On Symbol Grounding in Machine Learning

Sever Topan

School of Computer Science, McGill University, Montreal

February 2022

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science, under the supervision of Professor Xujie Si

Abstract (English)

Symbol grounding is an abstract phenomenon commonly referenced in cognitive science that concerns how symbols – such as letters, words, or mathematical formulae – obtain their meanings. This phenomenon has recently become of interest in the context of Neurosymbolic Machine Learning: an area of research that aims to integrate symbolic logical reasoning into deep learning architectures. While subtle, symbol grounding can dramatically affect the difficulty of learning problems and proposed solutions. This became apparent in a recently proposed differentiable MAXSAT Solver, SATNet. SATNet was a breakthrough in its capacity to integrate with a traditional neural network and solve visual reasoning problems. For instance, it can learn the rules of Sudoku purely from image examples. Despite its success, SATNet was found to be unable to map visual inputs to symbolic variables without intermediary supervision (referred to as "label leakage" in criticizing papers), thus failing to overcome the particularly difficult instance of symbol grounding for Visual Sudoku.

In this thesis, we propose a novel formalization of the Symbol Grounding Problem in the context of machine learning applications. This enables theoretical insights into the phenomenon, allowing us to describe a taxonomy of factors that make it difficult and relevant for different model architectures. A self-supervised pre-training pipeline is then presented that enables SATNet to overcome its prior limitation in solving the Symbol Grounding Problem for Visual MAXSAT tasks such as Visual Sudoku. This broadens the class of problems that SATNet architectures can solve to include datasets where no explicit intermediary supervision is possible. We demonstrate that our method allows SATNet to attain a new State-of-the-Art accuracy on Visual Sudoku with a harder problem setup that prevents label leakage. Thus, this thesis is an exploration of Symbol Grounding as it applies to machine learning, with an applied focus on solving Visual MAXSAT problems using SATNet architectures.

Abstract (French)

L'ancrage des symboles est un phénomène abstrait communément étudié en sciences cognitives, et qui concerne la manière dont les symboles – tels que les lettres, les mots ou les formules mathématiques – obtiennent leur signification. Ce phénomène s'est récemment manifesté dans le contexte de l'apprentissage automatique neuro-symbolique : un domaine de recherche qui intègre le raisonnement logique symbolique dans les architectures d'apprentissage profond. Bien que subtile, l'ancrage des symboles peut considérablement affecter la difficulté d'apprentissage de problèmes et de leurs solutions. Cela est devenu apparent dans le cas d'un solveur MAXSAT différentiable récemment proposé, nommé SATNet. SATNet a été une percée significative pour sa capacité d'intégration à un réseau neuronal traditionnel et de résolution de problèmes de raisonnement visuel. Par exemple, il peut apprendre les règles du jeu de Sudoku uniquement à partir d'exemples imagés. Malgré son succès, SATNet s'est avéré incapable d'associer des données visuelles à des variables symboliques sans supervision intermédiaire (un problème nommé « label leakage » dans les articles critiques). Par conséquent, il ne peut pas surmonter ce cas particulièrement difficile d'ancrage des symboles pour le Sudoku visuel.

Dans cette thèse, nous proposons une nouvelle formalisation du problème d'ancrage des symboles dans le contexte des applications de l'apprentissage automatique. Cela rend possible l'analyse théorique du phénomène, permettant de décrire une taxonomie de facteurs rendant le problème difficile et pertinent pour différentes architectures de modélisation. Un pipeline de pré-entraînement auto-supervisé est ensuite présenté pour permettre à SATNet de surmonter ses limitations antérieures dans la résolution du problème d'ancrage symbolique pour les tâches de Visual MAXSAT telles que le Sudoku visuel. Cela élargit la classe des problèmes que les architectures SATNet peuvent résoudre pour inclure des ensembles de données où aucune supervision intermédiaire explicite n'est possible. Nous démontrons que notre méthode permet à SATNet d'atteindre une nouvelle précision de

pointe pour le Sudoku visuel avec une configuration de problème plus difficile qui empêche la fuite d'étiquettes. Ainsi, cette thèse est une exploration de l'ancrage symbolique tel qu'il s'applique à l'apprentissage automatique, en portant une attention particulière à la résolution des problèmes Visual MAXSAT à l'aide des architectures SATNet.

Acknowledgements

I was very lucky to be supported by many kind individuals over the course of the past two years. I'd firstly like to thank my family, particularly my parents Sorina and Mihai for their constant support day in and day out. Additionally, I am thankful to Prof. Xujie Si for providing a welcoming and thought-provoking research environment throughout the program. I learned a lot during this time, and will have many fond memories of our work together. I'd like to also mention Michael Cox and Jonas Nilsson for helping me balance my career and academic goals. Working in a primarily online setting during a pandemic can be quite isolating; as such, I want to say how grateful I am for all the friends and colleagues who helped foster a sense of community during my study.

Finally, I'd like to thank collaborators for their contributions and guidance in putting this thesis together: Prof. David Rolnick for our joint work on Symbol Grounding with SATNet, Prof. Brigitte Pientka for her early guidance when I first started the masters, Prof. Robert Robere and Gabriel Arpino for their suggestions on complexity bounds. Marc-Antoine Ouimet and Marie-Estelle Cousein for help with French translation.

Contents

1 Introduction

Imagine that you are sitting at a table, looking out in front of you at a Sudoku Puzzle inscribed on a piece of paper. You see a grid, partially filled with numbers. You understand this is a puzzle, and that there are certain strict rules that must be followed to solve it correctly. In the case of Sudoku, the rules act upon the shape of the digits – of which there are 9 categories – as well as their positions within the grid. The font, slant, and thickness of the digits carry no relevant information. Thus, in order to learn how to solve Sudoku, you must both (a) learn how to identify the starting board states, and (b) learn how to solve the subsequent logical puzzle. This thesis is about these two sub-problems, and how we can design algorithms that learn to solve them. We will discuss how they require fundamentally different modalities of reasoning – one is example-driven, while the other is rule-based. We will show that the dichotomy between these two modalities exists broadly in machine learning, and that different model architectures can change how difficult it is to bridge the gap between them. We will begin, however, by spending some more time motivating our work by describing these two types of reasoning as they apply to Sudoku.

Digits are not what your brain is directly seeing when you look at the Sudoku puzzle. The understanding of the puzzle in front of you is constructed from a collection of electrical impulses originating from the cells within your retina – the biological analogy for pixels. These impulses are processed into the higher-level construct of digits within a grid. The way in which we as humans learn how to see digits is interesting. There are rules which govern how digits should look, but they are all fundamentally approximate. When seen visually, digits are images that can be represented in a variety of fonts, colors, or backgrounds. There are many corner cases where a digit is ambiguous – we all have seen cases where 4's look like 9's. Thus, the way in which we learn to see digits is inherently *example-driven*. There are many similar tasks which we as humans perform, from understanding writing, to hearing words, to riding a bicycle. Our brain seems to have a specialized modality of learning for these types of tasks.

We use this example-driven type of learning to solve many basic problems in our day-to-day life, often without expending much mental effort or attention. This modality of thought has been called "bottom-up," "statistical", "heuristic" or "connectionist" in literature, and is related to what Daniel Kahneman calls "System 1" thinking [1, 2, 3].

Now contrast the above with learning the actual rules of Sudoku. We are now no longer in an example-driven setting. Starting from the concept of digits and their positions, the rules of Sudoku can be precisely stated in a formal manner. You would never need to have seen a Sudoku puzzle before, but reading a description of the rules would allow you to solve any instance of it. This is an example of a second, more formal modality of thought that we seem to be able to operate in. Much of the higher-level function in our brain appears to work at this level, and operating in this modality requires explicit effort and attention. Kahneman terms this, "System 2" thinking, and it is also called "symbolic", "analytic" or "Bottom-up" reasoning [1, 3, 2]. Solving math problems, designing an engine and filling out our taxes are examples of this form of thinking.

Despite the differences between our statistical approach to identifying board state and our symbolic method of navigating Sudoku rules, we seem to be able to transparently combine both of them to solve Sudoku in a straightforward manner. Every day, we reason in ways that combine statistical and symbolic approaches to solve complex and impressive tasks. This applies far beyond Sudoku. Traffic rules are well-defined, but obstacles can be difficult to identify. Scientific experiments are often conducted by following specific protocols, but qualitative results can be very nuanced. There exists a confluence between statistical, bottom-up reasoning and symbolic, top-down reasoning in human thought, and we often cycle between them without realizing. Indeed, it can be argued that much of our intellectual success as a species stems from the ability to find a balance between these two modalities of thinking.

1.1 Neural and Symbolic Machine Learning

Interestingly, this tension between the statistical and the symbolic also exists in the current state of the art in machine learning. Recent years have seen significant advancements in statistical methods – most notably deep learning – providing breakthroughs in image, video, and audio processing [4]. Despite its success, deep learning has many known limitations which are common among statistical approaches, such as low interpretability, vulnerability to adversarial attacks, and difficulty in solving problems that require hard logical constraints [5, 6, 7, 8]. As machine learning systems are deployed across an increasing array of domains, it is becoming clear that purely optimizing for performance is not sufficient to make good products. This is particularly true in applications where safety is of concern.

To overcome these limitations, experts have described the need to migrate from purely statistical, deep learning-based systems to neurosymbolic artificial intelligence systems that more closely resembles the way humans approach complex reasoning [9]. Neurosymbolic systems merge statistical methods with symbolic approaches which were more typical in the older, "first wave of AI" [10]. As discussed, symbolic approaches can encode logical reasoning more formally, alleviating some of the drawbacks of pure statistical approaches.

