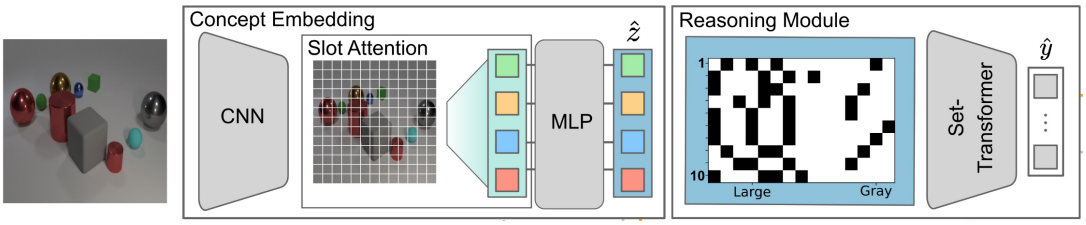


Figure 5.1: NeSy Pipeline. Given an input image, it is first passed to the Slot Attention which encodes it into a tensor which is an object-centric disentangled representation of it. The Set Transformer takes this representation and learns during training which attributes are the most relevant to derive the class assignment of the iamge. The original NeSy model also had a visual and semantic explainer which were ignored in this thesis out of concerns for fairness in model comparison, e.g., our model candidate αILP is not designed to output saliency maps [Stammer et al., 2021a]

- $material/(2,[obj,material])$: probability that the object is one of the material rubber or metal.
- $size/(2,[obj,size])$: probability that the object is one of the sizes large or small.

Both authors empirically found that their models could consistently outperform the pure deep learning baseline, a ResNet34. This means that both models can tolerate the visual confounder during testing, a sign that they were able to disentangle the concepts of color from shape or size, unlike the ResNet34 which overfits to it.

However, in our present thesis, in addition to assessing them again on CLEVR-Hans3, we aim to run fairer model comparisons between NeSy architectures and a pure deep learning baseline. This is because in their original works, we note striking unfairness in model comparison given that the pure deep learning baseline considered, ResNet34, is directly compared to Concept Learner or αILP . Although the ResNet34 have 40 times as many parameters than either NeSy architecture, it is pretrained on an substantially different dataset, ImageNet. Its NeSy counterparts, on the other hand, had a Slot Attention with the unfair advantage of being pretrained on CLEVR's scene graph annotations which is highly convenient for solving CLEVR-Hans3 [An et al., 2023] because it offers a disentangled representation of the input image. As a result of this, it is not evident the advantages of NeSy modelling over pure deep learning since it is not a carefully controlled comparison. In order to account for this, we introduce another model variant, a ResNet acting as the "symbolic component" which receives the object-centric representation of the Slot Attention in order to derive the class label. If NeSy architectures can improve over this baseline, then it would validate the strengths of NeSy modeling since it would imply the necessity of the symbolic component for solving complicated classification tasks like CLEVR-Hans3.

5.1.1 Models' Setups

Following the above motivation, we consider the below model variants to be assessed on CLEVR-Hans3:

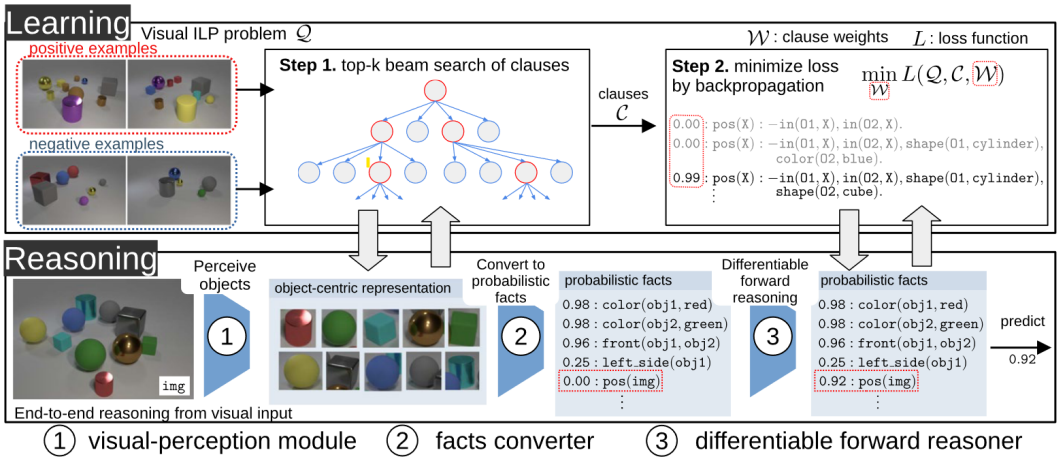


Figure 5.2: αILP Pipeline. At the initial stages of training, a group of candidate clauses are generated by top-k beam search. The weights for the generated clauses are trained to minimize the loss function measuring the goodness of each clause in predicting the class label of the input images. Through an iterative process, αILP tries to find clauses which better explain the visual scenes by gradient descent. Figure extracted from [Shindo et al., 2023]

1. **ResNet18:** Following from their works, we use ResNet18 as our pure deep learning baseline. We opted for this smaller version in contrast to the traditional ResNet34 for computational convenience during training. We load it pre-trained on ImageNet and replace its last layer with a linear one over 3 classes before training it on CLEVR-Hans3. We note that despite having half the size of ResNet34, we are still able to closely replicate results reported by both [Stammer et al., 2021a] and [Shindo et al., 2023] as Table A.3 shows.

