GLUECons: A Generic Benchmark for Learning under Constraints

Hossein Rajaby Faghihi[1], Aliakbar Nafar[1], Chen Zheng[1], Roshanak Mirzaee[1], Yue Zhang[1], Andrzej Uszok[2], Alexander Wan[3*], Tanawan Premsri[1], Dan Roth[4], and Parisa Kordjamshidi[1]

[1] Michigan State University
[2] Florida Institute for Human and Machine Cognition
[3] University of California Berkeley
[4] University of Pennsylvania
{rajabyfa, nafarali, zhengc12, mirzaeem, zhan1624, premsrit, kordjams}@msu.com,
auszok@ihmc.org, alexwan@berkeley.edu, danroth@seas.upenn.edu

Abstract

Recent research has shown that integrating domain knowledge into deep learning architectures is effective; It helps reduce the amount of required data, improves the accuracy of the models' decisions, and improves the interpretability of models. However, the research community lacks a convened benchmark for systematically evaluating knowledge integration methods. In this work, we create a benchmark that is a collection of nine tasks in the domains of natural language processing and computer vision. In all cases, we model external knowledge as *constraints*, specify the sources of the constraints for each task, and implement various models that use these constraints. We report the results of these models using a new set of extended evaluation criteria in addition to the task performances for a more in-depth analysis. This effort provides a framework for a more comprehensive and systematic comparison of constraint integration techniques and for identifying related research challenges. It will facilitate further research for alleviating some problems of state-of-the-art neural models.

1 Introduction

Deep Learning Shortcomings Recent advancements in machine learning are proven very effective in solving real-world problems in various areas, such as vision and language. However, there are still remaining challenges. First, machine learning models mostly fail to perform well on complex tasks where reasoning is crucial (Schubotz et al. 2018) while human performance does not drop as much when more steps of reasoning are required. Second, deep neural networks (DNNs) are known to be data-hungry, making them struggle on tasks where the annotated data is scarce (Li, Wang, and Yu 2020; Zoph et al. 2016). Third, models often provide results that are inconsistent (Li et al. 2019; Gardner et al. 2020) even when they perform well on the task. Prior research has shown that even large pre-trained language models performing well on a specific task may suffer from inconsistent decisions and indicate unreliability when attacked under adversarial examples and specialized test sets that evaluate their logical consistency (Gardner et al. 2020; Mirzaee et al. 2021a). This is especially a major concern when interpretability is required (Mathews 2019), or there are security concerns over applications relying on the decisions of DNNs (Brundage et al. 2020).

Knowledge Integration Solution To address these challenges, one direction that the prior research has investigated is neuro-symbolic approaches as a way to exploit both symbolic reasoning and sub-symbolic learning. Here, we focus on a subset of these approaches for the integration of external knowledge in deep learning. Knowledge can be represented through various formalisms such as logic rules (Hu et al. 2016; Nandwani, Pathak, and Singla 2019), Knowledge graphs (Zheng and Kordjamshidi 2022), Context-free grammars (Deutsch, Upadhyay, and Roth 2019), Algebraic equations (Stewart and Ermon 2017), or probabilistic relations (Constantinou, Fenton, and Neil 2016). A more detailed investigation of available sources of knowledge and techniques to integrate them with DNNs is surveyed in (von Rueden et al. 2019; Dash et al. 2022). Although integrating knowledge into DNNs is done in many different forms, we focus on explicit knowledge about the latent and/or output variables. More specifically, we consider the type of knowledge that can be represented as declarative constraints imposed (in a soft or hard way) on the models' predictions, during training or at inference time. The term knowledge integration is used in the scope of this assumption in the remainder of this paper.

Hurdle of Knowledge Integration Unfortunately, most prior research on knowledge integration has only focused on evaluating their proposed method compared to baseline DNN architectures that ignore the knowledge. Consequently, despite each method providing evidence of its effectiveness (Hu et al. 2016; Nandwani, Pathak, and Singla 2019), there is no comprehensive analysis that can provide a better understanding of the use cases, advantages, and disadvantages of methods, especially when compared with each other. The lack of such analysis has made it hard to apply these approaches to a more diverse set of tasks by a broader community and provide a clear comparison with existing methods. We mainly attribute this to three factors: 1) the lack of a standard benchmark with systematic baselines, 2) the difficulty of finding appropriate tasks where constraints

[*] Summer Intern at Michigan State University

are applicable, and 3) the lack of supporting libraries for implementing various integration techniques.