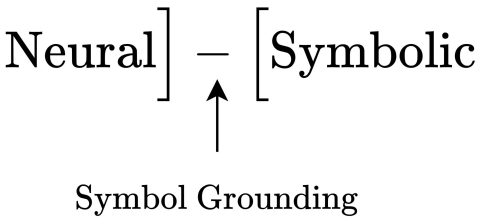

Figure 1: Symbol Grounding occurs at the confluence between statistical (neural) and symbolic systems. A mapping must be established between the artifacts which the symbolic system uses, and those which the neural component learns.

Whenever statistical and symbolic systems are composed, an interesting phenomenon occurs at their interface. The meanings of the symbols which the symbolic component uses

must be established in the context of the statistical model. In other words, a mapping must be learned between the artifacts which the symbolic system uses, and those which the neural component learns. This phenomenon is called *Symbol Grounding*, and it is the confluence between statistical and symbolic reasoning that is considered by some to be one of the fundamental prerequisites for artificial intelligence [11, 1]. Symbol Grounding is a fairly abstract phenomenon, and it may not be immediately obvious why it is difficult. We will describe it in more detail in the next section.

1.2 Symbol Grounding, and Why it is Difficult

Symbol Grounding was first proposed by Stevan Harnad in 1990 [1]. It deals with the question of how abstract symbols used in logical rule systems obtain their meanings. Harnad originally expressed the Symbol Grounding Problem in the context of coming across a dictionary for a language you are unfamiliar with. If one wanted to learn the meanings of the definitions in the dictionary, one would have to somehow establish a mapping between the words in the dictionary and external phenomena – such as colors, shapes, actions and sounds – which the words represent. Looking purely at the dictionary only maps definitions to other, unknown definitions [1]. These external phenomena which must be grounded to symbols are often called *referents*, and generally exist in an unstructured, high-dimensional representations such as images, audio, or other sensory modalities.

We can return to our Sudoku example to investigate what symbol grounding can look like in a practical setting. We can represent a visual version of Sudoku with a dataset containing input board states as images of digits, and the completed board state as a one-hot digit encoding of the resulting board. In order to solve this learning problem, a model must 1) learn the mappings from input images to symbolic digits representing board state, and 2) learn to derive output board state from input board state. This is a prototypical example in which a Neurosymbolic system would be a reasonable choice, since 1) can be modeled to

high fidelity with neural systems, while 2) can be approached via SAT solving – a symbolic method.

Symbol grounding for Visual Sudoku is the process in which we identify that it is the shape, not the font or orientation of input digits that matters for the downstream puzzle. Once this is learned, the digit categories can be mapped to symbolic variables, which can in turn be used within a rule-based system to solve the overall problem. The difficulty of this may not be immediately apparent, because as humans, we are equipped with prior knowledge about digits heading into the learning problem. We can illustrate what it would be like to approach this problem from a blank slate by making a small change to the problem setup to remove this bias. Imagine that we replace the numbers with images of animals and, crucially, we are not told how to group the images into the 9 categories relevant for Sudoku. The color of the animal could be one grouping, but so could the species, or the direction that the animal is facing. This version of Sudoku is conceptually the same problem, but we now have to learn what to actually look for in the images purely from our training dataset.

Another illustrative example of difficult symbol grounding occurs with Raven's Matrix puzzles, which are visual reasoning problems where a collection of panels are shown that contain some pattern. The player must then choose the final pattern that completes the puzzle. The difficulty, like with our Sudoku example, lies in finding what to look for, and what the pattern is. Figure 2 shows examples of Raven's Matrix Problems which illustrate the difficulty of symbol grounding. The two instances on the left introduce the problem, while the right-most is a difficult puzzle which we invite the reader to attempt.

In summary, Symbol Grounding involves identifying a mapping between some precept within a high-dimensional input signal, and a symbol system. It can range in difficulty significantly based on the amount of prior or contextual information present.

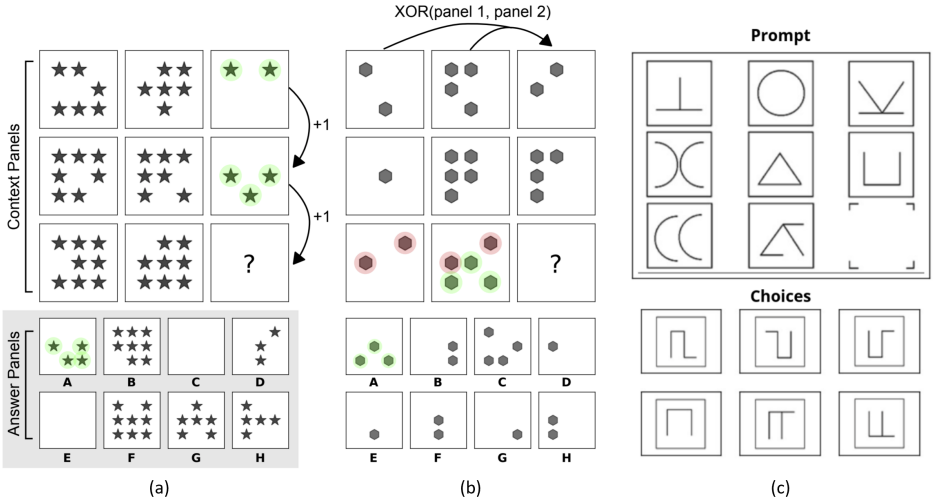

Figure 2: Examples of Raven's Matrix problems. The pattern in **(a)** is a simple arithmetic progression along the columns, while **(b)** is a `XOR` operation along the rows. **(a)** and **(b)** are both taken from [12]. **(c)** is a challenging instance of the problem taken from [11]. It illustrates the difficulty of symbol grounding, since both the rules and the symbols themselves need to be identified. We invite the reader to attempt to solve the puzzle, and check their answer against the solution provided in the original publication.

1.3 Contributions

This work is an exploration of Symbol Grounding from a Machine Learning perspective. In short, we provide the following two main contributions:

1. We formalize the definition of Symbol Grounding in the context of Machine Learning. Symbol Grounding historically is a fairly philosophical and abstract notion. We propose a more rigorous definition in the context of Neurosymbolic systems, accompanied by a taxonomy of factors which affect its difficulty. We survey the field and show how existing models solve different versions of the Symbol Grounding Problem.

2. We demonstrate a method of overcoming the symbol grounding problem for Visual MAXSAT problems using an architecture called SATNet [13]. We use this method to achieve a new state-of-the-art on the Ungrounded Visual Sudoku problem, which was previously unsolved.

Both sections can be read in isolation from one another, though the prior establishes context for the latter.

Unless otherwise noted, all experiments and writing were performed by Sever Topan, with guidance and edits by Prof. Xujie Si. The complexity bounds presented in section 2 benefited from helpful conversations with Prof. Robert Robere and Gabriel Arpino. Section 3 was written in collaboration with Prof. David Rolnick, who provided significant contributions, especially surrounding the concept of "Symbol Grounding Loss." Section 3 was accepted as a spotlight paper to NeurIPS 2021 [14]. Additionally, Marc-Antoine Ouimet and Marie-Estelle Cousein helped correct the French translation of the Abstract.

2 Proposed Symbol Grounding Formalism

In this section, we propose a novel formalization of Symbol Grounding from a Machine Learning perspective. We begin by introducing preliminaries and scoping our contribution relative to prior work. We then proceed to define our formalism, discuss its implications, and survey the field to find how existing algorithms fit within our definition.

2.1 Background

2.1.1 Connectionist vs. Symbolic Systems

A fundamental preliminary to our formalism will be to establish the definition of what a symbolic system in fact is. Here we will invoke Harnad's 1990 definition, which in itself is based on a survey of prior work [1]:

> A symbol system is: (1) a set of arbitrary "physical tokens" scratches on paper, holes on a tape, events in a digital computer, etc. that are (2) manipulated on the basis of "explicit rules" that are (3) likewise physical tokens and strings of tokens. The rule-governed symbol-token manipulation is based (4) purely on the shape of the symbol tokens (not their "meaning"), i.e., it is purely syntactic, and consists of (5) "rulefully combining" and recombining symbol tokens. There are (6) primitive atomic symbol tokens and (7) composite symbol-token strings. The entire system and all its parts – the atomic tokens, the composite tokens, the syntactic manipulations both actual and possible and the rules – are all (8) "semantically interpretable:" The syntax can be systematically assigned a meaning, e.g., as standing for objects, as describing states of affairs.

Harnad then proceeds to contrast these with connectionist systems such as neural networks, which he claims fail properties (7) and (8) above.

> The patterns of interconnections do not decompose, combine and recombine according to a formal syntax that can be given a systematic semantic interpretation.

Though much of the success and prominence of connectionist approaches is much more recent than Harnad's publication, the claim still holds [4]. In the author's words, a way to summarize the distinction between connectionist and symbolic systems is that symbolic systems bake the information that they carry into the atoms and rules which their algorithms directly manipulate. The information present in connectionist systems is instead an emergent phenomenon that manifests at a higher level than the atoms that are being manipulated.

For instance, contrast the weights within a (connectionist) neural network with a (symbolic) C++ program. Information flows through a C++ program at the level of the functions and operators present within it. Thus, reading the program local structure reveals the logic that it encodes. This is not the case for neural networks; Reading local weight relationships generally will not reveal the relevant logic encoded within the system, which instead results from large complexes of weight relationships that lack a higher-level interpretation. Symbolic systems thus lend themselves well to manual crafting, since the rules and atoms themselves are meaningful to humans. Connectionist systems on the other hand generally need to be constructed using data-driven approaches.