2. **Reasoning ResNet18 (Slot Attention + ResNet18):** We assemble the Slot Attention pre-trained on CLEVR and a ResNet18 pre-trained on ImageNet aiming for a more controlled comparison. To link them together, the object-centric representation output by the Slot Attention is given as an input to the ResNet18. Here, the ResNet18 can be thought of as the baseline for the symbolic component, i.e., either the Set Transformer or Forward Reasoner, hence we name it Reasoning ResNet18. Because the Slot Attention's output is tridimensional $B \times O \times D$, lacking the dimension for channels, we concatenate three replicas of this object-centric representation to obtain a $B \times 3 \times O \times D$ tensor that can be fed to the ResNet18.

3. **Concept Learner (Slot Attention + Set Transformer):** This model variant is the same as described in [Stammer et al., 2021a]

4. **αILP (Slot Attention + Forward Reasoner): This variant is the same as described in [Shindo et al., 2023]**. It is worth noting that αILP has a different training regime than the above models. By design of ILP problems, the model is unable to solve the ternary classification task of CLEVR-Hans3, but rather the problem has to be framed as 3 binary classification tasks, i.e., 3 ILP tasks. For example, αILP first learns to recognize class 0 images by looking at 3000 posi-

tive instances of it, as well as 6000 negative samples (class 1 and 2). Two other independent αILP repeat this procedure for class 1 and 2 images. The training, validation and testing accuracies achieved by the three models are averaged to obtain a representative score of how well it predicts on the CLEVR-Hans3 problem as a whole.

We trained them by minimizing the cross-entropy loss with the same hyperparameter configuration whenever possible: 50 training epochs, a learning rate of $1e^{-4}$, batch size of 128, Adam optimizer $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e^{-8}$ and zero weight decay. The Slot Attention have three more hyperparameters set: 10 slots, 3 iterations per slot attention, and 19 attribute. For αILP , extra hyperparameters concerning top-k beam search include for depth of search T_{beam}, which for class 0 is: $T_{beam} = 5$, class 1: $T_{beam} = 6$, class 2: $T_{beam} = 7$. All classes have width of beam $N_{beam} = 20$ and 2 as a maximum number of objects. To quantify the uncertainty of our models' performances, each is run for 3 random seeds, whereby for αILP it means each class is ran 3 times, with training, validation, and testing accuracies across classes averaged and reported. We also note that to train αILP for class 0 and class 1 binary classification tasks, we had to reduce batch size down to 16 given that our available computational resources face a bottleneck if the batch size is bigger. For training class 2, the batch size had to be scaled down to 8 to avoid memory allocation problems. This is most likely due to longer clauses, as well as higher amount of predicates and facts involved in recognizing class 2 images. The batch size for beam search during validation and testing was set to 24, albeit we note that this does not affect training, but is rather used to speed up inference by exploiting batch computation.

5.2 Results and Discussion

The results of our experiments in Figure 5.3 are mostly in tandem with the ones reached by the above authors. Namely, NeSy modeling outperforms the pure deep learning baseline of ResNet18 because the latter learnt a distributed representation of the training data which easily conflated the concepts of shape and color during training. This is evidenced by the perfect training accuracy, followed by a sharp drop in testing accuracy attributed to the confounder. Interestingly, however, our Reasoning ResNet18's performance was able to perform slightly better than the Concept Learner by about 3% (see Table A.3), proving that [Stammer et al., 2021a]'s Set Transformer is a replaceable "reasoning" component in the NeSy architecture. This is mainly due to the contribution of the Slot Attention being more important than the symbolic component's given that the former is greatly simplifying the CLEVR-Hans3 classification task by providing a very useful feature decomposition of the input space.

We did not match our classification scores for αILP to those reported by [Shindo et al., 2023] (almost 98% for validation and testing), partly because we did not follow their hyperparameter configuration out of the aspiration for controlled inter-model comparison, i.e., they employ a batch size of 256 with an optimizer of RMSProp with a learning rate of 0.01 and 100 training epochs. Despite that, in our experiments, αILP is the best performing model especially when we observe its testing accuracy which is the highest among the 4 variants, meaning that it successfully disentangled the confounder

