

Deep Adaptive Semantic Logic (DASL): Compiling Declarative Knowledge into Deep Neural Networks

Karan Sikka¹ Andrew Silberfarb¹ John Byrnes¹
 Indranil Sur¹ Ed Chow¹ Ajay Divakaran¹
 Richard Rohwer¹

Abstract

We introduce **Deep Adaptive Semantic Logic (DASL)**, a novel framework for automating the generation of deep neural networks that incorporates user-provided formal knowledge to improve learning from data. We provide formal semantics that demonstrate that our knowledge representation captures all of first order logic and that finite sampling from infinite domains converges to correct truth values. DASL's representation improves on prior neural-symbolic work by avoiding vanishing gradients, allowing deeper logical structure, and enabling richer interactions between the knowledge and learning components. We illustrate DASL through a toy problem in which we add structure to an image classification problem and demonstrate that knowledge of that structure reduces data requirements by a factor of 1000. We then evaluate DASL on a visual relationship detection task and demonstrate that the addition of commonsense knowledge improves performance by 10.7% in a data scarce setting.

1. Introduction

Early work on Artificial Intelligence focused on Knowledge Representation and Reasoning (KRR) through the application of techniques from mathematical logic (Genesereth & Nilsson, 1987). The compositionality of KRR techniques provides expressive power for capturing expert knowledge in the form of rules or assertions (*declarative knowledge*), but they are brittle and unable to generalize or scale. Recent work has focused on Deep Learning (DL), in which the parameters of complex functions are estimated from data (LeCun et al., 2015). DL techniques learn to recognize patterns not easily captured by rules and generalize well

¹SRI International, USA. Correspondence to: Karan Sikka <karan.sikka@sri.com>, John Byrnes <john.byrnes@sri.com>.

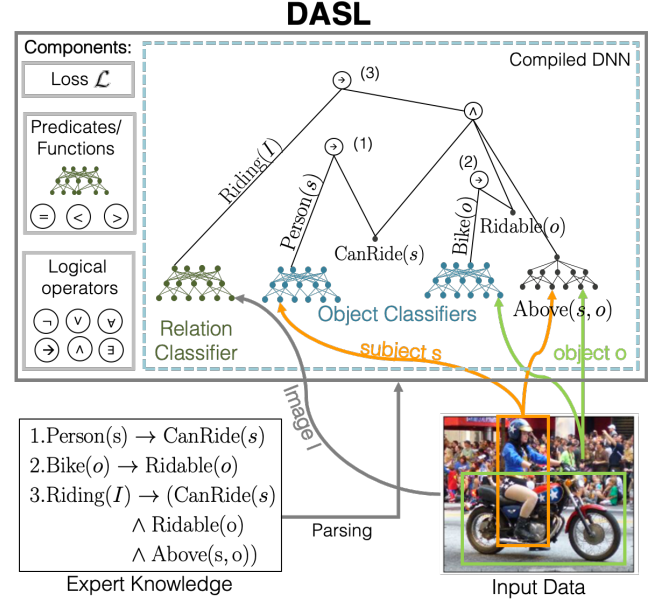


Figure 1. DASL integrates user-provided expert knowledge with training data to learn DNNs. It achieves this by compiling a DNN from knowledge, expressed in first order logic, and domain-specific neural components. This DNN is trained using backpropagation, fitting both the data and knowledge. Here DASL applies commonsense knowledge to the visual relationship detection task. \wedge and \rightarrow refer to ‘and’ and ‘implies’ connectives respectively.

from data, but they often require large amounts of data for learning and in most cases do not reason at all (Garcez et al., 2012; Marcus, 2018; Weiss et al., 2016; Yang et al., 2017). In this paper we present **Deep Adaptive Semantic Logic (DASL)**, a framework that attempts to take advantage of the complementary strengths of KRR and DL by fitting a model simultaneously to data and declarative knowledge. DASL enables robust abstract reasoning and application of domain knowledge to reduce data requirements and control model generalization.

DASL represents declarative knowledge as assertions in first order logic. The relations and functions that make up the vocabulary of the domain are implemented by neural networks that can have arbitrary structure. The logical connectives in the assertions compose these networks into a single deep network which is trained to maximize their truth.

Figure 1 provides an example network that implements a simple rule set through composition of network components performing image classification. Logical quantifiers “for all” and “there exists” generate subsamples of the data on which the network is trained. DASL treats labels like assertions about data, removing any distinction between knowledge and data. This provides a mechanism by which supervised, semi-supervised, unsupervised, and distantly supervised learning can take place simultaneously in a single network under a single training regime.

The field of neural-symbolic computing (Garcez et al., 2019) focuses on combining logical and neural network techniques in general, and the approach of (Serafini & Garcez, 2016) may be the closest of any prior work to DASL. To generate differentiable functions to support backpropagation, these approaches replace pure Boolean values of 0 and 1 for *True* and *False* with continuous values from $[0, 1]$ and select fuzzy logic operators for implementing the Boolean connectives. These operators generally employ maximum or minimum functions, removing all gradient information at the limits, or else they use a product, which drives derivatives toward 0 so that there is very little gradient for learning. DASL circumvents these issues by using a logit representation of truth values, for which the range is all real numbers.

Approaches to knowledge representation, both in classical AI and in neural-symbolic computing, often restrict the language to fragments of first order logic (FOL) in order to reduce computational complexity. We demonstrate that DASL captures full FOL with arbitrary nested quantifiers, function symbols, and equality by providing a single formal semantics that unifies DASL models with classical Tarski-style model theory (Chang & Keisler, 1973). We show that DASL is sound and complete for full FOL. FOL requires infinite models in general, but we show that iterated finite sampling converges to correct truth values in the limit.

In this paper we show an application of DASL to learning from small amounts of data for two computer vision problems. The first problem is an illustrative toy problem based on the MNIST handwritten digit classification problem. The second is a well-known challenge problem of detecting visual relationships in images. In both cases, we demonstrate that the addition of declarative knowledge improves the performance of a vanilla DL model. This paper makes the following contributions:

1. The novel framework DASL, which compiles a network from declarative knowledge and bespoke domain-specific reusable component networks, enabling gradient-based learning of model components;
2. Grounding of the proposed framework in model theory, formally proving its soundness and completeness for full first order logic;

3. A logit representation of truth values that avoids vanishing gradients and allows deep logical structures for neural-symbolic systems;
4. Syntactic extensions that allow (i) restricted quantification over predicates and functions without violating first order logic constraints, and (ii) novel hybrid network architectures;
5. Evaluation on two computer vision problems with limited training data, demonstrating that knowledge reduces data requirements for learning deep models.