2.1.2 Symbol Grounding Prior Work

The most concise recent definition that the author has found is the one proposed by Cangelosi in 2011 [15]:

> To summarize, we can say that there are three sub-problems in the development of a grounded symbol system:
>
> 1. how can a cognitive agent autonomously link symbols to referents in the world such as objects, events and internal and external states;

2. how can an agent autonomously create a set of symbol-symbol relationships and the associated transition from an indexical system to a proper symbol system;

3. how can a society of agents autonomously develop a shared set of symbols

It is worthwhile to mention that the topic of Symbol Grounding, and more precisely whether it has been "solved," is contested [16, 17, 18, 19, 20]. This is likely due to the fact it is an interdisciplinary phenomenon that lies at the intersection between philosophy, computer science, and psychology. The original formulation of Symbol Grounding was primarily philosophical one which dealt with human language, and discussed implications such as how human brains use symbolic reasoning [1]. This somewhat railroaded the conversation on symbol grounding into a discussion about the limits and properties of human-like artificial intelligence. Researchers have since argued that this angle has entangled a somewhat emotional, anthropomorphic theme into the more abstract debate about systems learning to ground symbols [16]. A second issue is that much symbol grounding literature appears to fixate on grounding human language as opposed to more general symbolic systems which would be desirable of a definition that applies more abstractly [19].

2.1.3 Formalism Scope

Items 1 and 2 will be the primary focus of this work. We consider item 3 to be an extension of the core symbol grounding problem, which will be discussed in section 2.4.2.

To the author's best knowledge, no existing comprehensive formalization of symbol grounding exists. The closest work found was proposed in [21], introducing a formalism for Cangelosi's item 1 (which is referred to as "Perceptual Anchoring" in their work). However, the existing formalism does not capture item 2. Furthermore, it is somewhat prescriptive, limiting the symbols present to predicate logic – essentially simple attributes associated to precepts. We would instead prefer to allow for more complex relations to be captured by the

connectionist system, such that it can be coupled with any rule-based system, even higher-order-logic. Thus we view our proposed formalism as a revision and generalization of their framework.

2.2 Formalization

Our goal is to provide a formalization of Symbol Grounding from a *machine learning perspective*, particularly in the context of Neurosymbolic architectures. This formalization will serve as a model to discuss the aspects that make symbol grounding difficult and relevant to machine learning applications, and will explicitly not attempt to address the aspects of symbol grounding that lie in the domain of philosophy or neuroscience. We will address the first two sub-problems in Cangelosi's formulation, but from an abstract perspective which generalizes past human language. To our knowledge, this is the first such proposed formalization.

2.2.1 General Form

Let Σ represent some symbolic alphabet, and Ξ represent some non-symbolic input data type. We define $f_{\mathcal{S}} : \Sigma \to \Sigma$ as a purely symbolic function as defined by Harnad in [1], and $f_{\mathcal{R}} : \Xi \to \Sigma$ to be a mapping function from a non-symbolic input to the symbol system. We say $f_{\mathcal{R}}(x_\Xi) = \hat{z_\Sigma}$ and $f_{\mathcal{S}}(\hat{z_\Sigma}) = \hat{y_\Sigma}$. We then introduce the following definitions:

Definition 1. *Training the composite system $f_{\mathcal{S}} \circ f_{\mathcal{R}}$ end-to-end on a dataset $\{x_\Xi : \Xi, z_\Sigma : \Sigma, y_\Sigma : \Sigma\}$ entails solving items 1 and 2 in Cangelosi's symbol grounding problem definition [15] for the learning problem defined by $\{x_\Xi, z_\Sigma, y_\Sigma\}$.*

Definition 2. *Symbol grounding is also possible in lieu of z_Σ. We call a dataset "grounded" if z_Σ is present, and "ungrounded" if it is not.*

A constructive reasoning for this definition follows. By definition, symbol grounding presupposes the existence of a formal symbolic system ($f_{\mathcal{S}}$), so we use this as our starting

point. We make no assumptions on the data formats of $\hat{y_\Sigma}, y_\Sigma, \hat{z_\Sigma}$ or z_Σ. We do not impose limitations on the form or syntax of Σ, or the complexity of the symbol. For example, different $f_\mathcal{S}$ implementations may take Σ to be first-order predicates, while others may leverage higher-order ones. Furthermore, z_Σ can be omitted entirely from the training dataset. In this case the correct $\hat{z_\Sigma}$ must be inferred indirectly from y_Σ.

In order to establish a link between referents external to the symbolic system described in Cangelosi's item 1, we need to define the additional function $f_\mathcal{R}$, which derives symbols $\hat{z_\Sigma}$ from some arbitrary data x_Ξ containing relevant referents. We will discuss in section 2.2.3 why it is necessary for $f_\mathcal{R}$ to be a separate non-symbolic system from $f_\mathcal{S}$. We place no constraints on x_Ξ, as we wish to be able to capture any possible data format. Finally, in order to resolve item 2, the composite $f_\mathcal{S} \circ f_\mathcal{R}$ must be learnable. We have thus constructed a system that encompasses both of Cangelosi's sub-problems.

The following list enumerates the fundamental properties of our formulation.

1. The composite function $f_\mathcal{S} \circ f_\mathcal{R}$ must be learnable.
2. The nature of compute present in $f_\mathcal{S}$ is symbolic.
3. The nature of compute present in $f_\mathcal{R}$ is non-symbolic.
4. The presence of z_Σ in the training dataset is optional.

We will spend some time further discussing the implications of each of these properties in our formalism.

2.2.2 The Composite Function $f_\mathcal{S} \circ f_\mathcal{R}$ Must be Learnable

It is well understood that Symbol Grounding involves learning a mapping from some precept to a symbol system, hence a reader might expect our definition to require that $f_\mathcal{R}$ is learnable. We actually make a different assertion, instead requiring that *at least one* of $f_\mathcal{R}$ and $f_\mathcal{S}$ be

learnable. That is to say, it is possible for both $f_{\mathcal{R}}$ and $f_{\mathcal{S}}$, or simply only $f_{\mathcal{S}}$ to be learnable and for symbol grounding to still occur.

This might appear unexpected since traditionally, symbol grounding seems to imply that the symbol system in question is static. In Harnad's original formulation, for example, the problem involved mapping dictionary definitions to external referents, where the difficulty lied in establishing this symbol-referent mapping [1]. The contents of the dictionary are assumed to be unchanging. Indeed, In many cases where symbol grounding is approached from the anthropomorphic angle focusing on language, the fact that the symbol system itself can change is ignored. It is important to avoid this assumption, as is made evident by Cangelosi in point 1. Some works do explicitly discuss the fact that symbol systems themselves can be learned [16] [1], but we would like to develop this idea further.

We claim that any learnable symbol system undergoes a form of symbol grounding when it is trained. The intuition here is if symbol grounding entails mapping some external precept to a symbol system, this can be achieved by both developing a mapping from precepts to symbols, or by changing the symbols themselves to more directly support the precept. We call this *dual symbol grounding*; The primary form being the traditional precept-symbol learning ($f_{\mathcal{R}}$), whereas the secondary form involves training the symbol system ($f_{\mathcal{S}}$).

An implication of this is that for certain learning problems, symbol grounding can be achieved exclusively though the secondary form. We propose that, given a fixed precept-symbol mapping, training the symbolic component of the system in isolation is still a form of symbol grounding. Since we have not imposed any constraints on x or $\hat{z_\Sigma}$, strictly speaking it is possible to devise a symbol system that operates directly with the precept x as its symbols for certain learning problems (i.e. $f_{\mathcal{R}}$ is identity).

We underline that our claims surrounding dual symbol grounding are in line with points 1 and 2 from Cangelosi's definition [15]. In fact, it forms the synthesis between these points:

[1] Social symbol grounding (Cangelosi's point 3) is another instance of symbol systems undergoing change, though in this case it is primarily for the purpose of coordination.

By allowing the system to generate its own symbol-symbol relationships, the system can better establish a link between referents and symbols. This is made explicit within our formalism.

2.2.3 The Nature of Compute in $f_{\mathcal{R}}$ is Non-Symbolic

The function $f_{\mathcal{R}}$ is defined to be non-symbolic in our formulation. This done simply because if $f_{\mathcal{R}}$ were symbolic, we could model the same system by absorbing it into $f_{\mathcal{S}}$, and find ourselves lacking a non-symbolic component in the loop. This is equivalent to having $f_{\mathcal{R}}$ fixed to an identity function, and thus only the secondary form of symbol grounding introduced in section 2.2.2 applies. The non-symbolic constraint allows us to capture both primary and secondary forms of symbol grounding.

2.2.4 Presence of z_{Σ} is Optional

The final facet of the formalism which we introduce is a way in which we differentiate between two types of datasets: grounded (z_{Σ} is present) and ungrounded (z_{Σ} is missing). The presence of z_{Σ} in a dataset in principle doesn't change the fact that symbol grounding *is occurring*, however it changes the nature of the learning problem. Training $f_{\mathcal{R}}$ on a grounded dataset can be cleanly split into separate learning tasks for $f_{\mathcal{R}}$ and $f_{\mathcal{S}}$. This learning dataset is thus already grounded in the primary form, requiring grounding only for the secondary case. The grounded/ungrounded dataset distinction is important because it makes a significant difference in the difficulty of certain problem settings, as evidenced in [11]. Our work in Section 3 heavily focuses on this aspect.

2.3 Implications of our Formalism

Now that we have introduced the formalism surrounding symbol grounding, we will take some time to discuss implications of this formalism on the categorization of different neurosymbolic

machine learning architectures, as well as implications on their learning complexity.

2.3.1 Symbol Grounding Model Architecture Taxonomy

We consider our formalism to be a taxonomy for Neurosymbolic architectures. This section is not meant to be a literature review of symbol grounding (for which we direct the reader to [22]), but instead shows how different architectures can be grouped into families.

Table 1 displays the general taxonomy that we will propose. The taxonomy is centered around three indicator variables: (a) whether $f_{\mathcal{R}}$ is learnable, (b) whether $f_{\mathcal{S}}$ is learnable, and (c) whether the dataset supported by the architecture is grounded or not. We note that the presence of a grounded dataset is only relevant in the context of a learnable $f_{\mathcal{R}}$. We will walk through the different relevant combinations next.