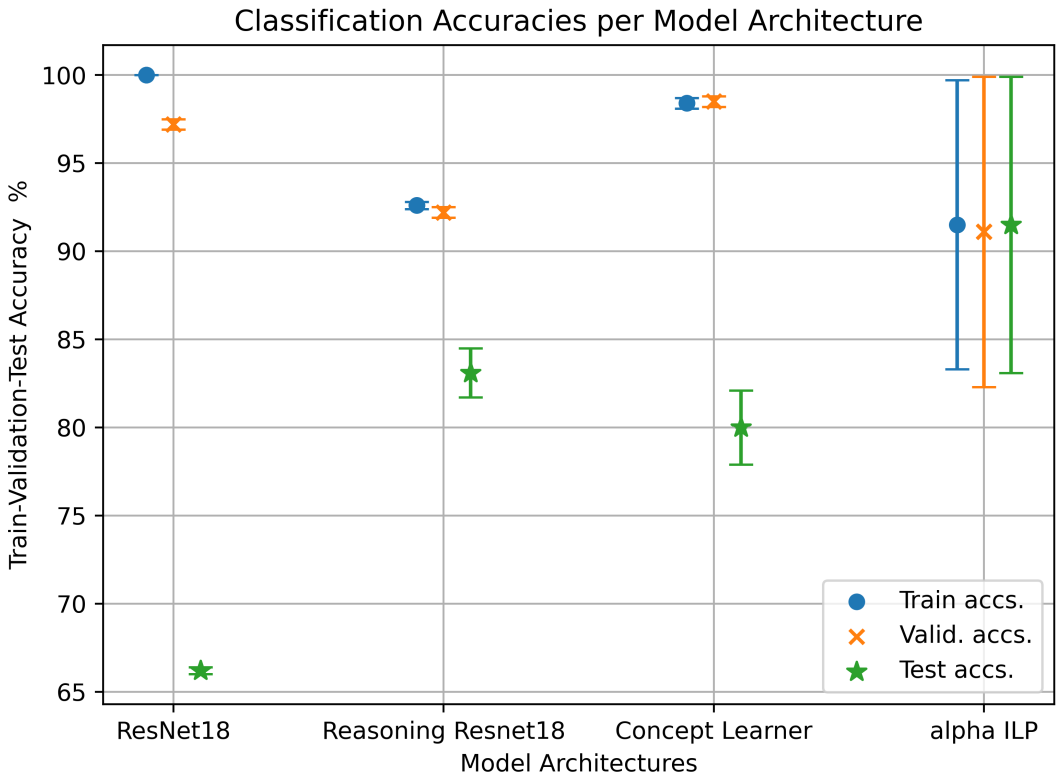


Figure 5.3: Classification accuracies obtained for the 4 model variants. αILP is the best performing model, albeit at the cost of being subject to unfair amount of supervision.

for class 0 and 1 images. Furthermore, another attractive feature of αILP is the transparency of the logic program. Namely, we can pinpoint that the cause of the high classification performance is due it accurately finding the clauses which closely or perfectly match the classification rules for each class (recall from Figure 4.1). Below is one example per class of the final clause found:

- $clevr0(X)$:- $in(O1,X), in(O2,X), shape(O1,cylinder), shape(O2,cube), size(O1,large), size(O2,large)$.
- $clevr1(X)$:- $in(O1,X), in(O2,X), material(O2,metal), shape(O1,sphere), shape(O2,cube), size(O1,small), size(O2,small)$.
- $clevr2(X)$:- $color(O1,blue), color(O2,yellow), in(O1,X), in(O2,X), shape(O1,sphere), shape(O2,sphere), size(O1,large), size(O2,small)$

The large variance of the accuracies can be attributed to it sometimes being unable to find the best rules due to only 50 training epochs and small learning rate. For example, in one of the random seeds for class 0, αILP was only able to find a clause "a small sphere and small metal cube" that minimally approximated to the ground truth rule: $clevr0(X) : -in(O1,X), in(O2,X), material(O2,metal), shape(O1,sphere), shape(O2,cube), size(O1,small), size(O2,small)$

Another interesting aspect of αILP is the series of evaluation metrics one can assess

it with, such as clause generation time and prediction time over batch size of images queried, both which are proxies for measuring the tractability of the logic program found. While we did not focus on running such evaluations despite it being a core aspect of our proposed SaSSY-CLEVR benchmark, we point to [Shindo et al., 2023] to the interested reader for the ablation study they performed.

One concerning limitation of αILP is that despite it supports batch computation for probabilistic inference over multiple images, because this model $Ne[FOL]$ belongs to the family of nested architectures, it faces bottlenecks when scaling. With the available computational resources, we have witnessed that during training, we are unable to train αILP with a batch size 128 without running into memory allocation errors, unlike with counterparts like ResNet18 or the Reasoning ResNet18 which are exponentially larger than αILP in terms of number of parameters, but simpler in terms of computation. Another issue with αILP is that by abstracting CLEVR-Hans3 as 3 binary classification tasks, it can be argued that it has an unfair amount of supervision since it has seen the whole dataset for 3 times in total. Further research can seek to address these problems.

5.3 Limitations and Future Prospects

We now proceed in discussing the general limitations of NeSy models used in our experiments, followed by what are the future directions that stem from this. While it is convincing that NeSy modeling can robustly complete a classification task under visual confounder, its seemingly over-reliance on the Slot Attention's object centric representation can be seen as its Achille's Heel if we wanted to test such methods on other tasks where such pretrained model is not conveniently available. Future research direction can therefore be aimed at exploring how we can relax the usage of the pretrained Slot Attention. One such possibility is to employ NeSy architectures like DeepProbLog [Manhaeve et al., 2021] or Scallop [Li et al., 2023]. For the former, consider a very simple ProbLog inspired from CLEVR-Hans3 program:

```
object(O) :- between(1,10,O).
nn(shape_network, [X, O], Y, [sphere, cube, cylinder]) :: shape(X, O, Y) :- object(O).
nn(size_network, [X, O], Y, [large,small]) :: size(X, O, Y) :- object(O).
clvr(X, C) :- size(X, Obj1 , large), shape(X, Obj1, cube), C = 0.
```

This probabilistic logic program is querying whether image X can be classified as 0 if there is an object of large size and cube shaped. The Problog built-in ternary predicate is a generator which consider object IDs between 1 and 10, meaning that the program will check among 10 objects whether one of them fulfills the queried properties. The probabilities of the associated predicates are estimated by neural networks, with the advantage that it is not required for images to be given labels of which objects are present and what are the attributes per object. The logic program specification is enough to drive the neural network to learn the probability distribution of each attribute like shape and size per object. Being able to relax the Slot Attention's usage is also especially relevant for models which have a dominant symbolic component in order to facilitate transferability to other domains, a known problem with NeSy ar-

chitectures [Hamilton et al., 2022]. For example, at its current design, it is debatable whether αILP or the Concept Learner can solve other problems like traditional image classification without a Slot Attention that decomposes the relevant features given an input.