Due to these three factors, many research questions are left open for the community, such as (1) The difference in the performance of models when knowledge is integrated during inference vs. training or both, (2) The comparison of the influence of integration methods when combined with simpler vs. more complex baselines, (3) The effectiveness of training-time integration models on reducing the constraint violation, (4) The impact of data size on the effectiveness of the integration methods.

Common Ground for Comparison The contribution of this paper is providing a common ground for comparing techniques for knowledge integration by collecting a new benchmark to facilitate research in this area. Our new benchmark, called GLUECons, contains a collection of tasks suitable for constraint integration, covering a spectrum of constraint complexity, from basic linear constraints such as mutual exclusivity to more complex constraints expressed in first-order logic with quantifiers. We organize the tasks in a repository with a unified structure where each task contains a set of input examples, their output annotations, and a set of constraints (written in first-order logic). We limit the scope of knowledge in GLUECons to logical constraints[1].

Selected Tasks GLUECons contains tasks ranging over five different types of problems categorized based on the type of available knowledge. This includes **1)** Classification with label dependencies: Mutual exclusivity in multiclass classification using MNIST (LeCun et al. 1998) and Hierarchical image classification using CIFAR 100 (Krizhevsky and Hinton 2009), **2)** Self-Consistency in decisions: What-If Question Answering (Tandon et al. 2019), Natural Language Inference (Bowman et al. 2015), BeliefBank (Kassner et al. 2021), **3)** Consistency with external knowledge: Entity and Relation Extraction using CONLL2003 (Sang and De Meulder 2003), **4)** Structural Consistency: BIO Tagging, **5)** Constraints in (un/semi)supervised setting: MNIST Arithmetic and Sudoku. These tasks either use existing datasets or are extensions of existing tasks, reformulated so that the usage of knowledge is applicable to them. We equip these tasks with constraint specifications and baseline results.

Evaluation For a fair evaluation and to isolate the effect of the integration technique, we provide a repository of models and code for each task in both PyTorch (Paszke and Gross 2019) and DomiKnows (Faghihi et al. 2021) frameworks. DomiKnows is an easy-to-use tool for expressing constraints in first-order logic with automatic conversion to linear constraints. It provides a modular interface for modeling and applying constraints, making it easier to consistently test different integration methods while the rest of the configurations remain unchanged.

For a more comprehensive evaluation, we introduce a set of new criteria in addition to the original task performances to measure 1) the effectiveness of the techniques in increasing the consistency with knowledge 2) the execution runtime, 3) the effectiveness of methods in the low-data regime, 4) the ability to reduce the need for complex models, and 5) the ability to express various forms of knowledge.

Baselines We analyze and evaluate a set of knowledge integration methods to serve as baselines for GLUECons. Our baselines cover a set of fundamentally different integration methods, where the integration is addressed either during inference or training of DNNs. GLUECons can be used as blueprints to highlight the importance of integrating constraints with DNNs for different types of tasks and provides inspiration for building such constraints when working on new tasks.

In summary, the contributions of this paper are 1) We propose the first extensive benchmark exclusively designed for evaluating constraint integration methods in deep learning (GLUECons), 2) We define new evaluation criteria in addition to the task performance for a comprehensive analysis of the techniques, and 3) We establish standard baselines for the tasks in this benchmark based on multiple constraint integration methods.