	$f_{\mathcal{R}}$ **Learnable?**	$f_{\mathcal{S}}$ **Learnable?**	**Presence of** z_{Σ}	**Example Model**
(i)	no	no	N/A	N/A - not learnable
(ii)	no	yes	N/A	Knowldege Graph Methods [23]
(iii)	yes	no	grounded	N/A - directly supervised
(iv)	yes	yes	grounded	SATNet [13], CLEVR models [24, 25]
(v)	yes	no	ungrounded	DeepProbLog [26]
(vi)	yes	yes	ungrounded	Our method in Section 3

Table 1: A taxonomy of symbol grounding models, with associated example architectures. Note that grounded datasets are only relevant when $f_{\mathcal{R}}$ is learnable.

(i) Fixed $f_{\mathcal{R}}$, Fixed $f_{\mathcal{S}}$ – No learning occurs within the problem setup, therefore it is not relevant for symbol grounding.

(ii) Fixed $f_{\mathcal{R}}$, Learnable $f_{\mathcal{S}}$ – This problem setup captures cases where only the secondary form of symbol grounding applies. Models such as the Knowledge graph building system introduced in [23], the Integer Linear Programming constraint miner from [27], or SATNet in the pure-symbolic setting [13] are examples of this category.

(iii) Learnable $f_{\mathcal{R}}$, Fixed $f_{\mathcal{S}}$, Grounded Dataset – In this problem setting there is direct supervision $f_{\mathcal{R}}$ while it is the only learnable component within the system. This implies that we are in a standard supervised learning setup in which $f_{\mathcal{S}}$ can be bypassed during training. Thus, no symbol grounding occurs.

(iv) Learnable $f_{\mathcal{R}}$, Learnable $f_{\mathcal{S}}$, Grounded Dataset – In this category of symbol grounding architectures, both the connectionist and symbolic components are learnable, but the dataset used to train them is grounded, meaning that intermediary labels are present which aid the training of $f_{\mathcal{R}}$. The original SATNet implementation [13] was show to fall into this category in [11]. Another example of models in this category are systems which train on the CLEVR dataset using signals such as scene graph information such as [24, 25].

(v) Learnable $f_{\mathcal{R}}$, Fixed $f_{\mathcal{S}}$, Ungrounded Dataset – This class of architectures contains models where a connectionist model must be mapped to a fixed symbolic system. This is the classical primary case of symbol grounding (Cangelosi point 1). The interesting aspect of these types of models is that generally the symbolic systems need to be differentiable and easily craftable or interpretable by humans, since they are fixed a priori. DeepProbLog is an example of such an architecture [26].

(vi) Learnable $f_{\mathcal{R}}$, Learnable $f_{\mathcal{S}}$, Ungrounded Dataset – This taxonomic category allows for the most degrees of freedom. Both symbol grounding components are learnable, and no intermediary labels are present in the training dataset. We postulate that this is the most difficult symbol grounding setting, since $f_{\mathcal{R}}$ must be trained through a learning signal

passed through $f_{\mathcal{S}}$, which must in turn also be trained. We discuss learning complexity further in Section 2.3.2. Our extension to the SATNet architecture which we introduce in Section 3 is an example of this problem setting where Ungrounded Visual MAXSAT problems are covered. [11] also introduces the "MNIST Mapping Problem," which falls into this taxonomic category.

2.3.2 Learning Complexity Bounds

Our formalism allows us to reason about model architecture choices that make symbol grounding more or less difficult. We are able to show a collection of simple bounds on the learning complexity of two of these design choices below.

Proposition 2.1. *Learning complexity for the ungrounded version of a learning problem is greater than or equal to that of the grounded version.*

Proof. The above can be shown by reduction. Given a system that is capable of solving some Ungrounded learning problem, we can allow it to solve the grounded version of the same dataset by simply removing the intermediary labels z_{Σ}[2]. □

Proposition 2.2. *For a given model, the learning complexity of training both $f_{\mathcal{R}}$ and $f_{\mathcal{S}}$ greater than or equal to that of simply training one of them while the other is fixed to the correct solution.*

Proof. This follows by a reduction in the starting conditions of the learning problem. Let there exist a model architecture in which both $f_{\mathcal{R}}$ and $f_{\mathcal{S}}$ are initialized to some known starting state. Assume this model is capable of solving some learning problem. We can reduce any model in which either $f_{\mathcal{R}}$ or $f_{\mathcal{S}}$ are fixed on the correct solution to our original model by re-initializing that component to the known starting state, and unfreezing it during training. □

[2]Note that in this problem setup we don't consider the intermediary labels to be constraints on the system, only hints

The above statements allow us to sketch a hierarchy of difficulty within the symbol grounding taxonomy introduced in Section 2.3.1. Our propositions imply that the taxonomic category where both $f_{\mathcal{R}}$ and $f_{\mathcal{S}}$ are learnable, and the training dataset is ungrounded is the most difficult symbol grounding problem setting.

While propositions 2.1 and 2.2 are relatively weak, we believe that stronger bounds can be shown, however we leave this to future work. Our empirical results in Section 3 appear support this claim for the SATNet architecture setting which we explore.

2.4 Discussion

2.4.1 On The Symbol Grounding Problem Being Solved

If one accepts our Machine Learning-centric definition of the Symbol Grounding Problem, it becomes clear that claiming that the Symbol Grounding Problem is "Solved" full stop, is nonsensical. It is similar to claiming that Dynamic Programming is solved. The claim of solvability only makes sense in the context of the problem that is being addressed, in the same way that we can say that we have a Dynamic Programming solution to Rod Cutting [28]. In this way we can say that, for example, symbol grounding has already been solved for certain logical reasoning datasets such as with DeepProbLog [26].

We do make this claim while understanding that Harnad's original Symbol Grounding Problem was tailored towards solving human cognition relating to language, and in that respect there might be a clear decision of whether there is a solution or not [1]. What is important, is that authors contextualize their use of the term "Symbol Grounding" with respect to the problem actually being targeted. As such, in Harnad's examples it should be clear that he discusses Symbol Grounding *specifically for human language*.

2.4.2 Relevance of Social Symbol Grounding

Social symbol grounding (item 3 from Cangelosi's definition), is generally considered to not be part of the base symbol grounding question [22]. It is an extension of symbol grounding that mainly stems from the anthropomorphic human language angle that we attempt to generalize past in this work. Harnad's original definition in fact does not imply that shared understanding is necessarily part of the question, his formulation primarily deals with a single actor interacting with symbolic systems (in his case these systems are dictionaries). We thus avoid the topic of developing shared symbols in this work.

Nonetheless, we believe that social symbol grounding can be modeled as an alternate learning objective within the framework which we have proposed. Social symbol grounding is primarily about consensus, so once could consider embedding this as a learning objective within the training dataset.

3 Symbol Grounding for Visual MAXSAT Problems Using SATNet

We will now turn our attention to a class problems in which symbol grounding is particularly difficult, known as *Ungrounded Visual MAXSAT*. To tackle these learning tasks, we focus on improving a promising development in the field of Neurosymbolic Machine Learning: the award-winning architecture known as SATNet [13]. SATNet is a differentiable MAXSAT solver based on a low-rank semidefinite relaxation approach. It can be integrated into traditional Deep Neural Networks (DNNs) to solve composite learning problems that require both logical reasoning and visual understanding. One such problem is Visual Sudoku, which was introduced in section 1. Here, the model must learn the rules of a Sudoku puzzle purely from visual examples. When trained end-to-end, SATNet is able to achieve 63.2% total board accuracy in this task while traditional DNN architectures are unable to exceed 0% [13]. This was regarded as a significant breakthrough for neurosymbolic architectures. However, it was recently noted that SATNet training relies upon "leakage" of labels through the logical constraint layer to the DNN used to classify digits [11].

This leakage essentially means that SATNet is learning in two supervised stages, where it first trains its digit classification component under direct supervision, and only then trains its SAT layer to learn the logical constraints delineating Sudoku. When the leakage is removed, SATNet's ability to solve Visual Sudoku drops to 0% [11]. This is significant, because taken independently, these two sub-problems are significantly easier. Digit classification is considered a solved problem, and while SAT constraint mining is more difficult, it could be argued that the differentiable aspect is no longer beneficial if the system needs supervision on its inputs to learn regardless. For instance, there exist other SAT constraint miners that are not differentiable but outperform SATNet [27]. Overall, the issue of being unable to learn to solve composite visual reasoning problems end-to-end is in fact the Symbol Grounding

Problem which we have discussed previously.

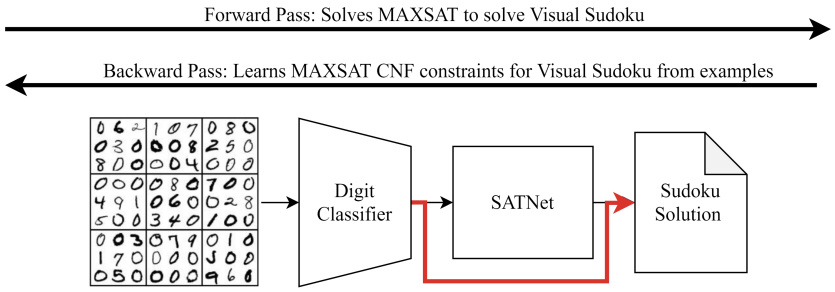

Figure 3: The SATNet architecture used to solve Visual Sudoku. The red line shows the label leakage issue, which when removed, results in the Symbol Grounding Problem.