2. Related Work

Neural-Symbolic Computing: Early efforts to augment DNNs with logic focused on propositional logic, which supports only logical connectives between (atomic) propositions (Garcez et al., 2012; 2019). For example, KBANN (Towell & Shavlik, 1994) maps a set of propositions into a graph, constructs a neural network, and then trains it. DASL follows this basic idea but fully supports full first order logic (FOL) as well as arithmetic expressions.

Similar to several prior efforts (Hu et al., 2016; Li & Srikumar, 2019; Rocktäschel et al., 2015), DASL replaces Booleans with real-valued *pseudo-probabilities* to make the logical operations differentiable. This circumstance has motivated the invention of a collection of *ad hoc* aggregation operators for representing logical connectives (Detyniecki, 2001). These include the *t-norm*, used by Logic Tensor Networks (LTNs) (Serafini & Garcez, 2016) and the above works. Instead, DASL uses a logit representation for truth values, whose range is all real numbers, which avoids vanishing gradients and enables learning with deeper logical structures. DASL also differs in supporting multiple entity types, arithmetic, and non-traditional operations such as *softmax* that enable richer interaction between the NN and knowledge (Section 4). DASL also represents the first time that soundness and completeness have been established for a FOL system applied to neural networks.

Compositional DL: DASL is related to works that execute a task by composing trainable neural modules by parsing a query (in natural language) (Andreas et al., 2016; Mao et al., 2019; Yi et al., 2018a;b). For example, (Yi et al., 2018b) focuses on visual question answering and employs a differentiable tree-structured logic representation, similar to DASL, but only in order to learn to translate questions into retrieval operations, whereas DASL learns the semantics of the application domain and can also integrate useful domain knowledge.

Structured Learning: Other work also exploits underlying structure in the data or the label space to learn DNNs using techniques such as conditional random fields, graph neural

networks, attention models, etc. (Battaglia et al., 2018; Belanger et al., 2017; Kim et al., 2017; Peng et al., 2018; Zheng et al., 2015). These methods impose structure by either adapting the DNN architecture (Battaglia et al., 2018) or the loss function (Zheng et al., 2015). DASL instead imposes soft constraints by compiling DNNs based on rules that can be stated in a flexible manner using FOL.

Weakly supervised learning: DASL is related to works that use structural constraints as side-information or implicit knowledge to improve learning, particularly in data scarce conditions (Chang et al., 2012; Hu et al., 2016; Oquab et al., 2014; Rocktäschel & Riedel, 2017; Stewart & Ermon, 2017; Xing et al., 2003).

Semantic Reasoning: By the *semantics* of a logical language we mean an interpretation of its symbols (which do not include logical connectives and quantifiers); a *model* in the sense of model theory (Weiss & D’Mello, 1997). In common with several methods (Xie et al., 2019), DASL grounds its entities in vector spaces (embeddings) and its predicates and functions in trainable modules. DASL builds on prior works on semantic representation techniques (Deerwester et al., 1990; Mikolov et al., 2013; Pennington et al., 2014) by enabling logical statements to modify the entity embeddings so as to mirror semantic similarity in the application.

Traditional theorem provers (Siekmann & Wrightson, 1983) operate at a purely syntactic level to derive statements that hold true regardless of the underlying semantics. This approach often fails catastrophically when its users fail to supply complete, accurate and consistent logical descriptions of their applications. Approaches such as DASL that incorporate semantic representations address this problem by treating the logic, like data, as merely suggestive. An intermediate approach (Cohen et al., 2017; de Jong & Sha, 2019; Rocktäschel & Riedel, 2017) applies a theorem prover to a query in order to generate a proof tree, which is then used to build a corresponding DNN. Such methods can benefit from ‘soft unification’ in which proof steps can be connected via entities that nearly match semantically, rather than symbols that match exactly or not at all.

Bayesian Belief Networks: Substitution of pseudo-probabilities for Booleans fails to capture uncertainty the way fully Bayesian methods do (Jaynes, 2003). Bayesian Belief networks (Pearl, 2009) accurately represent probabilities but lack expressivity and face computability challenges. Bayes nets are most comfortably confined to propositional logic. Efforts to extend them to first-order logic include Markov Logic Networks (Richardson & Domingos, 2006), which use an undirected network to represent a distribution over a set of models, *i.e.*, *groundings* or *worlds* that can interpret a theory. The lifted inference approach (Kimmig et al., 2004) reasons over populations of entities to render

the grounded theory computationally tractable. These methods generally do not support the concept of (continuous) soft semantics through the use of semantic embedding spaces, as DASL does.

3. Approach

In this section we describe our approach to integrate data with relevant expert knowledge. Consider the task, depicted in Figure 1, of predicting the relationship between bounding boxes containing a subject and an object. In addition to learning from labeled training samples, we want to incorporate the commonsense knowledge that if the predicted relationship is “Riding” then the subject must be able to ride, the object must be rideable, and the subject must be above the object. Incorporating such knowledge results in a more robust model that uses high-level semantics to improve generalization and learn from a small number of examples. DASL achieves integration of the continuous representations in DNNs with the discrete representations typically used for knowledge representation by compiling a DNN from the knowledge assertions and grounding the vocabulary of the domain in component networks, enabling gradient-based learning of the model parameters.

We begin by providing the theoretic underpinning of DASL in FOL. We then describe the underlying representations of the DASL model including: model components, language elements, etc., which ground the formal language and allow end-to-end learning of model parameters.

3.1. DASL Model Theory

A DASL theory is specified in a language \mathcal{L} containing constants a_0, \dots , function symbols f_0, \dots , and relation symbols R_0, \dots . In addition, we have variables x_0, \dots understood to range over objects of some universe, logical connectives \neg (‘not’) and \wedge (‘and’), the quantifier \forall signifying ‘for all’, and the single logical binary relation symbol ‘=’ indicating equality. For presentation purposes we treat \vee (‘or’), \rightarrow (‘implies’), and \exists (‘there exists’) as defined in terms of \neg , \wedge , and \forall (although they are implemented as first class connectives in DASL). Constants and variables are *terms*; an n -ary function symbol applied to n -many terms is a *term*. An n -ary relation symbol (including equality) applied to n -many terms is a *formula*; if ϕ and ψ are formulas and x is a variable then $(\forall x)\phi$, $\neg\phi$, and $\phi \wedge \psi$ are *formulas*.