2 Constraint Integration in Prior Research

Knowledge integration, often, is considered a subset of Neuro-symbolic (De Raedt et al. 2019; Amizadeh et al. 2020; Huang et al. 2021) approaches that build on the intersection of neural learning and symbolic reasoning. von Rueden et al. surveyed prior research on knowledge integration in three directions: knowledge source, knowledge representation, and the stage of knowledge integration. Dash et al. has also studied existing methods where the integration can be done through either transforming the input data, the loss function, or the model architecture itself. Knowledge integration has also been investigated in probabilistic learning frameworks (De Raedt, Kimmig, and Toivonen 2007; Richardson and Domingos 2006; Bach et al. 2017) and their modern extensions which use neural learning (Manhaeve et al. 2018; Huang et al. 2021; Winters et al. 2021). Recent research has explored knowledge integration via bypassing the formal representations and expressing knowledge in the form of natural language as a part of the textual input (Saeed et al. 2021; Clark, Tafjord, and Richardson 2020). As of formal representations, knowledge integration has been addressed at both inference (Lee et al. 2019; Scholak, Schucher, and Bahdanau 2021; Dahlmeier and Ng 2012) and training time (Hu et al. 2016; Nandwani, Pathak, and Singla 2019; Xu et al. 2018).

Inference-Time Integration: The inference-based integration techniques optimize over the output decisions of a DNN, where the solution is restricted by a set of constraints expressing the knowledge (Roth and Yih 2005; Chang, Ratinov, and Roth 2012).

These methods aim at finding a valid set of decisions given the constraints, while their objective is formed using the output scores/probabilities generated by the learning models. As a result of this fixed objective and the fact that approximation approaches are generally used to find the best solution, we expect that the type of optimization technique will not significantly affect the performance of inference-time integration methods –our results discussed in the later

[1]Throughout this paper, we use the terms constraint integration, knowledge integration, or integration methods interchangeably to refer to the process of integration of knowledge into the DNNs.

sections provide multiple pieces of evidence confirming this hypothesis. Prior research has investigated such integration by using variants of beam search (Hargreaves, Vlachos, and Emerson 2021; Borgeaud and Emerson 2020; Dahlmeier and Ng 2012), path search algorithm (Lu et al. 2021), linear programming (Roth and Yih 2005; Roth and Srikumar 2017; Chang, Ratinov, and Roth 2012), finite-state/push-down Automata (Deutsch, Upadhyay, and Roth 2019), or applying gradient-based optimization at inference (Lee et al. 2019, 2017). We use Integer Linear Programming (ILP) (Roth and Yih 2005; Roth and Srikumar 2017) approach to evaluate the integration of the constraints at inference time. We use off-the-shelf ILP tools that perform an efficient search and offer a natural way to integrate constraints. However, constraints should be converted to a linear form to be able to exploit these tools (Faghihi et al. 2021; Kordjamshidi, Roth, and Wu 2015; Kordjamshidi et al. 2016).

Training-Time Integration: Several recent techniques have been proposed for knowledge integration at training time (Nandwani, Pathak, and Singla 2019; Hu et al. 2016; Xu et al. 2018). Using constraints during training usually requires finding a differentiable function expressing constraint violation. This will help to train the model to minimize the violations as a part of the loss function. Integrating knowledge in the training loop of DNNs is a challenging task. However, it can be more rewarding than the inference-based integration methods as it reduces the computational overhead by alleviating the need for using constraints during inference. Although such methods cannot guarantee that the output decisions would follow the given constraints without applying further operations at inference-time, they can substantially improve the consistency with the constraints (Li et al. 2019). Prior research has investigated this through various soft interpretations of logic rules (Nandwani, Pathak, and Singla 2019; Asai and Hajishirzi 2020), rule-regularized supervision (Hu et al. 2016; Guo et al. 2021), re-enforcement learning (Yang et al. 2021), and black-box semantic (Xu et al. 2018) or sampling (Ahmed et al. 2022) loss functions, which directly train the network parameters to output a solution that obeys the constraints.

To cover a variety of techniques based on the previous research, we select Primal-Dual (PD) (Nandwani, Pathak, and Singla 2019) and Sampling-Loss (SampL) (Ahmed et al. 2022) methods as baselines for our new benchmark. The PD approach relies on a soft logic interpretation of constraints, while the SampL is a black-box constraint integration. We discuss some of the existing methods in more detail in Section 'Baselines.'

2.1 Applications and Tasks

Constraint integration has been investigated for several applications in prior research including SQL query generation (Scholak, Schucher, and Bahdanau 2021), program synthesize (Austin et al. 2021; Ellis et al. 2021), semantic parsing (Clarke et al. 2010; Lee, Gottschlich, and Roth 2021), question answering (Asai and Hajishirzi 2020), entity and relation extraction (Guo et al. 2021), sentiment analysis (Hu et al. 2016), visual question answering (Huang et al. 2021), image captioning (Anderson et al. 2017), and even text generation (Lu et al. 2021).