We observe a key challenge of symbol grounding is the large gap between the compositional nature of logical reasoning and the end-to-end gradient-based nature of neural networks. The former helps to reduce a sophisticated reasoning system into simple, independent modules, each of which can be designed manually or learned, while the latter encourages fusing all components together and using gradients as a universal means for learning. Many recent approaches aim to bridge this gap by relaxing logical constraint solving through numerical optimisations [29, 30, 31, 32]. Although such end-to-end gradient-based optimisation is appealing, it can fail to address seemingly simple tasks like Visual Sudoku. The success of SATNet is in fact due to *inadvertent* supervision of intermediate modules. We argue that compositionality does not have to be the opposite of the end-to-end learning design. The latter is particularly preferable because it eliminates the need for supervision of intermediate modules, which is often required by a compositional design. If compositionality can be trained using self-supervision (i.e. without manual effort), compositionality would then be at least equally preferable. This is the approach that we take in the present work, synergistically combining compositionality with end-to-end learning without any explicit intermediate supervision. We envision our methodology forming a new paradigm for tackling

neurosymbolic learning.

We describe a self-supervised pre-training method that can be used to bootstrap SATNet in order to overcome the Symbol Grounding Problem. Our methodology enables us to tackle a class of what we call *Ungrounded* MAXSAT problems, where label data are available only for the output variables of the MAXSAT problem. In the Visual Sudoku case, this formulation manifests itself as a dataset where, as before, inputs consist of images of digits describing the input cells of a Sudoku board. The labels of the dataset, however, consist of numerical representations only for the board cells that were not given as inputs. This means that there is no way of identifying what digit each input image refers to except by learning the rules of the Sudoku puzzle *in parallel* to predict the non-input values. We refer to this problem as *Ungrounded Visual Sudoku*. We show that our method improves the state of the art on this problem from 0% to 64.8%, achieving similar performance on Ungrounded Visual Sudoku as SATNet with label leakage does in the grounded version of the same problem. In short, our main contributions are the following:

1. We describe a self-supervised clustering and distillation process for training a visual classifier within a SATNet architecture.

2. We introduce a *Symbol Grounding Loss* that makes it possible to train logical constraint layers on an ungrounded symbol representation.

3. We show empirically that our methodology allows SATNet to achieve full performance on *ungrounded* Visual Sudoku (where label leakage is impossible), a task where previous state-of-the-art was 0%.

4. We introduce a *Proofreader* that improves the performance of any SATNet system (grounded or ungrounded), achieving state-of-the-art performance on Visual Sudoku.

3.1 Background

Our contribution draws from several areas. We begin with preliminaries describing the problem, before discussing related work.

3.1.1 The Problem

MAXSAT, the optimisation analog of SAT, represents a rich set of problems to which many program complexity classes can be reduced. A MAXSAT Solver $\mathcal{S}$ aims to maximally satisfy a set of n boolean clauses over m variables by modulating the values of the variables. These clauses are typically written in Conjunctive Normal Form, and represented numerically as a matrix $M \in \mathbb{R}^{n \times m}$. We can further enrich this system by partitioning our variables $a_{1,\dots,m}$ into a subset of fixed inputs $a^{in}_{1,\dots,k}$, and variable outputs $a^{out}_{k+1,\dots,m}$. The system can then be framed functionally:

$$a^{out}_{k+1,\dots,m} = \mathcal{S}(a^{in}_{1,\dots,k}, M), \qquad \text{for } 1 \leq k \leq m. \tag{1}$$

This formulation can be used to capture Sudoku, an example used extensively in this work, where a^{in} represents the input cells of a given Sudoku board, a^{out} represents the cells that we aim to solve for, and M encodes the rules of Sudoku.

MAXSAT Solvers can be leveraged to solve a broader class of problems that we refer to here as *Visual MAXSAT Problems*. These entail a MAXSAT problem where the inputs a^{in} must first be derived from some other representation a^{in}_{visual}. This essentially results in a two-step training problem for which neurosymbolic architectures are optimised[3].

We have now established the preliminaries necessary to describe the Symbol Grounding Problem in the context of Visual MAXSAT solvers. It is the problem of identifying a^{in} given only a^{in}_{visual} and a^{out}. This motivates the distinction between two types of Visual MAXSAT

[3]While it is also possible to train a system end-to-end to derive a^{out} directly from a^{in}_{visual}, we argue that internally the system would need to have some form of representation of this two-step approach regardless.

Datasets: *grounded* and *ungrounded*. An ungrounded dataset contains a^{in}_{visual} as data and a^{out} as labels, while a grounded dataset additionally contains a^{in} in its labels (See Figure 4.

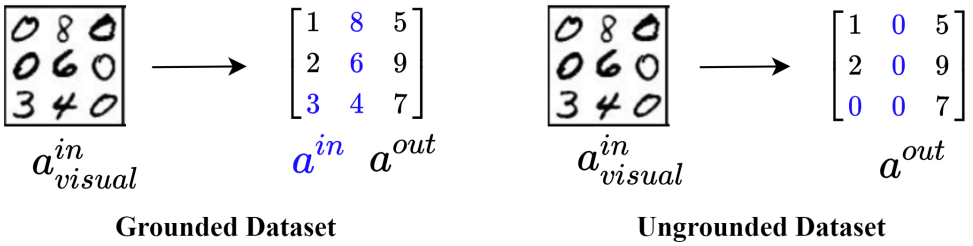

Figure 4: Examples of Grounded and Ungrounded Visual MAXSAT Datasets, focusing on a 3×3 portion of a larger Sudoku board. Blue entries represent input cells. In previous work, SATNet is able to solve only the grounded version of the problem.

We note that it is significantly more difficult to solve the ungrounded version of a Visual MAXSAT problem, as training cannot be trivially broken up into two stages. It is this the class of problems that we tackle in this work.

3.1.2 Logical Constraint Solvers & SATNet

There has been significant recent interest in architectures that can integrate symbolic reasoning layers within neural networks. Many approaches, however, are only capable of integrating pre-existing logical constraints into these models [33, 26, 34, 35, 36]. In the context of our formalism, this is analogous to having a fixed set of clauses M for a particular problem. Conversely, there exists a family of approaches that are not differentiable, but are able to learn logical constraints by example [27, 37, 38]. SATNet, however, sits somewhere in between these approaches, as it is both differentiable and able to learn a matrix M in order to fit some input data [13]. There are a few other algorithms in this class, such as OptNet and ∂-Explainer [29, 30, 39].

3.1.3 Self-Supervised Pre-Training

Self-supervised pre-training has a long history in machine learning, notably being used to navigate highly non-convex loss landscapes in Deep Belief Networks (DBNs) [40, 41]. More recently, better methods for end-to-end training have emerged and self-supervision has now been used to pre-train image tasks on large, cheap unlabeled datasets to obtain slightly better performance on supervised tasks [42, 43].

In our work we return to the insight that motivated the original use of self-supervision for DBNs. The Symbol Grounding Problem essentially represents significant non-convexity in the problem space – both symbol meanings and the way in which symbols interact with one another must be learned in parallel, with local optima existing for many combinations of prospective groundings. Self-supervised pre-training enables us to start training from a favorable position on this loss landscape.

3.1.4 Clustering Algorithms & InfoGAN

Data clustering is a long-standing and rich field of computer science [44]. We leverage clustering in our method in order to conduct self-supervised pre-training. While many clustering algorithms exist, for our purposes we choose to use InfoGAN, as it is able to cluster across the semantic dimension which we are interested in for MNIST with very high accuracy [45].

InfoGAN is a Generative Adversarial Network architecture which boasts disentangled, interpretable latent encodings [46]. It maximizes the mutual information between a subset of the noise fed into its generator, and the observation which the discriminator makes. It is thus able to cluster data according to several interpretable variables. In the case of MNIST, these include handwritten digit thickness, slant, and most useful to us, the actual digit shape. This latter property is what we aim to leverage in this work. Specifically, InfoGAN can cluster MNIST digits according to their numerical value with 95% accuracy in a completely unsupervised fashion [45].

3.1.5 Knowledge Distillation

Knowledge Distillation is a technique for training machine learning models to reach comparable performance at inference time to a larger reference model, or an ensemble of models [47, 48, 49, 50]. While more complex distillation techniques exist, our work leverages the concept in one of its most basic forms – simply training a smaller model from a dataset generated by a larger one in order to drastically improve inference time.

3.2 Method

Our main contribution is a pre-training pipeline used to bootstrap the learning process such that SATNet can bypass the Symbol Grounding Problem. Overall our method entails the following steps.

1. **Clustering**: We first perform unsupervised clustering of the input data, and distill the knowledge of the clusters into a digit classifier.

2. **Self-Grounded Training**: We then employ a custom *Symbol Grounding Loss* to identify how clusters map to the labels we have in our training data. Once the grounding is learned, we freeze it and train the rest of the system.

3. **Proofreading**: We conclude with an optional *proofreading* step which trains an additional layer in the SATNet architecture while the rest remain frozen. This was found to slightly improve performance in all SATNet architectures tested.

Before diving in to each of these steps, we will formalize the composite visual understanding/logical reasoning problem. Assume we look at a single instance of a MAXSAT problem with N variables which can fall into one of K classes, where each of the N variables is represented by an image of size $C \times H \times W$. Our input data is then a tensor $x \in \mathbb{R}^{N \times C \times H \times W}$, and our desired one-hot encoded output $y \in \mathbb{R}^{N \times K}$. Our digit classifier $\mathcal{D}$ takes input x and

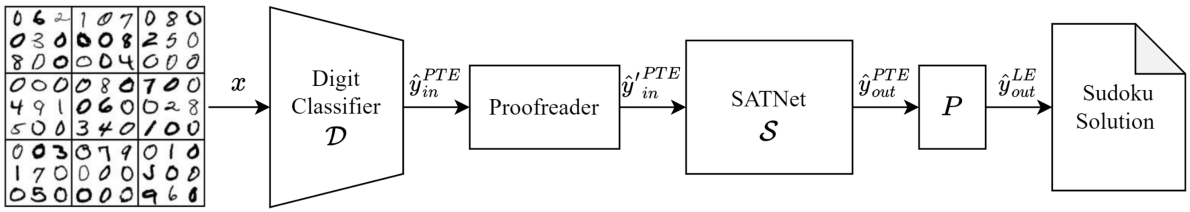

Figure 5: The architecture proposed in this work. It leverages self-supervised pre-training to solve Grounded Visual Sudoku, thereby overcoming the Symbol Grounding Problem affecting the original SATNet method.

returns output $\mathcal{D}(x) = \hat{y}_{in} \in \mathbb{R}^{N \times K}$. We feed this result into our SATNet layer $\mathcal{S}$ such that $\mathcal{S}(\hat{y}_{in}) = \hat{y}_{out} \in \mathbb{R}^{N \times K}$. For Ungrounded Visual Sudoku using MNIST, we have $N = 81$ (one MAXSAT variable for each cell of the 9×9 Sudoku board), $K = 9$ (digits 1 through 9), and $C \times H \times W = 1 \times 28 \times 28$ (MNIST images).