Formal semantics for \mathcal{L} are provided by structures interpreting the symbols of \mathcal{L} . We generalize the typical Tarski-style (Weiss & D’Mello, 1997) model theoretic definitions to capture DASL models. A model maps every term to an element of the domain and every formula to a *truth value*. In classical semantics a model maps every formula either to *True* or to *False*, frequently represented as 1 and 0. To apply

general optimization techniques such as stochastic gradient descent (SGD), we define the truth values to be the closed real interval $[0, 1]$, denoted as \mathbb{T} .

We begin by specifying a class of objects A , the *domain* of the model. A *variable map* V for A maps variables in \mathcal{L} into A . An *interpretation* for \mathcal{L} and A is a structure $\mathcal{I} = (I, \mathcal{I}_{f_0}, \dots, \mathcal{I}_{R_0}, \dots)$ such that I maps constants into A , $\mathcal{I}_{f_i} : A^{m_i} \rightarrow A$ and $\mathcal{I}_{R_j} : A^{n_j} \rightarrow \mathbb{T}$ for each i and j where m_i and n_j are the arities of functions f_i and R_j . Given \mathcal{I} and V for A , $\mathfrak{A} = (A, \mathcal{I}, V)$ is called a *model* for \mathcal{L} .

We use connectives to define functions on truth values. For truth values t_1, t_2, \dots we define $\neg t_1 = 1 - t_1$, $t_1 \wedge t_2 = t_1 \cdot t_2$, and $\forall_i t_i = \prod_i t_i$. We also allow different definitions of these functions. We interpret $=$ using a function $\mathfrak{D}_=$ on objects $u, v \in A$ such that $\mathfrak{D}_=[u, v] \in \mathbb{T}$ and is 1 if and only if $u = v$. Finally, we define a *sampling function* \mathcal{S} that maps the domain A to an arbitrary subset of A .

Given $\mathfrak{A} = (A, \mathcal{I}, V)$, variable x , and $u \in A$, $\mathfrak{A}_{u/x}$ is the model (A, \mathcal{I}, V^*) where $V^*(x) = u$ and $V^*(y) = V(y)$ for y other than x . We now define interpretation in \mathfrak{A} of variable x , constant a , term t_1, \dots, t_n , function symbol f , relation symbol R , and formulas ϕ and ψ , all from \mathcal{L} , by the following inductive definition:

$$\begin{aligned} \mathfrak{A}[x] &= V(x) \\ \mathfrak{A}[a] &= I(a) \\ \mathfrak{A}[f(t_1, \dots, t_n)] &= \mathcal{I}_f(\mathfrak{A}[t_1], \dots, \mathfrak{A}[t_n]) \\ \mathfrak{A}[R(t_1, \dots, t_n)] &= \mathcal{I}_R(\mathfrak{A}[t_1], \dots, \mathfrak{A}[t_n]) \\ \mathfrak{A}[t_1 = t_2] &= \mathfrak{D}_=(\mathfrak{A}[t_1], \mathfrak{A}[t_2]) \\ \mathfrak{A}[\neg \phi] &= \neg \mathfrak{A}[\phi] \\ \mathfrak{A}[\phi \wedge \psi] &= \mathfrak{A}[\phi] \wedge \mathfrak{A}[\psi] \\ \mathfrak{A}[(\forall x)\phi] &= \forall_{u \in \mathcal{S}(A)} \mathfrak{A}_{u/x}[\phi] \end{aligned}$$

When $\mathfrak{A}[\phi] = 1$ we write $\mathfrak{A} \models \phi$ and we say \mathfrak{A} is a model of ϕ and satisfies ϕ . If Γ is a set of formulas and $\mathfrak{A} \models \phi$ for every $\phi \in \Gamma$ then we write $\mathfrak{A} \models \Gamma$.

The standard semantics from model theory are achieved when the range of $\mathfrak{D}_=$ and \mathcal{I}_R is $\{0, 1\}$ and when $\mathcal{S}(A) = A$. For basic DASL semantics, $A = \mathbb{R}^N$ for some fixed N . DASL also extends to many-sorted logic; *i.e.*, when bound variables have types, the single universe A is replaced by a universe $A_i = \mathbb{R}^{N_i}$ for each sort and supported by the above definitions. We allow the sampling function $\mathcal{S}(A)$ to return different samples on different invocations. The mapping I from constants to A is referred to as *embedding* (as done in deep learning). A function \mathcal{L} that maps sequences of truth values to non-negative reals is a *DASL loss function* if $\mathcal{L}(\langle 1 \rangle) = 0$ and \mathcal{L} is monotonic in every element of its input sequence. We define $\mathfrak{A} \models_{\theta} \Gamma$ whenever $\mathcal{L}(\mathfrak{A}[\Gamma]) \leq \theta$. Thus $\mathfrak{A} \models_0 \Gamma$ is equivalent to $\mathfrak{A} \models \Gamma$.

DASL is sound and complete: To prove formally that DASL models capture full first order logic, we show that for any set of formulas Γ there is a Tarski model $\mathfrak{A} \models \Gamma$ if and only if there is a DASL model $\mathfrak{B} \models \Gamma$. By the definitions provided, a DASL model can be constructed equivalent to any given Tarski model by replacing objects by vectors and sets by functions. Since we do not restrict the class of functions, this is trivial. When a DASL model has 0 loss for Γ , construction of a Tarski model is straightforward as well.

The more interesting questions come when we restrict DASL to what is computationally feasible and when we generalize to $\mathfrak{A} \models_{\theta} \Gamma$. Suppose the domain A of \mathfrak{A} can be expressed as $A_1 \cup A_2 \dots$ where the disjoint A_i are all finite and of fixed cardinality. If $\mathcal{L}(A_i \cup A_j) = \frac{1}{2}\mathcal{L}(A_i) + \frac{1}{2}\mathcal{L}(A_j)$ for all $i \neq j$ then $\sum_i \mathcal{L}(\mathfrak{A}[\Gamma, A_i]) = \mathcal{L}(\mathfrak{A}[\Gamma])$ (where we oversimplify slightly by omitting the details of defining average loss over infinite domains). Thus, even for a finite sampling function from an infinitary domain, we compute correct loss in the limit when repeated applications of the sampler yield A_1, A_2, \dots . When the interpretation functions of \mathcal{I} are implemented as neural networks, there will be restrictions on the classes of functions and relations that can be computed, and these have been well-studied.