3 Criteria of Evaluation

We extend the evaluation of the constraint integration methods beyond measuring task performance. The list of proposed evaluation criteria for such an extended comparison is as follows.

Individual metrics of each task: The first criterion to evaluate the methods is the conventional metric of each task, such as accuracy or precision/recall/F1 measures.

Constraint Violation: Even when the integration method cannot improve the model's performance, improving the consistency of its predictions will make the neural models more reliable. A consistency measure quantifies the success of the integration method in training a neural network to follow the given constraints. We measure consistency in terms of constraint violation. We compute the ratio of violated constraints over all predicted outputs. A smaller number indicates fewer constraint violations and, consequently, a higher consistency with the available knowledge.

Execution Run-Time: Another critical factor in comparing the constraint integration methods is the run-time overhead. This factor becomes even more critical when the integration happens during inference. This criterion helps in analyzing the adequacy of each technique for each application based on the available resources and the time sensitivity of the decision-making for that application. We measure this evaluation criteria by simply computing the execution time of each integration method both during training and inference. This metric can reflect the overhead of each integration method more accurately by taking into account the new parameters that should be optimized and the additional computations with respect to the complexity of the constraints.

Low-data vs full-data performance: For many problems, there is no large data available either due to the high cost or infeasibility of obtaining labeled data. Integrating constraints with deep neural learning has been most promising in such low-resource settings (Nandwani, Pathak, and Singla 2019; Guo et al. 2021). We measure the improvement resulting from the integration methods on both low and full data. This evaluation will help in choosing the most impactful integration method based on the amount of available data when trying to apply integration methods to a specific task.

Simple baseline vs Complex baseline: An expected impact of constraint integration in DNNs is to alleviate the need for a large set of parameters and achieve the same performance using a smaller/simpler model. Additionally, it is important to evaluate whether the integration method can only affect the smaller network or the very large SOTA models can be improved too. This will indicate whether large networks/pre-trained models can already capture the underlying knowledge from the data or explicit constraint integration is needed to inject such knowledge. In addition to the number of parameters, this metric also explores whether knowledge integration can reduce the need for pre-training. This is especially important for the natural language domain, where large pre-trained language models prevail.

Constraint Complexity: This criterion evaluates the limitations of each method for integrating different types of

knowledge. Some methods consider the constraints a black box with arbitrary complexity, while others may only model a specific form of constraint. This criterion specifies the form/complexity of the constraints that are supported by each technique. To evaluate this, we characterize a set of constraint complexity levels and evaluate whether each technique can model such constraints.

4 Selected Tasks

GLUECons aims to provide a basis for comparing constraint integration methods. We have selected/created a collection of tasks where constraints can potentially play an important role in solving them. We provide five different problem categories containing a total of nine tasks. More details of tasks' constraints are available in the Appendix. This collection includes a spectrum of very classic tasks for structured output prediction, such as multi-class classification to more involved structures and knowledge, such as entity relation extraction and Sudoku.

4.1 Classification with Label Dependency

Simple Image Classification. In this task, we utilize the classic MNIST (Deng 2012) dataset and classify images of handwritten digits in the range of 0 to 9. The constraint used here is the mutual exclusivity of the ten-digit classes. Each image can only have one valid digit label as expressed in the following constraint,

$$\text{IF } digit_i(x) \Rightarrow \neg \vee_{j=!i}^{j\in[0-9]} digit_j(x),$$

where $digit_i(x)$ is 1 if the model has predicted x to be an image representing the digit i. This task is used as a basic validation of the constraint integration methods, though it is not very challenging and can also be addressed by a "Softmax" function.