Furthermore, there are also a plethora of experimental conditions that would be interesting to test using CLEVR-Hans3, such as exploring how would NeSy variants perform when faced with dataset problems like class skewness or small training data, in order to empirically assess whether the promises outlined in 1.2 such as being able to learn from low training data are being upheld (for a more thorough assessment in the field of Natural Language Processing see [Hamilton et al., 2022]). While our benchmark strikes to be encompassing, by no means we have covered the whole breadth of diverse problems. For instance, theorem proving, working with sequence of mathematical symbols is absent in our benchmark.

Chapter 6

Conclusions

In the present thesis, we contribute to the growing literature on NeSy the following: 1) a general taxonomy for classifying current and future NeSy models, helping readers first-hand estimate the streghts and weaknesses of different families of NeSy models. 2) We propose a common benchmark, SaSSY-CLEVR, which encompass important reasoning tasks and we envision it can be used to comprehensively test a variety of NeSy models. Finally, 3) we run experiments on one aspect of our benchmark which enabled us to compare and contrast two different NeSy models, identify their strengths and limitations to overcome in future research.

Neuro-Symbolic AI is still in its nascent period as evidenced by the carefully-designed laboratory problems. However, it is quickly taking off by iteratively improving over predecessor NeSy candidates. We look forward to empirically confirming whether NeSy catalyzes the Third Wave of AI as argued by [Garcez and Lamb, 2020].

Bibliography

[Ahmed et al., 2022a] Ahmed, K., Teso, S., Chang, K.-W., Van Den Broeck, G., and Vergari, A. (2022a). Semantic probabilistic layers for neuro-symbolic learning.

[Ahmed et al., 2022b] Ahmed, K., Teso, S., Chang, K.-W., Van den Broeck, G., and Vergari, A. (2022b). Semantic probabilistic layers for neuro-symbolic learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 29944–29959. Curran Associates, Inc.

[An et al., 2023] An, X., Chen, J., and Paolo, C. (2023). Sassy (symbolic and subsymbolic) attention: Assessing the robustness of neuro-symbolic modelling in the clevr-hans3 dataset. *Final report for Machine Learning Practical (INFR 11132)*.

[Banburski et al., 2020] Banburski, A., Gandhi, A., Alford, S., Dandekar, S., Chin, S., and tomaso a poggio (2020). Dreaming with ARC. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*.

[Barbu et al., 2019] Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., and Katz, B. (2019). Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *NeurIPS Proceedings*.

[Barredo Arrieta et al., 2020] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115.

[Barrett et al., 2018] Barrett, D., Hill, F., Santoro, A., Morcos, A., and Lillicrap, T. (2018). Measuring abstract reasoning in neural networks.

[Branwen, 2020] Branwen, G. (2020). The scaling hypothesis.

[Chavira and Darwiche, 2008] Chavira, M. and Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172:772–799.

[Chollet, 2018] Chollet, F. (2018). On the measure of intelligence. *arXiv*.

[Clark, 1997] Clark, A. (1997). *Introduction: A Car with a Cockroach Brain*, pages 1–8. Being there: putting brain, body, and world together again. Cambridge, Mass. MIT.

[De Raedt et al., 2020] De Raedt, L., Dumančić, S., Manhaeve, R., and Marra, G. (2020). From statistical relational to neuro-symbolic artificial intelligence.

[Diligenti et al., 2017] Diligenti, M., Gori, M., and Saccà, C. (2017). Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165. Combining Constraint Solving with Mining and Learning.

[Donadello et al., 2017] Donadello, I., Serafini, L., and Garcez, A. D. (2017). Logic tensor networks for semantic image interpretation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1596–1602. AAAI Press.

[Dong et al., 2019] Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. (2019). Neural logic machines. In *International Conference on Learning Representations*.

[Ellis et al., 2021] Ellis, K., Wong, C., Nye, M., Sablé-Meyer, M., Morales, L., Hewitt, L., Cary, L., Solar-Lezama, A., and Tenenbaum, J. B. (2021). Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, page 835–850, New York, NY, USA. Association for Computing Machinery.

[Evans and Grefenstette, 2018] Evans, R. and Grefenstette, E. (2018). Learning explanatory rules from noisy data. *J. Artif. Int. Res.*, 61(1):1–64.

[Garcez and Lamb, 2020] Garcez, A. d. and Lamb, L. C. (2020). Neurosymbolic ai: The 3rd wave. *arXiv:2012.05876 [cs]*.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Hamilton et al., 2022] Hamilton, K., Nayak, A., Božić, B., and Longo, L. (2022). Is neuro-symbolic ai meeting its promises in natural language processing? a structured review. *Semantic Web*, pages 1–42.

[Harmelen and Teije, 2019] Harmelen, F. v. and Teije, A. t. (2019). A boxology of design patterns for hybrid learning and reasoning systems. *Journal of Web Engineering*, 18:97–124.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Hersche et al., 2023] Hersche, M., Zeqiri, M., Benini, L., Sebastian, A., and Rahimi, A. (2023). A neuro-vector-symbolic architecture for solving raven's progressive matrices. *Nature Machine Intelligence*, 5(4):363–375.