3.2. DASL Models as Neural Networks

Given a DASL theory Γ , DASL searches for a model \mathfrak{A} that satisfies Γ with minimal loss. Γ in general contains both data for a standard machine learning task and knowledge assertions such as those in Figure 1. We implement DASL in the popular deep learning library PyTorch (Paszke et al., 2019). The DASL semantics defined above are both compositional and a function of the syntax of Γ , at least down to choice of \mathcal{I} . Since neural networks are also compositional, DASL constructs independent networks for each function in \mathcal{I} and assembles these into a single network based on the parse tree of Γ . This makes DASL compositional, where DNNs are assembled on the fly based on the theory. We then use backpropagation through the tree to minimize the loss to learn the model parameters. We next describe details for the internal representations, implementation of language elements, optimization, and extensions of logical language.

Representation of model components: Implementation of a DASL model requires specification of a domain A_i for each logical type. The domains can include both parameterized members and static members, which can be continuous (*e.g.* visual embeddings), ordinal and categorical (*e.g.* labels) types. For each domain A_i having elements represented by constants, we need to specify the embedding I_i for the constants. Any neural network implementable in PyTorch can be used to realize \mathcal{I}_f and \mathcal{I}_R .

DASL works with logits rather than truth values. The logit for a truth value $t \in \mathbb{T}$ is calculated as $\text{logit}(t) = \ln \frac{t}{1-t}$.

and its inverse is a sigmoid non-linearity ($t = \sigma(\text{logit}(t))$).

Implementation of the logical connectives: For truth values t_1 and t_2 and corresponding logits l_1 and l_2 , we define negation (\neg) and conjunction (\wedge) operators as:

$$\neg l_1 = \text{logit}(1 - t_1) = -l_1$$

$$l_1 \wedge l_2 = \text{logit}(t_1 t_2) = \ln \sigma(l_1) + \ln \sigma(l_2) - \ln(1 - \sigma(l_1)\sigma(l_2))$$

This formula for \wedge is numerically unstable when $t_1 t_2$ gets close to 1. Whenever this occurs, we instead use the numerically stable approximation:

$$l_1 \wedge l_2 \approx -\ln(e^{-l_1} + e^{-l_2}).$$

We use PyTorch functions `logsigmoid` and `logsumexp` that provide efficient and numerically robust computations for terms arising in these equations.

Conjunction and universal quantification are naturally represented as products of truth values, but the product of a large number of positive terms all less than 1 gets arbitrarily close to 0, and so does its derivative, meaning that learning is slow or will stop altogether. Under the logit space equations above, however, conjunctions are sums, so increasing the number of conjuncts does not diminish the gradient. Two typical alternatives for $t_1 \wedge t_2$ in systems that operate directly on truth values are $\min(t_1, t_2)$ and $\max(0, t_1 + t_2 - 1)$ (Serafini & Garcez, 2016). When many terms are conjoined, the former formula yields gradient information only for the minimal truth value, and the second yields no gradient information at all whenever $t_1 + t_2 < 1$, again restricting the ability of a system to learn.

Equality: DASL functions can include standard regression tasks $f(x) = y$ for real-valued y . The behavior of DASL on such rules is governed by $\mathcal{D}_=(f(x), y)$, so $\mathcal{D}_=$ needs to be a function that allows for backpropagation. Since we reason in logit space, $\mathcal{D}_=$ cannot be implemented as mean squared error since its logit would rapidly diverge towards infinity as the error gets small. Instead we take the logit transform to be a log likelihood $\ln(\frac{\text{Pr}(u=v)}{\text{Pr}(u \neq v)})$ and we model $d = u - v$ as normally distributed noise when u and v are “equal” (with mean 0 and variance ε^2) and as normally distributed distance when u and v are genuinely different (with mean μ and variance σ^2). Ignoring the scaling factor, the density for $x = |d|$ in the latter case is given by $e^{-(x-\mu)^2/2\sigma^2} + e^{-(x+\mu)^2/2\sigma^2}$. Using the ratio of these densities in place of the ratio of probabilities, we derive: $\text{logit}(\mathcal{D}_=(u, v)) = \ln \frac{2\sigma}{\varepsilon} + \frac{x^2}{2\varepsilon^2} - \ln(e^{-(x-\mu)^2/2\sigma^2} + e^{-(x+\mu)^2/2\sigma^2})$ When u and v are vectors rather than scalars, we can use the same distribution on $\|u - v\|$.

Quantifiers and sampling: As mentioned previously, the sampler may return different samples on different

invocations. A sampler is implemented as a PyTorch `dataloader`, so returned samples are similar to mini-batches in SGD. The types of quantified variables are always specified, and may be drawn from a fixed table of values (such as images), a fixed table of parameterized values (vectors to be learned), or all vectors in a vector space. A sampler is defined and can be customized for each quantifier instance, so that different quantifiers over the same type can sample differently. When quantifiers are nested, samples obtained by outer samplers are available as input to inner samplers. For example, in $(\forall x : T_1)(\exists y : T_2)\phi$, the sampler which selects some set of y ’s from T_2 may rely on x to determine which y ’s to sample. In this sense, the samplers are similar to Skolem functions (Hodges et al., 1997). Because samples are always finite, \forall is implemented as the product of all elements of the sample.

Optimization: DASL replaces Γ with the conjunction of all of its elements and thus the loss function is applied to a single truth value. We define $\mathcal{L}(t)$ as the cross-entropy between the distributions $(t, 1 - t)$ and $(1, 0)$, which yields $\mathcal{L}(t) = -\ln(t) = \ln(1 + e^{-l})$, where l is the logit of t . Not only is this loss function known to perform well in general on binary data, but together with our interpretations of the logical connectives it satisfies the condition above for sampling to converge to the total loss (even under infinite domains).

Extending the logical language: We describe the implementation of equality above; less than and greater than are implemented similarly. We do not require functions to be implemented as learned neural networks; they can be coded deterministically in PyTorch if desired. Several arithmetic operations are incorporated into the language in this way. We further extend the DASL language to allow for convenient specification of efficient networks. Firstly, connectives automatically operate on arbitrary sequences of logits. For example, $\wedge(u_0, \dots, u_n) = u_0 \wedge u_1 \wedge \dots \wedge u_n$. The connectives also generalize to work component-wise on tensors and support broadcasting as is familiar for tensor operators in PyTorch. For example, if \mathbf{X} is a matrix and \mathbf{y} is a vector, both of logits, then $\mathbf{X} \wedge \mathbf{y} = \mathbf{Z}$, where $Z_{ij} = X_{ij} \wedge y_i$.