Hierarchical Image Classification. The hierarchical relationships between labels present a more complex label dependency in multi-label and multi-class tasks. We use the CIFAR-100 (Krizhevsky, Sutskever, and Hinton 2012), which includes 100 image classes, each belonging to 20 parent classes forming a hierarchical structure. This dataset with 60k images is an extension of the classic CIFAR-10 (Krizhevsky, Sutskever, and Hinton 2012). To create a smaller dataset, we select 10% of these 60k images. For this task, the output is a set of labels for each image, including one label for each level. The constraints are defined as,

$$\text{IF } L_1 \subset L_2 : L_1(x) \Rightarrow L_2(x),$$

where L_1 and L_2 are labels, $L_1(x)$ is $True$ only if the models assigns label L_1 to x, and $L_1 \subset L_2$ indicates that L_1 is a subclass of L_2.

4.2 Self Consistency in Decisions

DNNs are subject to inconsistency over multiple decisions while being adept at answering specific questions (Camburu et al. 2019). Here, we choose three tasks to evaluate whether constraints help ensure consistency between decisions.

Causal Reasoning. WIQA (Tandon et al. 2019) is a question-answering (QA) task that aims to find the line of causal reasoning by tracking the causal relationships between cause and effect entities in a document. The dataset contains 3993 questions. Following (Asai and Hajishirzi 2020), we impose symmetry and transitivity constraints on the sets of related questions. For example, the symmetry constraint is defined as follows: $symmetric(q, \neg q) \Rightarrow F(q, C) \wedge \neg F(\neg q, C)$ where q and $\neg q$ represent the question and its negated variation, C denotes the document, and $\neg F$ is the opposite of the answer F.

Natural Language Inference. Natural Language Inference (NLI) is the task of evaluating a hypothesis given a premise, both expressed in natural language text. Each example contains a premise (p), hypothesis (h), and a label/output (l) which indicates whether h is "entailed," "contradicted", or "neutral" by p.

Here, we evaluate whether NLI models benefit from consistency rules based on logical dependencies. We use the SNLI (Bowman et al. 2015) dataset, which includes 500k examples for training and 10k for evaluation. Furthermore, we include A_{1000}^{ESIM} (Minervini and Riedel 2018), which is an augmented set over the original dataset containing more related hypotheses and premise pairs to enforce the constraints. Four consistency constraints (symmetric/inverse, transitive) are defined based on the (Hypothesis, Premise) pairs. An example constraint is as follows:

$$\text{neutral}\,(h, p) \Rightarrow \neg\,\text{contradictory}\,(p, h)\,,$$

where $\text{neutral}\,(h, p)$ is *True* if h is undetermined given p. The complete constraints are described in (Minervini and Riedel 2018).

Belief Network Consistency. The main goal of this task is to impose global belief constraints to persuade models to have consistent beliefs. As humans, when we reason, we often rely upon our previous beliefs about the world, whether true or false. We can always change our minds about previous information based on new information, but new beliefs should not contradict previous ones. Here, entities and their properties are used as facts. We form a global belief network that must be consistent with those derived from a given knowledge base. We use Belief Bank (Kassner et al. 2021) dataset to evaluate the consistency perseverance of various techniques. The dataset consists of 91 entities and $23k$ ($2k$ train, $1k$ dev, $20k$ test) related facts extracted from ConceptNet (Speer, Chin, and Havasi 2016). There are $4k$ positive and negative implications between the facts in the form of a constraint graph. For example, the fact "Is a bird" would imply "can fly," and the fact "can fly" refute the fact "Is a dog". Formally, the constraints are defined as follows:

$$\forall F_1, F_2 \in \text{Facts};$$
$$\text{IF } F_1, F_2 \in \text{Pos Imp} \Rightarrow \neg F_1(x) \vee F_2(x)$$
$$\text{IF } F_1, F_2 \in \text{Neg Imp} \Rightarrow \neg F_1(x) \vee \neg F_2(x),$$

"Pos Imp" means a positive implication.

4.3 Consistency with External Knowledge

This set of tasks evaluates the constraint integration methods in applying external knowledge to the DNNs' outputs.

Entity Mention and Relation Extraction (EMR). This task is to extract entities and their relationships from a

document. Here, we focus on the CoNLL2003 (Sang and De Meulder 2003) dataset, which contains about 1400 articles. There are two types of constraints involved in this task: 1) mutual exclusivity between entity/relationship labels and 2) a restriction on the types of entities that may engage in certain relationships. An example constraint between entities and relationship types is as follows:

$$\text{IF Work\_for}(x1, x2) \Rightarrow \text{Person}(x1) \wedge \text{Org}(x2),$$

where $\text{Predicate}(x)$ is $True$ if the network predicted input x to be of type Predicate.