[Higgins et al., 2018] Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a definition of disentangled representations. *arXiv:1812.02230 [cs, stat]*.

[Johnson et al., 2017] Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. (2017). Clevr: A diagnostic dataset for compositional

language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Kaplan et al., 2020] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv:2001.08361 [cs, stat]*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84–90.

[Lample and Charton, 2019] Lample, G. and Charton, F. (2019). Deep learning for symbolic mathematics. *arxiv*.

[Li et al., 2018] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399.

[Li et al., 2023] Li, Z., Huang, J., and Naik, M. (2023). Scallop: A language for neurosymbolic programming. *arXiv:2304.04812 [cs]*.

[Liu et al., 2021] Liu, H., Dai, Z., So, D. R., and Le, Q. V. (2021). Pay attention to mlps. *arXiv:2105.08050 [cs]*.

[Lu et al., 2021] Lu, K., Grover, A., Abbeel, P., and Mordatch, I. (2021). Pretrained transformers as universal computation engines. *arXiv:2103.05247 [cs]*.

[Manhaeve et al., 2021] Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., and De Raedt, L. (2021). Neural probabilistic logic programming in deepproblog. *Artif. Intell.*, 298(C).

[Mao et al., 2019] Mao, J., Gan, C., Deepmind, P., Tenenbaum, J., and Wu, J. (2019). The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arxiv*.

[Marra et al., 2020] Marra, G., Diligenti, M., Giannini, F., Gori, M., and Maggini, M. (2020). Relational neural machines. *arXiv*.

[Marra and Kuželka, 2021] Marra, G. and Kuželka, O. (2021). Neural markov logic networks. In de Campos, C. and Maathuis, M. H., editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 908–917. PMLR.

[Marra and Kuželka, 2020] Marra, G. and Kuželka, O. (2020). Neural markov logic networks. *arXiv*.

[McCarthy et al., 1955] McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (1955). A proposal for the dartmouth summer research project on artificial intelligence. *AI Magazine*, 27:12–14.

[McClelland et al., 1986] McClelland, J., Rumelhart, D., and Hinton, G. (1986). The appeal of parallel distributed processing. In *Parallel Distributed Processing: Explo-*

rations in the Microstructure of Cognition, Vol. 1: Foundations, pages 3–44. MIT Press.

[Minsky, 1990] Minsky, M. (1990). Logical vs.analogical or symbolic vs. connectionist or neat vs. scruffy.

[Misino et al., 2022] Misino, E., Marra, G., and Sansone, E. (2022). Vael: Bridging variational autoencoders and probabilistic logic programming. *arxiv*.

[Moor, 2006] Moor, J. (2006). The dartmouth college artificial intelligence conference: The next fifty years. *AI Magazine*, 27:87–91.

[Moskvichev et al., 2023] Moskvichev, A. K., Odouard, V. V., and Mitchell, M. (2023). The conceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *Transactions on Machine Learning Research*.

[Muthukrishnan et al., 2020] Muthukrishnan, N., Maleki, F., Ovens, K., Reinhold, C., Forghani, B., and Forghani, R. (2020). Brief history of artificial intelligence. *Neuroimaging Clinics of North America*, 30:393–399.

[Nilsson, 2009] Nilsson, N. J. (2009). *Part I - Beginnings*, pages 17–71. The Quest for Artificial Intelligence. Cambridge University Press.

[Rocktäschel and Riedel, 2017] Rocktäschel, T. and Riedel, S. (2017). End-to-end differentiable proving. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3791–3803, Red Hook, NY, USA. Curran Associates Inc.

[Saker et al., 2021] Saker, K., Zhou, L., Eberhart, A., and Hitzler, P. (2021). Neuro-symbolic artificial intelligence current trends. *arxiv*.

[Santoro et al., 2017] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4974–4983, Red Hook, NY, USA. Curran Associates Inc.

[Shindo et al., 2023] Shindo, H., Pfanschilling, V., Dhami, D. S., and Kersting, K. (2023).

α

ilp: thinking visual scenes as differentiable logic programs. *Machine Learning*.

[Singh et al., 2022] Singh, A., Hu, R., Goswami, V., Couairon, G., Galuba, W., Rohrbach, M., and Kiela, D. (2022). Flava: A foundational language and vision alignment model (preprint). *arXiv:2112.04482 [cs]*.

[Stammer et al., 2021a] Stammer, W., Schramowski, P., and Kersting, K. (2021a). Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations.

[Stammer et al., 2021b] Stammer, W., Schramowski, P., and Kersting, K. (2021b). Right for the right concept: Revising neuro-symbolic concepts by interacting with

their explanations. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3618–3628.

[Sun et al., 2022] Sun, J., Tjandrasuwita, M., Sehgal, A., Solar-Lezama, A., Chaudhuri, S., Yue, Y., and Costilla-Reyes, O. (2022). Neurosymbolic programming for science. *arxiv*.

[Van Emden and Kowalski, 1976] Van Emden, M. H. and Kowalski, R. A. (1976). The semantics of predicate logic as a programming language. *Journal of the ACM*, 23:733–742.