The above property makes it possible to conveniently express knowledge as vectors of formulas, which can take advantage of tensor operations in PyTorch. We use this ability in subsection 4.2 to reduce the learning requirements on a classifier $\text{classify}(x)$ that maps input x to a value per class; these values would typically then pass to a softmax operation. We know that certain classes A could only be correct under conditions ϕ , which are detected outside the classifier, so we write $\text{classify}(\mathbf{x}) \wedge (\text{AClasses} \rightarrow \phi(\mathbf{x}))$ where AClasses is a constant vector over all classes with value 1 for the classes which are in A and 0 otherwise. The effect of this operation is to mask the output of the classes

in A whenever ϕ does not hold. Since the ground truth label will be compared to the output of the \wedge node, the classifier only receives feedback on these classes when ϕ holds, which is the only time it could ever receive positive feedback. The overall system is capable of learning to generate correct labels, while the classifier itself does not have the burden of learning to suppress the classes in A when ϕ does not hold. Boolean vectors are implemented by defining $\text{logit}(1)$ to be a fixed large constant.

Finally, we provide an explicit operator $\text{softselect}(\Gamma, i)$ (denoted as $\pi_i(\Gamma)$) which outputs the logit value for the i^{th} formula of Γ after application of the logit version of the softmax operator. This allows us to directly specify standard architectures for multi-class classification problems and to allow rules to operate on the classifier output within the model. Because i is an integer argument, we can quantify over it, effectively quantifying over a fixed finite list of predicates, providing syntactic convenience without violating the constraints of FOL.

4. Experiments

We evaluate DASL on two computer vision problems in data scarce conditions. We show that DASL augments deep learning with declarative knowledge to achieve better generalization. The first task is a toy problem based on digit classification on the MNIST dataset (LeCun et al., 1998), where knowledge is provided as an arithmetic relation satisfied by unlabeled triplets of digit images that are arranged artificially to satisfy that relation (subsection 4.1). We then focus on the problem of detecting visual relationships between object pairs and use commonsense knowledge about the plausible arguments of the relationship (subsection 4.2).

4.1. Toy Example on MNIST

Problem statement: We use this toy example to demonstrate DASL’s ability to train a NN from a few labeled samples and large number of unlabeled samples satisfying a rule. We denote a grayscale input image of a MNIST digit as \mathbf{X} and its label (if provided) as $y(\mathbf{X}) \in \mathbb{Z}_{10}$, where $\mathbb{Z}_{10} = \{0, 1, \dots, 9\}$. The task is to learn a NN $\text{digit}(\mathbf{X})$ to predict the digit in a test image.

For our toy example, we split the training data (50K images) into two disjoint sets: **Labeled**, containing a small number N_{tr} of labeled examples per digit class, and **Unlabeled**, used to generate the set **Triples** containing triplets of images $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$ satisfying the rule $y(\mathbf{X}_1) + y(\mathbf{X}_2) = y(\mathbf{X}_3) \bmod 10$. **Triples** contains only unlabeled images that together satisfy this relationship. We wish to learn the classifier by using **Labeled** and **Triples**, and thus the challenge is to compensate for the small size of **Labeled** by leveraging the prior knowledge about how the unlabeled

images in **Triples** are related. We formulate this problem within DASL by using its *softselect* operator π_i (see subsection 3.2) that, applied to the NN output $\text{digit}(\mathbf{X})$, returns the normalized score for the i^{th} class. This rule is written:

$$\begin{aligned} &(\forall(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) : \mathbf{Triples})(\forall y_1 : \mathbb{Z}_{10})(\forall y_2 : \mathbb{Z}_{10}) \\ &[(\pi_{y_1}(\text{digit}(\mathbf{X}_1)) \wedge \pi_{y_2}(\text{digit}(\mathbf{X}_2))) \\ &\quad \rightarrow \pi_{(y_1 + y_2) \bmod 10}(\text{digit}(\mathbf{X}_3))] \end{aligned}$$

We quantify over the triplets from **Triples** and all possible pairs of digits from \mathbb{Z}_{10} . We use this theory to augment the theory corresponding to the labeled training examples $(\forall(\mathbf{X}) : \mathbf{Labeled})(\pi_{y(\mathbf{X})}(\text{digit}(\mathbf{X})))$. The model is required to correctly infer the (unknown) labels of the triplet members and then use them for indirect supervision. We evaluate the model using the average accuracy on the test set (10K images). For $\text{digit}(\mathbf{X})$, we used a two-layer perceptron with 512 hidden units and a sigmoid non-linearity. We performed experiments in data scarce settings with $N_{tr} = 2, 5, 10$, and 20, and report mean performance with standard deviation across 5 random training subsets as shown in Figure 2. We use an equal number of examples per-class for constructing the triplets. We use a curriculum based training strategy (see supplementary) to prevent the model from collapsing to a degenerate solution, especially for lower values of N_{tr} . We train the model with the Adam optimizer (Kingma & Ba, 2014), learning rate of 5×10^{-5} , and batch size of 64. We report performance after 30K training iterations. A test image is classified into the maximum scoring class.

Results: Figure 2 shows a plot of digit classification accuracy versus the number of samples per class used for creating the triplets. We observe that the NN trained with both knowledge and data (*With-knowledge*) outperforms its counterpart trained with only labeled samples (*No-knowledge*). The improvement is particularly notable when training with smaller labeled training sets; e.g., for $N_{tr} = 2$, using all the knowledge raises performance from 53.3 ± 1.01 to 97.7 ± 0.00 . We also note that the performance of the *With-knowledge* model improves as the number of triplets increases and converges to similar values for different values of N_{tr} , indicating that the knowledge renders extra labels largely superfluous. The mean performance is 97.6 ± 0.00 , which is competitive with the performance of a model trained with all 50K labeled examples in MNIST (98.1 for $N_{tr} = \text{all}$). These results demonstrate the strength of DASL for exploiting knowledge to dramatically reduce data requirements. It also shows how DASL optimizes NNs that represent the domain language, using backpropagation to fit data and knowledge.