3.2.1 Clustering

Our first step in solving an Ungrounded MAXSAT problem is identifying the patterns that exist in the input data. Intuitively, we do not have to start training a digit classifier from scratch when training composite visual reasoning architectures. There exists some semantic aspect of the input image which is of relevance to the MAXSAT problem at hand, and it can often be at least partially identified in a self-supervised setting. In the Visual Sudoku case, this entails clustering our images into 9 groups (corresponding values 1 through 9).

In our experiments, we use InfoGAN to perform the clustering, as it is capable of clustering MNIST digits with 95% accuracy [45]. Any clustering algorithm may be used here however, even ones that are not differentiable. Once the clustering is complete, we can distill the clustering knowledge back into a differentiable digit classifier. In our case, we generate a dataset using the clustering algorithm, and train LeNet on the cluster allocations of the

training data [51]. By doing this we implicitly map each cluster onto some one-hot representation within $\hat{y}_{in}$. However, this one-hot encoding of the MAXSAT variables may not match with the encoding present in the labels y. We deal with this next.

3.2.2 Self-Grounded Training

While our clustering algorithm might be able to achieve high accuracy, it doesn't have any information about which numerical digit each cluster is actually associated with, since we don't have access to input cell labels. This is the crux of the Symbol Grounding Problem. In an Ungrounded MAXSAT setting, the only way to learn the association between digits and numerical clusters involves *jointly* learning the MAXSAT problem. In the case of Sudoku, this means that we must solve for the rules of puzzle and learn what each digit means simultaneously.

To reason about this, we consider two sets of encodings for digits: the pre-trained encoding (PTE) and the (correct) label encoding (LE), which we notate using superscripts. The digit classifier from the previous step outputs PTE-encoded predictions $\hat{y}_{in}^{PTE}$. There exists some unknown permutation matrix $P \in \mathbb{R}^{K \times K}$ that translates between encodings via $\hat{y}_{in}^{PTE} P = \hat{y}_{in}^{LE}$. Our goal is to align the PTE encoding with the LE encoding, so that we can make use of the training labels. The question of performing this translation before or after the SATNet layer is irrelevant, however. This is because the MAXSAT CNF clauses which SATNet implements are permutation-invariant [52]. This means that as long as supervision is provided correctly, we can train the SATNet layer $\mathcal{S}$ on either $\hat{y}_{in}^{PTE}$ or $\hat{y}_{in}^{LE}$. [4] In our approach we pass the prior through SATNet, and are left with $\hat{y}_{out}^{PTE}$ predictions.

We learn the correct permutation (without access to any of the input labels) simultaneously with training the SATNet layer, by introducing a *Symbol Grounding Loss*, which is

[4] While this is expected, this was not explicitly stated in the original SATNet paper. We were able to verify this empirically by applying any permutation on the one-hot encodings of the digits in the nonvisual Sudoku setting and SATNet's performance is identical even without re-training.

intended to be a smooth function that is minimized when $\hat{y}_{out}^{PTE} P \approx y^{LE}$ for some permutation matrix P.

Note that $\hat{y}_{out}^{PTE}$ and y^{LE} are $N \times K$ matrices, and let $\hat{y}_{out}^{PTE}(i)$ and $y^{LE}(i)$ denote the ith columns of these matrices. Then, $y^{LE}(i)$ is a 1-hot vector capturing the entries of the output that are labeled i (in the correct label encoding), while $\hat{y}_{out}^{PTE}(i)$ is a vector of predicted probabilities that the output is labeled i (in the pre-trained label encoding). We define the following loss $\mathcal{L}$:

$$\mathcal{L}(\hat{y}_{out}^{PTE}, y^{LE}) := 1 - \text{mean}_i(\max_j(\exp[-\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i))])), \tag{2}$$

where $\text{BCE}(\cdot, \cdot)$ denotes the binary cross-entropy loss between two vectors:

$$\text{BCE}(v, w) = -\frac{1}{n}\left(\sum_{k=1}^{n} v_k \log(w_k) + \sum_{k=1}^{n}(1 - v_k)\log(1 - w_k)\right).$$

Proposition 3.1. *Suppose $\mathcal{L}$ is defined as in (2). Then:*

1. *$\mathcal{L}(\hat{y}_{out}^{PTE}, y^{LE})$ is minimized if and only if $\hat{y}_{out}^{PTE} P = y^{LE}$ for a permutation matrix P.*
2. *In this case, the matrix P is given by $P_{ij} := \exp[-\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i))]$.*

Proof. We first consider part (1), recalling that:

$$\mathcal{L}(\hat{y}_{out}^{PTE}, y^{LE}) := 1 - \text{mean}_i(\max_j(\exp[-\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i))])).$$

Since the BCE loss is minimized at 0, we have:

$$\begin{aligned}
\mathcal{L}(\hat{y}_{out}^{PTE}, y^{LE}) &= 1 - \text{mean}_i(\max_j(\exp[-\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i))])) \\
&\geq 1 - \text{mean}_i(\max_j(\exp[0])) \\
&= 0.
\end{aligned}$$

and equality holds if and only if $\max_j(\exp[-\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i))]) = 1$ for every i. This statement is true if and only if for every i there exists a j such that $\exp[-\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i))] = 1$, or equivalently such that $\text{BCE}(y^{LE}(j), \hat{y}_{out}^{PTE}(i)) = 0$.

Therefore, $\mathcal{L}$ reaches its minimum at 0 if and only if for every i there exists a j such that $y^{LE}(j) = \hat{y}^{PTE}_{out}(i)$, proving part (1).

We now prove part (2). Suppose that $\hat{y}^{PTE}_{out} P = y^{LE}$, and suppose that σ is the permutation defined by P, so that $\sigma(i^{PTE}) = j^{LE}$. Then, for each i, j, we have:

$$\mathrm{BCE}(y^{LE}(j), \hat{y}^{PTE}_{out}(i)) = \begin{cases} 0 & \text{if } \sigma(i) = j \\ +\infty & \text{otherwise,} \end{cases}$$

and therefore

$$\exp[-\mathrm{BCE}(y^{LE}(j), \hat{y}^{PTE}_{out}(i))] = \begin{cases} 1 & \text{if } \sigma(i) = j \\ 0 & \text{otherwise.} \end{cases}$$

This proves part (2). □

This proposition shows that by minimizing $\mathcal{L}$, we learn an approximate permutation matrix $\hat{P} \approx P$, given by:

$$\hat{P}_{ij} := \exp[-\mathrm{BCE}(y^{LE}(j), \hat{y}^{PTE}_{out}(i))]. \tag{3}$$

In practice, we do not minimize $\mathcal{L}$, since the max function presents an obstacle to effective training. Therefore, we relax the max operation to a function approxmax. This finally gives us our *Symbol Grounding Loss* $\mathcal{L}_{SG}$:

$$\mathcal{L}_{SG}(\hat{y}^{PTE}_{out}, y^{LE}) := 1 - \mathrm{mean}_i(\mathrm{approxmax}_j(\exp[-\mathrm{BCE}(y^{LE}(j), \hat{y}^{PTE}_{out}(i))])). \tag{4}$$

In our experiments, we set approxmax equal to the 2-norm; however, we did not find that performance was sensitive to the exact choice of approxmax, and other choices are also reasonable.

Having defined $\mathcal{L}_{SG}$, we incorporate it into our training pipeline as follows: First, we freeze the digit classifier $\mathcal{D}$, and train $\mathcal{S}$ under $\mathcal{L}_{SG}$. This begins to train $\mathcal{S}$ while also learning a permutation matrix $\hat{P} \approx P$ (defined by (3)). Note that since we are working with

the Ungrounded Visual Sudoku task, the permutation matrix is learned by means of SATNet itself, and it is impossible for labels to be leaked, since the training process does not even have access to labels for the input entries.

Second, once $\hat{P}$ has converged to a clear permutation matrix, we freeze this permutation and use it to align the PTE labels with the correct LE labels by multiplying the final outputs $\hat{y}_{out}^{PTE}$ by the learned $\hat{P}$. Now that the Symbol Grounding Loss is no longer needed, we switch to the traditional BCE loss and complete the training of $\mathcal{S}$, also unfreezing $\mathcal{D}$ to allow additional training.

3.2.3 Proofreading

The performance of a SATNet architecture can be improved by the addition of a *Proofreader* layer. This consists of a linear layer added just before the SATNet layer $\mathcal{S}$, initialized to a slightly noisy identity transform $\mathbb{R}^{N\times K} \to \mathbb{R}^{N\times K}$. (In the Sudoku case, $N = 81$ and $K = 9$.) We freeze the layers in the original model, and train only the proofreader layer. This is an optional final step resulting in a slight performance improvement. We find that the Proofreader layer also improves the performance of the original SATNet (with label leakage), in both the visual and nonvisual Sudoku settings.

3.3 Results

The above procedure allows us to achieve comparable results on an Ungrounded Visual Sudoku Dataset as the original SATNet architecture has in the grounded setting[5], with results being presented in Table 2. We may thus claim to solve the Symbol Grounding Problem in the case of Visual Sudoku.