[Van Krieken et al., 2022] Van Krieken, E., Thanapalasingam, T., Tomczak, J., Van Harmelen, F., and Ten Teije, A. (2022). A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. *arXiv (preprint)*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Šourek et al., 2015] Šourek, G., Aschenbrenner, V., Železny, F., and Kuželka, O. (2015). Lifted relational neural networks. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*, COCO'15, page 52–60, Aachen, DEU. CEUR-WS.org.

[Weber et al., 2019] Weber, L., Minervini, P., Münchmeyer, J., Leser, U., and Rocktäschel, T. (2019). NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. Association for Computational Linguistics.

[Wong et al., 2021] Wong, C., Ellis, K. M., Tenenbaum, J., and Andreas, J. (2021). Leveraging language to learn program abstractions and search heuristics. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11193–11204. PMLR.

[Xu et al., 2018a] Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Van Den Broeck, G. (2018a). A semantic loss function for deep learning with symbolic knowledge.

[Xu et al., 2018b] Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Van den Broeck, G. (2018b). A semantic loss function for deep learning with symbolic knowledge. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5502–5511. PMLR.

[Yang et al., 2017] Yang, F., Yang, Z., and Cohen, W. W. (2017). Differentiable learning of logical rules for knowledge base reasoning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2316–2325, Red Hook, NY, USA. Curran Associates Inc.

[Yi et al., 2020] Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. (2020). Clevrer: Collision events for video representation and reasoning. *International Conference on Learning Representations*.

[Yi et al., 2018] Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. B. (2018). Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 1039–1050, Red Hook, NY, USA. Curran Associates Inc.

[Yu et al., 2022] Yu, D., Yang, B., Liu, D., Wang, H., and Pan, S. (2022). Recent advances in neural-symbolic systems: A survey.

[Yu et al., 2014] Yu, W., Yang, K., Bai, Y., Yao, H., and Rui, Y. (2014). Visualizing and comparing convolutional neural networks. *arxiv.org*.

[Zhang et al., 2019] Zhang, C., Gao, F., Jia, B., Zhu, Y., and Zhu, S.-C. (2019). Raven: A dataset for relational and analogical visual reasoning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5312–5322.

[Zhang et al., 2021] Zhang, D., Mishra, S., Brynjolfsson, E., Etchemendy, J., Ganguli, D., Grosz, B., Lyons, T., Manyika, J., Niebles, J. C., Sellitto, M., Shoham, Y., Clark, J., and Perrault, R. (2021). Chapter 2: Technical performance in artificial intelligence index report 2021 artificial intelligence index report 2021 2.

[Zhang et al., 2023] Zhang, H., Li, L. H., Meng, T., Chang, K.-W., and den Broeck., G. V. (2023). On the paradox of learning to reason from data. In *IJCAI*.

Appendix A

First appendix

A.1 Survey of NeSy Models and Taxonomical Categorization

NeSy models	Neural components	Symbolic component	Taxonomical categorization	Description
DreamCoder [Ellis et al., 2021]	Program recognition module	Program synthesis	{[Ne]}HOL or {[Ne]}{[HOL]}	NN decides how to chunk primitive operators into reusable programs
NeSy-Visual Question Answering (VQA) [Yi et al., 2018]	Mask RCNN for image recognition + LSTM to parse questions	SQL-like Query executor	{[Ne]}HOL	NN learns to output SQL-table representation and symbolic component execute SQL-query
NeSy-Dynamic Reasoner (DR) [Yi et al., 2020]	Video frame and Question Parsers	SQL-like Query executor	{[Ne]}HOL	See NeSy-VQA
αILP [Shindo et al., 2023]	Pretrained Slot Attention	Differentiable forward reasoner	Ne{[FOL]}	NN output object-centric representation and forward reasoner learns classification rule
DeepProbLog [Manhaeve et al., 2021]	User-specified neural network	ProbLog using probabilistic circuits for scalable inference	Ne{[FOL]} or {[Ne]}{[FOL]}	Probabilities of neural predicates are estimated by NNs with feedback from ProbLog program
Scallop [Li et al., 2023]	User-specified neural network	Probabilistic inference	Ne{[FOL]} or {[Ne]}{[FOL]}	See DeepProbLog
δILP [Evans and Grefenstette, 2018]	User-specified neural network	Inductive Logic Programming component	Ne{[FOL]}	See DeepProbLog
Neural Logic Machine (NLM) [Dong et al., 2019]	User-specified neural network	User-specified logic program	Ne{[FOL]}	See DeepProbLog
Neural Markov Logic Networks (NMLM) [Marra and Kuželka, 2021]	Standard feed-forward neural networks to represent factor graph	Probabilistic inference	Ne{[FOL]}	The NN learns under constraints imposed by FOL prior knowledge