4.2. Visual Relationship Detection

Many problems in machine learning are endowed with inherent structure that can often be described explicitly. We

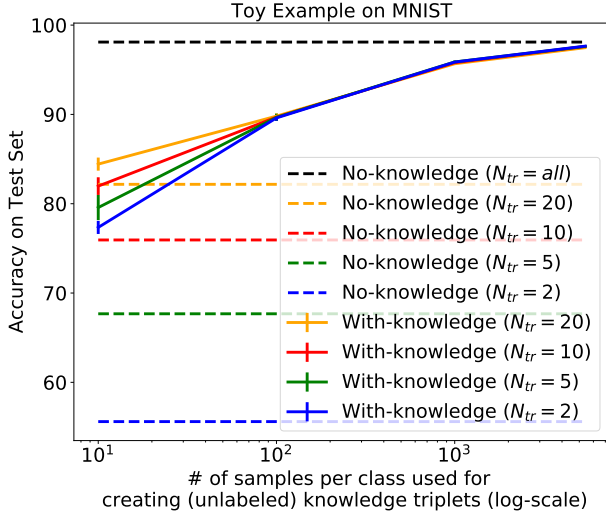


Figure 2. Figure showing the results for the MNIST toy example with a plot of accuracy of digit classification versus number of samples per class used for creating the unlabeled knowledge triplets. The labels *With-knowledge* and *No-knowledge* denote whether the training included the knowledge-augmented unlabeled triplets satisfying the given modular arithmetic (see subsection 4.1). N_{tr} refers to the number of labeled training examples per class (*all* refers to the entire training set). Best seen in color.

show this in the visual relationship detection task, where DASL incorporates commonsense knowledge into a DNN to improve learning with a small amount of training data.

Problem Statement: We use the Visual Relationship Detection (VRD) benchmark (Lu et al., 2016) to evaluate the **Predicate Detection Task:** Given an image and a set of ground-truth bounding boxes with object category labels, predict the predicates that describe the relationships between each pair of objects. The VRD dataset contains 5000 images spanning 37993 relationships covering 100 object classes and 70 predicate classes. We use splits provided by the authors that contain 4000 train and 1000 test images. The dataset also provides a *zero-shot* test subset of 1877 relationships built from the same classes as the training data but containing novel combinations of predicate classes with object class pairs.

Baseline model: We begin with a NN $\text{vrd}(\mathbf{I}, s, o)$ that outputs raw scores for predicate classes, where \mathbf{I} is the input RGB image and s and o are the indices of the subject and object classes respectively. We implement two variants of vrd similar to that proposed in (Liang et al., 2018). The first variant, referred to as *VGG*, extracts visual features from the last layer of a pre-trained VGG-16 network from the bounding box of the subject, the object, and their union. These features are projected into a 256 dimensional space by using a projection layer P (made of a fully-connected (FC) layer

and a ReLU non-linearity) and then fused by concatenation. The features are passed through another P layer followed by a FC layer to predict the class-wise scores. The second variant, referred to as *VGG-SS*, additionally incorporates the word-embedding features of the subject and the object (300 dimensional Glove features (Pennington et al., 2014)) along with the normalized relative spatial coordinates (see supplementary). These features are first projected using additional P layers and then concatenated with visual features, as done for VGG, prior to predicting the class-scores. We train the model with Adam optimizer (Kingma & Ba, 2014), learning rate of 5×10^{-5} , and batch size of 128.

DASL based Approach: We deviate from the simplified model of Figure 1, instead expressing knowledge as vectors of formulas as discussed in subsection 3.2. We begin by defining CanRide as a constant vector of truth values for all objects which is *True* at indices of objects which can ride and *False* elsewhere. $\text{CanRide}(s)$ selects its s^{th} element. Similarly, we define Ridable as a vector which is *True* at exactly the indices of objects which can be ridden. Finally we define a one-hot vector of truth values $\mathbf{h}_{Cls} \in \mathbb{R}^{70}$, which is *True* at the index of the predicate class “Cls” and *False* elsewhere. The theory which asserts that vrd should output the class labels assigned in the training data and that the “Riding” predicate should only apply when the subject can ride and the object can be ridden is written as:

$$\begin{aligned}
 (\forall(\mathbf{I}, s, o, y) : \mathbf{D}_{\text{vrd}})[\pi_y(\text{vrd}(\mathbf{I}, s, o) \\
 \wedge (\mathbf{h}_{\text{Riding}} \rightarrow \text{CanRide}(s) \\
 \wedge \text{Ridable}(o)))]
 \end{aligned}$$

where y is the given training label and \mathbf{D}_{vrd} is the training dataset. This rule reduces the learning burden of the classifier for “Riding” class by allowing feedback only when $\text{CanRide}(s)$ is *True*. We introduce a few more rules by adding them in conjunction with the above rules (see supplementary). These rules can be obtained from taxonomies (e.g. ConceptNet) or meta-data of prior datasets (e.g. VisualGenome (Krishna et al., 2017)).

Evaluation: We follow (Yu et al., 2017), reporting Recall@N ($R@N$), the recall of the top-N prediction scores in an image where we take into account all 70 predictions per object pair. This strategy is different from (Lu et al., 2016), which only considers the top prediction for each object pair penalizing cases where multiple predicates apply equally well but were omitted by the annotators.

Results: Table 1 shows the results on the VRD dataset when training with knowledge (+ *Knowledge*) and without knowledge (*baseline*) for the two variants and for both the standard and zero-shot settings. We observe consistent improvements across all cases with augmentation of knowledge. The improvements are higher for the 1% data (+7.7% for $R@100$

Method	R@50		R@100	
	Standard	Zero-Shot	Standard	Zero-Shot
1% Data				
VGG (baseline)	60.8 \pm 6.7	40.7 \pm 5.8	75.4 \pm 7.8	59.4 \pm 8.1
+ Knowledge	68.5 \pm 1.8**	49.5 \pm 1.5**	83.1 \pm 1.6**	70.1 \pm 2.4**
VGG-SS (baseline)	67.9 \pm 8.5	47.6 \pm 8.5	80.3 \pm 7.6	65.6 \pm 9.2
+ Knowledge	74.0 \pm 0.7*	54.4 \pm 1.4*	85.9 \pm 0.5*	73.4 \pm 1.2*
5% Data				
VGG (baseline)	70.3 \pm 0.5	48.4 \pm 1.0	83.5 \pm 0.4	68.3 \pm 0.9
+ Knowledge	73.8 \pm 0.5**	53.4 \pm 0.9**	86.4 \pm 0.4**	73.7 \pm 1.1**
VGG-SS (baseline)	79.6 \pm 0.4	58.1 \pm 1.2	89.6 \pm 0.3	77.1 \pm 1.1
+ Knowledge	79.9 \pm 0.4	59.6 \pm 0.9**	89.7 \pm 0.3	78.5 \pm 0.8**

Table 1. Performance on the predicate detection task from the Visual Relationship Dataset (Lu et al., 2016) with and without commonsense knowledge. We conduct the experiments in data scarce condition using 1% and 5% training data and report Recall@N averaged (with standard deviation) across 10 random subsets. “VGG” refers to a network using VGG-16 based visual features (Liang et al., 2018) and “VGG-SS” combines semantic and spatial features with the visual features. We report the statistical significance between “baseline” and corresponding knowledge augmented model (“+ Knowledge”) (p-value < 0.01 as ** and p-value < 0.05 as *).

for Standard) than the 5% data (+2.9% for R@100 for Standard) showing that knowledge has more benefits in lower data regimes. We made similar observation for the MNIST toy example. The improvements are generally higher for the zero-shot setting (+10.7% for R@100 in the 1% case) since this setting is inherently data starved and prior semantic knowledge helps to regularize the model in such conditions. We also note that the improvements are comparatively smaller for the VGG-SS network since semantic and spatial information are being explicitly injected as features into the model. Although there are some overlaps between the semantic features and provided declarative knowledge, they are fundamentally different, and could complement each other as observed above (59.4% of VGG versus 78.5% of VGG-SS + Knowledge). Our results show that DASL obtains better generalization in data scarce settings by augmenting the NN with commonsense rules.