All experiments were carried out on a Nvidia GTX1070 across 100 epochs, with each epoch taking roughly 2 minutes. The Adam optimiser was used with learning rate of $2 \times$

[5]Note that training under a grounded dataset is equivalent to the label leakage problem described in [11]

10^{-3} for the SATNet layer, and 10^{-5} for the digit classifier [53]. Standard deviations were calculated across 5 runs. We used the Sudoku Dataset made available under an MIT License from the original SATNet work [13].

Model Configuration	**Grounded vs. Ungrounded Data**	**Total Board Accuracy** (%)	**Per-Cell Accuracy** (%)	**Visual Accuracy** (%)
Original SATNet	grounded	66.5±1.0	98.8±0.1	99.0±0.0
Original SATNet	ungrounded	0±0.0	11.2±0.1	11.6±0.0
Our Method	**ungrounded**	**64.8±3.0**	**98.4±0.2**	**98.9±0.1**

Table 2: Performance of our method compared to the original SATNet architecture between grounded and ungrounded versions of the Visual Sudoku problem. Note that we distinguish the total board accuracy (how many 81-cell boards are completely correct) from per-cell accuracy (how many board cells are correct) and visual accuracy (how many input board cells are correct). Our method achieves comparable performance on a significantly more difficult version of the problem, thus solving the Symbol Grounding Problem.

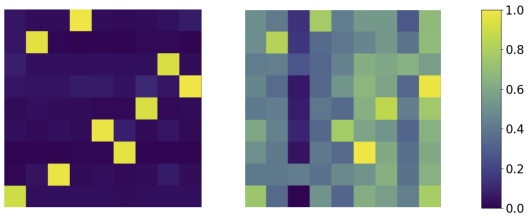

Figure 6: Permutation matrices extracted from the Symbol Grounding Loss function. On the left is a matrix extracted given a clustering with high accuracy, and the right matrix shows the results in a case where the clustering accuracy was below the necessary threshold (see Section 3.3.1).

During our pre-training pipeline, the clustering step achieves $95.6 \pm 0.4\%$ clustering ac-

curacy. Under the Symbol Grounding Loss, our self-grounded training achieves $22.3 \pm 1.0\%$ per-cell accuracy. One thing to note is that the self-grounded training step is susceptible to overfitting, and one needs to employ early stopping on the basis of per-cell error in order to learn the permutation matrix $\hat{P}$. See Figure 6 for an example of a learned $\hat{P}$ matrix.

Note that it is expected that the ungrounded version of the dataset will produce slightly worse results since it carries less information in its labels than its grounded counterpart. Another relevant aspect is that InfoGAN itself is sensitive to random seed. 4/10 runs converge to a clustering below the threshold necessary to ground symbols. We discuss this limitation further in Section 3.3.1.

3.3.1 Effect of Clustering Accuracy

An important ablation test to define some limitations of our approach is a study on the effect of clustering accuracy on our pre-training performance. It is difficult to measure this, as performance could vary based on the distribution of predictions across clusters, not only raw clustering accuracy. In this study we run our pipeline against InfoGAN at different stages of its training. In this way the cluster assignments start out uniform (based on noisy initialization) and gradually anneal to a $89.6 \pm 7.7\%$ accurate clustering. We find that our system requires roughly at least 88% clustering accuracy in order for the rest of the pipeline to progress. This is shown in Figure 7. While this is a notable limitation to our approach, solving Ungrounded Visual Sudoku was not at all possible with SATNet prior to this work. Furthermore, a threshold of 88% accuracy is not nearly as high as one might naively expect. Given that our input dataset contains on average 36.2 input cells per board, 88% digit classification accuracy gives less than a 0.1% chance of identifying an input board state perfectly with the initial clustering.

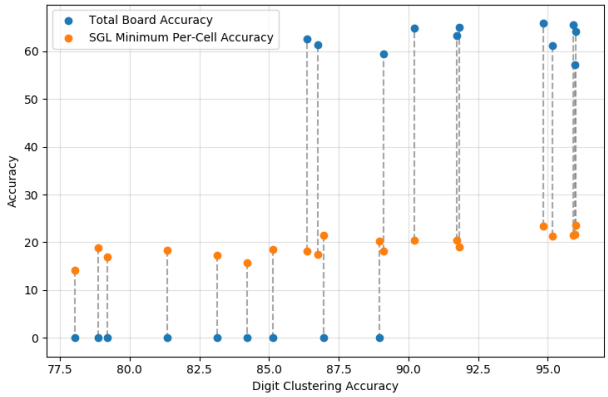

Figure 7: The effect of InfoGAN's clustering accuracy on our method's total board accuracy (blue) and per-cell accuracy during the Symbol Grounding Loss training phase (orange). Each pair of points connected by a dashed line indicates a different experiment. We note the sharp performance drop at roughly 88% clustering accuracy.

3.3.2 Effect of Distillation

In the case that the clustering algorithm used in our first pre-training phase is differentiable, the distillation step becomes optional. Despite this, it is desirable to distill our clustering model if there exists some smaller architecture that can achieve similar performance in the supervised setting. This is the case with InfoGAN, which in its standard form uses an architecture with 7,307,997 parameters. We distill this into a LeNet-derived architecture [51], with only 1,049,080 parameters and comparable performance, as shown in Table 3. Training speed changes from 602 ± 5 seconds/epoch to 255 ± 3 seconds/epoch between the two architectures.

3.3.3 Effect of Proofreading

Proofreading improves the performance of both visual and non-visual Sudoku, as seen in Table 4. We achieve the following results by training the proofreader with ungrounded

Digit Classifier	**Digit Clustering Accuracy** (%)
InfoGAN	89.6±7.7
Distilled LeNet	86.2±13.5

Table 3: The effect of distilling InfoGAN into a smaller LeNet-based convolutional architecture. InfoGAN performance has a tendency to plateau at different levels based on seed. Here we show performance across all runs, whereas successful ones are used in the downstream pipeline. A "successful" InfoGAN run will plateau at roughly 95% accuracy.

Datasets even if the original model which it augments was trained with the grounded version.

Model Configuration	**Proofreader Present?**	**Total Board Accuracy** (%)	**Per-Cell Accuracy** (%)	**Visual Accuracy** (%)
Original Non-visual	no	96.6±0.3	**99.9±0.0**	N/A
Original Non-visual	**yes**	**97.1±0.3**	**99.9±0.0**	N/A
Original Visual	no	66.5±1.0	**98.8±0.1**	**99.0±0.0**
Original Visual	**yes**	**67.6±1.2**	98.6±0.1	**99.0±0.0**
Our Method	no	62.8±3.2	**98.6±0.1**	**98.9±0.1**
Our Method	**yes**	**64.8±3.0**	98.4±0.2	**98.9±0.1**

Table 4: The effect of adding a proofreading layer to the original versions of SATNet for both visual and non-visual Sudoku datasets, as well as the pre-training method proposed in this paper. We show that a proofreader uniformly improves the Total Board Accuracy of SATNet.

We note that the numbers above from the original architectures reflect our reproduction of the results in the original paper. Please see Appendix A for further details.

3.4 Discussion

3.4.1 Sensitivity of SATNet to Random Seeds

It was described in [11] that SATNet exhibits a high sensitivity to the choice of random seed. For instance, 8 out of 10 random seeds would fail even with label leakage. While we initially reproduced this behavior, such sensitivity can in fact be circumvented with a minor correction to the PyTorch implementation, detailed further in Appendix A. We use the corrected, stable model for comparison in all our results.

3.4.2 Incorrect Upper Performance Bound

In the original SATNet paper, it is argued that the performance of the visual Sudoku model is bound by the probability of identifying all the input cells on a particular board correctly. Thus when using LeNet, which has a classification accuracy of 99.2%, the best performance we can expect on our dataset with 36.2 input cells on average is $0.992^{36.2} = 74.8\%$ [51, 13]. This is not exactly accurate.

It is not necessarily true that the SATNet layer cannot solve a board correctly if some number of input cells are wrong. Intuitively, if one finds two of the same numbers as inputs in a row of a Sudoku puzzle, one can infer that one of those inputs might have been classified incorrectly. This can then be used to make an educated guess about the correct final board state. We are able to show that the SATNet layer is actually able to reason about incorrect input cells to a certain extent. Interestingly, SATNet's ability to reason is affected by whether an incorrectly labeled digit results in an unsolvable board or not. It is also affected by the presence of a Proofreader layer. Details on these experiments can be found in Appendix B.

While the upper bound posed originally may not be strictly correct, it is still a good guideline. Deriving a strict upper performance bound is likely quite difficult as the mathematics of logical problems such as Sudoku are not fully understood.

4 Limitations & Future Work

4.1 Learning Complexity Bound Strength

In section 2.3.2 we discussed a pair of complexity bounds which show that learning ungrounded datasets is at least as hard as learning grounded ones, and that training both neural and connectionist models is at least as difficult as training both in isolation.

We do however believe that stronger bounds could be derived for these problems. Our empirical results, as well as the findings in [11] seem to indicate that symbol grounding for visual sudoku is *strictly* harder in the ungrounded case. Further investigation is required in order to prove this, however.

4.2 Clustering Limitations

While our method is able to address a new class of Visual MAXSAT problems with SATNet, it is limited by the need to prime the digit classifier with correct data clusters (see Section 3.3.1). This imposes a constraint on which datasets can be used as visual inputs to this pipeline. One facet of this limitation is the fact that the current Symbol Grounding Loss function only supports inferring a permutation between the pre-trained encoding and the label encoding. This means that if there are K label classes, the clustering algorithm must cluster the input data accurately in K clusters. One might imagine allowing the Symbol Grounding Loss to support a more general surjective mapping between encoding domains, allowing for a higher number of clusters (and consequently a higher accuracy).