Logic Tensor Networks (LTN) [Donadello et al., 2017]	User-specified neural network	User-specified logic program in Real Logic	Ne : {Fuzzy FOL}	See NMLM
Relational Neural Machine (RLM) [Marra et al., 2020]	Neural Markov Logic Networks	Weighted probabilistic inference	Ne : {Fuzzy FOL}	See NMLM
DiffLog (Si et al. 2019)	Markov Logic Network	Probabilistic inference	Ne{[FOL]}	See NMLM
Lifted Relational Neural Networks (LRNN) [Šourek et al., 2015]	Neural Markov Logic Networks	Probabilistic inference	Ne : {Fuzzy FOL}	See NMLM
Tensorlog [Cohen et al., 2017]	Neural Markov Logic Networks	Probabilistic inference	Ne{[FOL]}	See NMLM
Neural Logic Programming (NeuralLP) [Yang et al., 2017]	Neural Markov Logic Networks	Probabilistic inference	Ne{[FOL]}	See NMLM
Neural Theorem Provers (NTP) [Rocktäschel and Riedel, 2017]	Standard feedforward network	Backward chaining	Ne : {Fuzzy FOL}	Vector representations of logical predicates
Semantic-based regularization (SBR) [Diligenti et al., 2017]	Markov Logic Networks	Probabilistic inference	Ne : {Fuzzy FOL}	
NLProlog [Weber et al., 2019]	Pretrained sentence encoders	Prolog Backward chaining	Ne : {Fuzzy FOL}	
Semantic Probabilistic Layer (SPL) [Ahmed et al., 2022b]	CNN	Approximate inference using probabilistic circuit	{Ne} : Fuzzy Prop	Hard logical constraints over learning
Semantic Loss Function (SLF) [Xu et al., 2018b]	Standard feedforward neural network	Maximum likelihood	{Ne} : Fuzzy Prop	Constrained learning
Variational Auto-Encoder + Probabilistic Logic Programming (VAEL) [Misino et al., 2022]	CNN	Probabilistic inference	Ne : {Fuzzy FOL}	Constrained generative approach
* NeSY-Explainable Interactive Learning (XIL) [Stammer et al., 2021b]	Pretrained Slot Attention	Set Transformer	N/A	

Table A.1: A more detailed classification of NeSy architectures based on our proposed taxonomy. Notice that NeSy-XIL, despite being proposed as a NeSy method by [Stammer et al., 2021a], we argue that it's not evident how the Set Transformer act as a symbolic component in terms of whether it's defined through a logical system like a logic program.

A.2 Survey of Benchmarks and Associated Models

Datasets	Nature of task	Format	Challenging aspect	Assessed NeSy models
CLEVR	Object-centric relational reasoning			NS-VQA
CLEVR-CoGenT	Object-centric relational reasoning	Split A: 70K images and 700K natural language questions for training. Split B: 15K images and 150K questions for validation and testing. Image-Text	VQA with confounding Variables	NS-VQA
CLEVR-Hans	Object-centric relational reasoning	Each class is represented by 3000 training, 750 validation, and 750 test image	Classification with visual confounder during training	αILP; NeSy-XIL
Procedurally Generated Matrices Datasets	Object-centric relational and Abstract reasoning	1.2M training set questions, 20K validation set questions, 200K testing set questions	Implicit patterns resembling IQ-tests	
Kadinsky Patterns	Object-centroic relational reasoning	Variable, user can generate desired amount of images according to 3 classification rules	Implicit patterns resembling IQ-tests	αILP
ILP experiments (find successor, determine even or odd, fuzz-buzz, etc.)	Task-driven reasoning	20 Benchmark ILP provided by δILP	Have to construct new rules, ensure satisfiability and tractability	δILP
Digit image classification (eveness, equal to 1, less than)	Task-driven reasoning; Object-centric relational reasoning	10 tasks about arithmetic	Satisfiability; Tractability	δILP
Find if digit is followed by subsequent digits	Task-driven and Object-centric relational reasoning	Variable, can syntethize as many samples as desired	Digits are MNIST images; scalability to multi-digits	RNM
Citeseer for document classification by symbolic bag of words	Task-driven reasoning	4732 links (relations), 3703 words (constants)	Satisfiability; Tractability	RNM
MNIST Digit addition	Task-driven reasoning; Object-centric relational reasoning	Variable, can synthetize as many sums by picking any combination of digits	Digits are MNIST images; scalability to multi-digits addition	DeepProbLog; LTN
Inflammation, polysite, sql datasets	Knowledge graph reasoning	34 benchmarks	Multi-hop reasoning, KB-completion, satisfiability	DiffLog
LRNN datasets	Knowledge graph reasoning	78 datasets (1 mutagenesis, 4 predictive toxicology, 73 National Cancer Institute datasets)	Multi-hop reasoning	LRNN
Unsupervised data clustering	Task-driven reasoning	Variable, can syntethize as many datapoints as desired	Clustering can be written as a logic program	LTN
Classification of Leptograpsus crabs	Task-driven reasoning	200 examples of 5 morphological measurements of 50 crabs	Classification can be written as a logic program	LTN
Regression on real state dataset	Task-driven reasoning	414 examples with 6 real-numbered features	Regression can be written as a logic program	LTN
Smokers-friends	Knowledge graph reasoning	14 friends 7 with smoking habits	Satisfiability; Tractability	LTN; NMLM; Tensorlog