5. Conclusion

In this paper, we introduced **Deep Adaptive Semantic Logic (DASL)** to unify machine reasoning and machine learning. DASL is fully general, encompassing all of first order logic and arbitrary deep learning architectures. DASL improves deep learning by supplementing training data with declarative knowledge expressed in first order logic. The vocabulary of the domain is realized as a collection of neural networks. DASL composes these networks into a single DNN and applies backpropagation to satisfy both data and knowledge. We provided a formal grounding which demonstrates the correctness and full generality of DASL for the representation of declarative knowledge in first order logic, including correctness of mini-batch sampling for arbitrary domains. This gives us to freedom to apply DASL in new

domains without requiring new correctness analysis.

We demonstrated a 1000-fold reduction in data requirements on the MNIST digit classification task by using declarative knowledge in the form of arithmetic relation satisfied by unlabeled image triplets. The knowledge *restricted* the behavior of the model, preventing erroneous generalization from the small number of labeled data points. We then demonstrated the application of commonsense knowledge to visual relationship detection, improving recall from 59.4 to 70.1. Here, knowledge was used to *free* the model from the burden of learning cases covered by the knowledge, allowing the model to do a better job of learning the remaining cases.

First order logic provides a uniform framework in which we plan to support transfer learning and zero-shot learning by training DASL models on theories where data is abundant and then creating new theories on the same vocabulary that address problems where data is sparse. We also plan to demonstrate the converse capability, training distinct models of a single theory, allowing us to sample models as a technique for capturing true probabilities, similar to Markov Logic Networks (Richardson & Domingos, 2006). Finally, we are exploring ways to allow DASL to learn rules from data while retaining explainability and integrating smoothly with user defined logic.

6. Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001118C0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the

views of DARPA. The authors would like to acknowledge Karen Myers, Bill Mark, Rodrigo de Salva Braz, and Yi Yao for helpful discussions.

References

- Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *CVPR*, pp. 39–48. IEEE Computer Society, 2016. ISBN 978-1-4673-8851-1.
- Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- Belanger, D., Yang, B., and McCallum, A. End-to-end learning for structured prediction energy networks. In *ICML*, pp. 429–439. JMLR. org, 2017.
- Chang, C. C. and Keisler, H. J. *Model theory*. Elsevier, 1973.
- Chang, M.-W., Ratnoff, L., and Roth, D. Structured learning with constrained conditional models. *Machine learning*, 88(3): 399–431, 2012.
- Cohen, W. W., Yang, F., and Mazaitis, K. R. Tensorlog: Deep learning meets probabilistic dbs. *arXiv preprint arXiv:1707.05390*, 2017.
- de Jong, M. and Sha, F. Neural theorem provers do not learn rules without exploration. *arXiv preprint arXiv:1906.06805*, 2019.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of American Society for Information Science*, 41:391–407, 1990.
- Detyniecki, M. Fundamentals on aggregation operators. Technical report, University of California, Berkeley, 2001.
- Garcez, A. S. d., Broda, K. B., and Gabbay, D. M. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2012.
- Garcez, A. S. d., Gori, M., Lamb, L. C., Serafini, L., Spranger, M., and Tran, S. N. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Human Level Artificial Intelligence*, 1:1–10, 2019.
- Genesereth, M. R. and Nilsson, N. J. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1987.
- Hodges, W. et al. *A shorter model theory*. Cambridge university press, 1997.
- Hu, Z., Ma, X., Liu, Z., Hovy, E., and Xing, E. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.
- Jaynes, E. T. *Probability theory: The logic of science*. Cambridge University Press, Cambridge, 2003.
- Kim, Y., Denton, C., Hoang, L., and Rush, A. M. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.
- Kimmig, A., Mihalkova, L., and Getoor, L. Lifted graphical models: A survey. *Machine Learning*, 99:1–45, 2004.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.
- Li, T. and Srikumar, V. Augmenting neural networks with first-order logic. *arXiv preprint arXiv:1906.06298*, 2019.
- Liang, K., Guo, Y., Chang, H., and Chen, X. Visual relationship detection with deep structural ranking. In *AAAI*, 2018.
- Lu, C., Krishna, R., Bernstein, M., and Fei-Fei, L. Visual relationship detection with language priors. In *ECCV*, pp. 852–869. Springer, 2016.
- Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*. OpenReview.net, 2019.
- Marcus, G. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *NIPS*, pp. 3111–3119. Curran Associates, Inc., 2013.
- Oquab, M., Bottou, L., Laptev, I., Sivic, J., et al. Weakly supervised object recognition with convolutional neural networks. In *NIPS*, pp. 1545–1596, 2014.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*, pp. 8024–8035, 2019.
- Pearl, J. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann, San Francisco, Calif., 2009. ISBN 9781558604797 1558604790.
- Peng, H., Thomson, S., and Smith, N. A. Backpropagating through structured argmax using a spigot. *arXiv preprint arXiv:1805.04658*, 2018.
- Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP*, pp. 1532–1543, 2014.
- Richardson, M. and Domingos, P. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-5833-1.
- Rocktäschel, T. and Riedel, S. End-to-end differentiable proving. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *NIPS*, pp. 3788–3800. Curran Associates, Inc., 2017.