A second limitation is the tendency of the Symbol Grounding Loss to overfit somewhat quickly. While we experimented with several loss function formulations, further experimentation may prove useful, potentially with the inclusion of regularizers.

We believe neither of these limitations are fundamental; future investigation may help to alleviate them.

5 Conclusion

Our work lays out a foundation for understanding symbol grounding in the context of machine learning applications. We formalize the phenomenon, and discuss how different aspects of symbol grounding affect learning difficulty; Such as the distinction between grounded and ungrounded datasets, as well as the difference between fully and partially learnable architectures. We then demonstrate a self-supervised pre-training method for solving Ungrounded Visual Sudoku. The ability to solve the more difficult Ungrounded Visual MAXSAT problems contrasts markedly with the previous state of the art, which was unable to surpass 0% accuracy on these tasks. This work extends the current state of the art for logical constraint-learning neurosymbolic methods, a promising area of research which boasts the potential to dramatically broaden the range of problems which machine learning can address.

References

[1] Stevan Harnad. The symbol grounding problem. *Physica D Nonlinear Phenomena*, 42(1-3):335–346, June 1990.

[2] Jonathan St B. T. Evans. Heuristic and analytic processes in reasoning*. *British Journal of Psychology*, 75(4):451–468, 1984.

[3] Daniel Kahneman. *Thinking Fast and Slow*. Anchor Canada, 2013.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *Preprint arXiv:1902.06705*, 2019.

[6] Samuel Henrique Silva and Peyman Najafirad. Opportunities and Challenges in Deep Learning Adversarial Robustness: A Survey. *Preprint arXiv:2007.00753*, 2020.

[7] Erico Tjoa and Cuntai Guan. A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI. *Preprint arXiv:1907.07374*, 2019.

[8] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of Gradient-Based Deep Learning. In *International Conference on Machine Learning (ICML)*, 2017.

[9] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd Wave. *Preprint arXiv:2012.05876*, 2020.

[10] John Haugeland. *Artificial Intelligence: the Very Idea*. MIT Press, 1985.

[11] Oscar Chang, Lampros Flokas, Hod Lipson, and Michael Spranger. Assessing SATNet's ability to solve the symbol grounding problem. In H. Larochelle, M. Ranzato, R. Hadsell,

M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[12] David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 511–520. PMLR, 10–15 Jul 2018.

[13] Po-Wei Wang, Priya L. Donti, Bryan Wilder, and Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning (ICML)*, 2019.

[14] Sever Topan, David Rolnick, and Xujie Si. Techniques for Symbol Grounding with SATNet. *Neural Information Processing Systems 2021*, 2021.

[15] Angelo Cangelosi. Solutions and open challenges for the symbol grounding problem. *Int. J. Signs Semiot. Syst.*, 1:49–54, 2011.

[16] Luc Steels. The symbol grounding problem has been solved. so what's next? *Symbols, Embodiment and Meaning. Oxford University Press, Oxford, UK*, 01 2007.

[17] Mariarosaria Taddeo and Luciano Floridi. A praxical solution of the symbol grounding problem. *Minds and Machines*, 17(4):369–389, Dec 2007.

[18] Mariarosaria Taddeo and Luciano Floridi. Solving the symbol grounding problem: a critical review of fifteen years of research. *Journal of Experimental & Theoretical Artificial Intelligence*, 17(4):419–445, 2005.

[19] Selmer Bringsjord. The symbol grounding problem ... remains unsolved. *Journal of Experimental & Theoretical Artificial Intelligence*, 27(1):63–72, 2015.

[20] Krystyna Bielecka. Why taddeo and floridi did not solve the symbol grounding problem. *Journal of Experimental & Theoretical Artificial Intelligence*, 27(1):79–93, 2015.

[21] Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robotics and autonomous systems*, 43(2-3):85–96, 2003.

[22] Silvia Coradeschi, Amy Loutfi, and Britta Wrede. A short review of symbol grounding in robotic and intelligent systems. *KI - $Ku_nstlicheIntelligenz, 27 : 129 - -136, 052013$.*

[23] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *2010 IEEE International Conference on Robotics and Automation*, pages 1486–1491, 2010.

[24] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

[25] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *International Conference on Learning Representations 2019*, 2019.

[26] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[27] Tao Meng and Kai-Wei Chang. An Integer Linear Programming Framework for Mining Constraints from Data. *Preprint arXiv:2006.10836*, 2020.

[28] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[29] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-End Constrained Optimization Learning: A Survey. *Preprint arXiv:2103.16378*, 2021.

[30] Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *International Conference on Machine Learning (ICML)*, 2017.

[31] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning (ICML)*, 2020.

[32] Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*, 2020.

[33] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT Solver from Single-Bit Supervision. In *International Conference on Learning Representations (ICLR)*, 2019.

[34] Wang-Zhou Dai, Qiu-Ling Xu, Yang Yu, and Zhi-Hua Zhou. Tunneling Neural Perception and Logic Reasoning through Abductive Learning. *Preprint arXiv:1802.01173*, 2018.

[35] Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. Recurrent Relational Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[36] Richard Evans and Edward Grefenstette. Learning Explanatory Rules from Noisy Data. *Journal of Artificial Intelligence Research*, 2017.

[37] Sebastian Nowozin and Christoph Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6:185–365, 01 2011.

[38] Christian Bessière, Abderrazak Daoudi, Emmanuel Hébrard, George Katsirelos, Nadjib Lazaar, Younes Mechqrane, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. New Approaches to Constraint Acquisition. In *Data Mining and Constraint Programming*, volume 10101 of *Lecture Notes in Computer Science*, pages 51–76. Springer International Publishing AG, 2016.

[39] Mokanarangan Thayaparan, Marco Valentino, Deborah Ferreira, Julia Rozanova, and André Freitas. ∂-Explainer: Abductive Natural Language Inference via Differentiable Convex Optimization. *Preprint arXiv:2105.03417*, 2021.

[40] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput*, 18(7):1527–1554, Jul 2006.

[41] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

[42] Huanru Henry Mao. A Survey on Self-supervised Pre-training for Sequential Transfer Learning in Neural Networks. *Preprint arXiv:2007.00800*, 2020.

[43] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised Pre-Training of Image Features on Non-Curated Data. In *International Conference on Computer Vision (ICCV)*, 2019.

[44] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, Jun 2015.

[45] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[46] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[47] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *Preprint arXiv:1503.02531*, 2015.

[48] Lei Jimmy Ba and Rich Caruana. Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[49] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation. In *International Conference on Learning Representations (ICLR)*, 2016.

[50] Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do Deep Convolutional Nets Really Need to be Deep and Convolutional? In *International Conference on Learning Representations (ICLR)*, 2016.

[51] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[52] Luis C. Lamb, Artur Garcez, Marco Gori, Marcelo Prates, Pedro Avelar, and Moshe Vardi. Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

A Random Seed Sensitivity Fix

There are two aspects of the implementation which alleviate the sensitivity of SATNet on random seed. Please note that for this section we discuss the SATNet architecture from the original paper trained on an Grounded Visual Sudoku dataset [13]. The first was a minor bug in the CUDA implementation of SATNet's backprop calculation. The second relates to how supervision is provided on the digit classifier $\mathcal{D}$ during training.

Recall in section 3.1.1 that we can partition any Grounded Visual MAXSAT labels into input and output variable labels. Let us also reference our notation from section 3.2, where we have our digit classifier $\mathcal{D}(x) = \hat{y}_{in}^{PTE}$ and our SATNet layer $\mathcal{S}(\hat{y}_{in}^{PTE}) = \hat{y}_{out}^{PTE}$. We essentially have two options for returning the architecture's predictions for the input variables, as they are present in both $\hat{y}_{in}^{PTE}$ and $\hat{y}_{out}^{PTE}$. The choice of which of these to return results in a significant performance difference. The original SATNet model returned the input variable predictions from $\hat{y}_{out}^{PTE}$, whereas we return the ones in $\hat{y}_{in}^{PTE}$. Note that since the input variables are held constant in the SATNet layer $\mathcal{S}$, the nominal value of the input variables is the same between these two returns. The only difference is the implication of how gradients are computed.

B Effect of Error Injection for Visual Sudoku

We construct an experiment by running the nonvisual Sudoku model, and perturbing input cell labels in the test datasets. Under no change to the original architecture, a SATNet layer trained on correct inputs is able to solve some small percentage of the boards with erroneous inputs. Interestingly, SATNet's ability to solve these puzzles depends on whether the error injection resulted in an unsolvable board or not. While adding a proofreader improves performance under normal circumstances, in the presence of injected errors it worsens performance.

Number of Injected Input	**Total Board Accuracy** (%)	
Cell Errors per Board	**Solvable**	**Unsolvable**
0	96.6 ± 0.3	96.6 ± 0.3
1	3.2 ± 2.0	0.0 ± 0.0
2	0.1 ± 0.0	0.0 ± 0.0
3	0.0 ± 0.0	0.0 ± 0.0

Table 5: Solvable/Unsolvable split under error injection for traditional nonvisual Sudoku solver. Some boards are still solved by SATNet even when not all input cells are correct.

Number of Injected Input	Total Board Accuracy (%)	
Cell Errors per Board	Solvable	Unsolvable
0	97.1 ± 0.3	97.1 ± 0.3
1	1.9 ± 0.0	0.0 ± 0.0
2	0.0 ± 0.0	0.0 ± 0.0
3	0.0 ± 0.0	0.0 ± 0.0

Table 6: Solvable/Unsolvable split under error injection for nonvisual Sudoku solver with proofreader.

C Codebases used for Experimentation

Our experiments are built on top of two codebases. We leverage the SATNet implementation provided by the original authors at `https://github.com/locuslab/SATNet`, in addition to an InfoGAN implementation available at `https://github.com/Natsu6767/InfoGAN-PyTorch`. We have made our implementation public at `https://github.com/SeverTopan/SATNet`.