Nations, Kinship and Unified Medical Language System (UMLS)	Knowledge graph reasoning	Nation: 56 binary predicates, 14 entities, 2565 facts. Kinship: 26 predicates, 104 entities, 10686 facts. UMLS: 49 predicates, 135 entities, 6525 facts	Multi-hop reasoning; Satisfiability; Tractability	Neural LP; NMLN; NTP; Tensorlog
Classification on WebKB	Task-driven reasoning	4100 webpages and 10000 hyperlinks	Satisfiability	SBR
Wiki-Movies KB	Knowledge graph reasoning	196453 train examples of QA pairs and 10000 test QA pairs. 43230 entities (constants) and 9 relations (predicates) [according to neural lp]	Multi-hop reasoning; Satisfiability	Neural LP
Freebase15K	Knowledge graph reasoning	483142 training examples and 50000 validations examples and 59071 test examples: 14951 entities (constants), 1345 relationships (predicates)	Multi-hop reasoning; Satisfiability	Neural LP; NMLN
Wordnet	Knowledge graph reasoning	141442 training, 5000 validation, 5000 testing examples ,40943 entities, 18 relationships	Multi-hop reasoning; Satisfiability	Neural LP; Tensorlog;NMLN;
Wiki-Hop	Knowledge graph reasoning	43738 training prompts and QAs, 5128 validation, 2451 test	KB combines natural language with some FOL	NLProlog
Med-Hop	Knowledge graph reasoning on KG	1620 training prompts and QAs, 342 validation, 546 tests	KB combines natural language with some FOL	NLProlog
Sorting Arrays	Task-driven reasoning	Variable, can syntethize as many arrays of variable size	Training with short arrays must generalize to longer arrays	NLM
Block's World Problem	Task-driven reasoning	Variable, can add several rules and entities to test scalability	Satisfiability, Tractability	NLM
Family tree reasoning	Knowledge graph reasoning	Variable, can add several rules and entities to test scalability	Multi-hop reasoning; Satisfiability	NLM; DiffLog
Finding shortest-path	Task-driven reasoning	256 grid cells (entities), 2116 edges (relations)	Multi-hop reasoning; Satisfiability	NLM ; Tensorlog
Countries	Knowledge graph reasoning	272 entities (244 countries, 5 continents, 23 subregions), 1158 facts	Multi-hop reasoning; Satisfiability	NTP
CORA citation-matching task	Task-driven reasoning	13K facts, 10 rules	Satisfiability, tractability	Tensorlog
ChEMBL molecule database	Knowledge graph reasoning	Variable given it's a manually curated dataset	Multi-hop reasoning; Satisfiability	NMLN
* CLEVRER	Dynamic object-centric relational and Counterfactual reasoning	20,000 synthetic videos of colliding objects and more than 300,000 questions and answers	VQA over counterfactual events	NeSy-DR
* MNIST, FASHION, and CIFAR-10	semi-supervised classification		Constrained learning	SLM
* Prediction task using PrefLib	Preference learning	10 types of sushi for 5000 individuals	Constrained learning	SLM
* Finding shortest-path	Grid exploration	4x4 grid to represent a graph with uniform edge weights. 1600 examples	Constrained learning	SLM

Table A.2: Non-exaustive, detailed overview of datasets and models identified. Generally, satsifiability refers to the percentage of tasks solved with the correct answer, such as identifying correctly a queried family relationship. Tractability is often measured as the time it takes to complete the query given a larger size of the knowledge base or a more complicated problem, for example the addition of new predicates in UMLS. Datasets marked with a * either fall outside the scope of our taxomony as they are not reasoning tasks which require the manipulation of logical predicates, rather learning tasks where the symbolic component simply acts as hard constrains driving optimization. We also add a * to the CLEVRER dataset since it mainly consists of videos rather than images

	Training accuracy	Validation accuracy	Testing accuracy
ResNet18	**1**	0.972 ±0.003	0.662 ±0.002
Slot Attention + ResNet18	0.9286±0.002	0.922 ±0.003	0.831 ±0.014
Slot Attention + Set Transformer	0.984 ±0.003	**0.985 ±0.003**	0.8 ±0.021
Slot Attention + Forward Reasoner	0.915 ±0.082	0.911±0.088	**0.915 ±0.084**

Table A.3: Breakdown of classification accuracies of model architectures considered obtained in the training, validation and testing sets of the ternary classification task CLEVR-Hans3. For the αILP , we obtain the accuracies by averaging over the 3 ILP tasks.

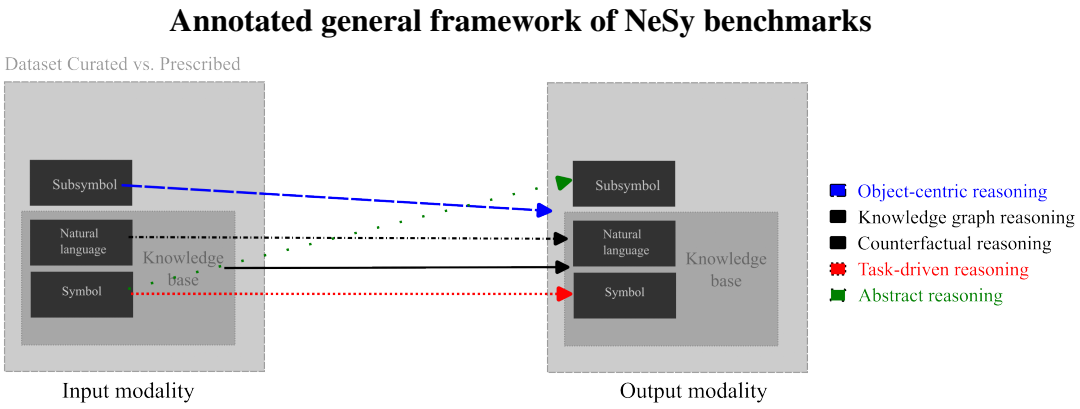


Figure A.1: General framework with pathways annotated indicating the most common transformation from input to output modality.

Appendix B

Participants' information sheet

If you had human participants, include key information that they were given in an appendix, and point to it from the ethics declaration.

Appendix C

Participants' consent form

If you had human participants, include information about how consent was gathered in an appendix, and point to it from the ethics declaration. This information is often a copy of a consent form.