- Rocktäschel, T., Singh, S., and Riedel, S. Injecting logical background knowledge into embeddings for relation extraction. In *ICML*, pp. 1119–1129, 2015.
- Serafini, L. and Garcez, A. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *CoRR*, abs/1606.04422, 2016.
- Siekmann, J. and Wrightson, G. *Automation of Reasoning*. Springer, 1983.
- Stewart, R. and Ermon, S. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, 2017.
- Towell, G. G. and Shavlik, J. W. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.
- Weiss, K., Khoshgoftaar, T. M., and Wang, D. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.
- Weiss, W. and D’Mello, C. Fundamentals of model theory, 1997.
- Xie, Y., Xu, Z., Meel, K., Kankanhalli, M. S., and Soh, H. Semantically-regularized logic graph embeddings. *CoRR*, abs/1909.01161, 2019.
- Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. Distance metric learning with application to clustering with side-information. In *NIPS*, pp. 521–528, 2003.
- Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pp. 2319–2328, 2017.
- Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *NIPS*, pp. 1039–1050, 2018a.
- Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NIPS*, pp. 1031–1042, 2018b.
- Yu, R., Li, A., Morariu, V. I., and Davis, L. S. Visual relationship detection with internal and external linguistic knowledge distillation. In *CVPR*, pp. 1974–1982, 2017.
- Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. Conditional random fields as recurrent neural networks. In *ICCV*, pp. 1529–1537, 2015.

7. Supplementary Material

7.1. Curriculum learning for MNIST Toy Example

In **section 4.1** we trained a NN for digit classification on the MNIST dataset in a data scarce setting. We used a few labeled samples and a large number of unlabeled triplets satisfying some rules (modular arithmetic in our experiments). We used a curriculum based learning strategy to prevent the model from collapsing to a degenerate solution, especially for cases with extremely small number of labeled samples (*e.g.* 2 samples per class). In such cases the model tends to get trapped in a local minimum where the axiom corresponding to the unlabeled triplets can be satisfied by a solution with all digits being classified as 0 since $0 + 0 = 0$. Within the curriculum, we begin the training with all the labeled examples and a small working set of the unlabeled triplets. We progressively expand the working set during training as the model becomes more confident on the unlabeled examples. The confidence score p_c^t is computed using a low-pass filter:

$$p_c^t = (1 - \alpha) * p_c^{t-1} + \alpha * p_{max}$$

where $*$ is scalar multiplication, t is the iteration index, $p_c^0 = 0$, $\alpha = 0.1$, and p_{max} is the average probability of the highest scoring class on the first digit of the triplet. When $p_c^t > 0.9$, we increase the working set of unlabeled triplets by a factor of 2 until it reaches the maximum number of unlabeled triplets. When $p_c^t > 0.9$, we reset p_c^t to let the model fit well to the new working set before reaching the condition again. This curriculum ensures that the model is able to find a decent initialization using the labeled examples and then progressively improve using the unlabeled samples. The initial set of unlabeled triplets contained 10 samples per class and the maximum number of triplets is bounded by the class with minimum number of samples. During the final curriculum step we remove all labeled data, allowing the model to train solely on the rules. This allows the model to trade off errors on the labeled data for better overall performance.

7.2. Normalized Relative Spatial Features for Visual Relationship Detection

We provide the implementation details for the spatial features used in the visual relationship detection experiments in **section 4.2**. These features capture the relative spatial configuration of the subject and the object bounding boxes and were used to augment visual and semantic features for predicting the visual relationship (VGG-SS). We denote the coordinates of the object and subject bounding boxes as (x_s, y_s, w_s, h_s) and (x_o, y_o, w_o, h_o) respectively, where (x, y) are the coordinates of the (box) center with width w and height h . The relative normalized features is an eight dimensional feature and computed as $\left[\frac{x_s - x_o}{w_o}, \frac{y_s - y_o}{h_o}, \frac{x_o - x_s}{w_s}, \frac{y_o - y_s}{h_s}, \log\left(\frac{w_s}{w_o}\right), \log\left(\frac{h_s}{h_o}\right), \log\left(\frac{w_o}{w_s}\right), \log\left(\frac{h_o}{h_s}\right) \right]$. These features were also used in the baseline model (Liang et al., 2018).

7.3. Commonsense rules for Visual Relationship Detection

In addition to the rule described in **section 4.2**, we used additional rules for incorporating commonsense knowledge in predicting visual relationships using DASL. These rules follow the same format as the rule for the ‘‘Riding’’ predicate described earlier and are outlined below:

1. ‘‘Wear’’ predicate should only apply when the subject is a *living entity* and the object is *wearable*.
2. ‘‘Sleep-On’’ predicate should only apply when the subject is a *living entity* and the object is *sleepable*.
3. ‘‘Eat’’ predicate should only apply when the object is *eatable*.
4. Predicates- ‘‘Above’’, ‘‘Over’’, ‘‘Ride’’, ‘‘On-The-Top-Of’’, ‘‘Drive-on’’, ‘‘Park-On’’, ‘‘Stand-On’’, ‘‘Sit-On’’, ‘‘Rest-On’’ should apply only when the subject is spatially *above* the object. We defined *above* as a function that is *True* when $y_s \geq y_o$.
5. Predicates- ‘‘Under’’, ‘‘Beneath’’, ‘‘Below’’, ‘‘Sit-Under’’ should apply only when the subject is spatially *below* the object. We defined *below* as a function that is *True* when $y_s \leq y_o$.
6. Predicates- ‘‘On-The-Right-Of’’ should apply only when the subject is spatially *right of* the object. We defined *right of* as a function that is *True* when $x_s \geq x_o$.
7. Predicates- ‘‘On-The-Left-Of’’ should apply only when the subject is spatially *left of* the object. We defined *left of* as a function that is *True* when $x_s \leq x_o$.

These rules cover facts related to both semantic and spatial commonsense knowledge. We incorporated these rules by adding them in conjunction with original theory presented in **section 4.2**.

$$\begin{aligned}
 &(\forall(\mathbf{I}, s, o, y) : \mathbf{D}_{\text{vrd}})[\pi_y(\text{vrd}(\mathbf{I}, s, o) \\
 &\quad \wedge (\mathbf{h}_{\text{Riding}} \rightarrow \text{CanRide}(s) \\
 &\quad \quad \wedge \text{Ridable}(o)) \\
 &\quad \wedge (\mathbf{h}_{\text{Wear}} \rightarrow \text{Living}(s) \\
 &\quad \quad \wedge \text{Wearable}(o)) \dots]
 \end{aligned}$$

where $\mathbf{h}_{\text{Cls}} \in \mathbb{R}^{70}$ is a one-hot vector of truth values, which is *True* at the index of the predicate class “Cls” and *False* elsewhere. *Living* is a constant vector of truth values for all objects which is *True* at indices of objects which are living entities and *False* elsewhere. Similarly, *Wearable* is a constant vector, which is *True* at exactly the indices of objects which are wearable. We refer readers to **section 4.2** for detailed explanation about the application of these rules.