4.4 Structural Consistency

In this set of tasks, we evaluate the impact of constraint integration methods in incorporating structural knowledge over the task's outputs.

BIO Tagging. The BIO tagging task aims to identify spans in sentences by tagging each token with one of the "Begin," "Inside," and "Outside" labels. Each tagging output belongs to a discrete set of BIO tags $T \in$ ['O', 'I-*', 'B-*'], where '*' can be any type of entity. Words tagged with O are outside of named entities, while the 'B-*' and 'I-*' tags are used as an entity's beginning and inside parts. We use the CoNLL-2003 (Sang and De Meulder 2003) benchmark to evaluate this task. This dataset includes 1393 articles and 22137 sentences. The constraints of the BIO tagging task are valid BIO sequential transitions; for example, the "before" constraint is defined as follows:

$$\text{If } I(x_i + 1) \rightarrow B(x_i),$$

where 'B-*' tag should appear before 'I-*' tag. x_i and x_{i+1} are any two consecutive tokens.

4.5 (Un/Semi) Supervised Learning

We select a set of tasks for which the constraints can alleviate the need for direct supervision and provide a distant signal for training DNNs.

Arithmetic Operation as Supervision for Digit Classification. We use the MNIST Arithmetic (Bloice, Roth, and Holzinger 2020) dataset. The goal is to train the digit classifiers by receiving supervision, merely, from the sum of digit pairs. For example, for image pairs of 5 and 3 in the training data, we only know their labels' sum is 8. This dataset is relatively large, containing 10k image pairs for training and 5k for testing. This task's constraint forces the networks to produce predictions for pairs of images where the summation matches the ground-truth sum. The following logical expression is an example constraint for this task:

$$S(\{img_1, img_2\}) \Rightarrow \bigvee_{M=max(0,S-9)}^{M=min(S,9)} M(img_1) \wedge \{S - M\}(img_2),$$

where $S(\{img_1, img_2\})$ indicates that the given summation label is S and $M(img_i)$ indicates that the ith image has the label M.

Sudoku. This task evaluates whether constraint integration methods can help DNNs to solve a combinatorial search problem such as Sudoku. Here, integration methods are used as an inference algorithm with the objective of solving one Sudoku, while the only source of supervision is the Sudoku constraints. As learning cannot be generalized in this setting, it should be repeated for each input. The input is one Sudoku table partially filled with numbers, and the task is to fill in a number in each cell such that: "There should not be two cells in each row/block/column, with the same value" or formally defined as:

$$\begin{aligned}&\text{IF } \text{digit}_\text{i}(x) \wedge \\ &(\text{same\_row}(x, y) \vee \text{same\_col}(x, y) \vee \text{same\_block}(x, y)) \\ &\Rightarrow \neg\, \text{digit}_\text{i}(y),\end{aligned}$$

where x and y are variables regarding the cells of the table, $i \in [0, n]$ for a $n * n$ Sudoku, $digit_i(x)$ is $True$ only if the value of x is predicted to be i. For this task, we use an incomplete $9 * 9$ Sudoku for the full-data setting and a $6 * 6$ Sudoku representing the low-data setting.

5 Baselines

For constraints during training, we use the two following approaches.

Primal-Dual (PD). This approach (Nandwani, Pathak, and Singla 2019) converts the constrained optimization problem into a min-max optimization with Lagrangian multipliers for each constraint and augments the original loss of the neural models. This new loss value quantifies the amount of violation according to each constraint by means of a soft logic surrogate. During training, they optimize the decisions by minimizing the original violation, given the labels, and maximizing the Lagrangian multipliers to enforce the constraints. It is worth noting that all related work in which constraint violation is incorporated as a regularization term in the loss objective follows very similar variations of a similar optimization formulation.

Sampling-Loss (SampL). This approach (Ahmed et al. 2022) is an extension of the semantic loss (Xu et al. 2018) where instead of searching over all the possibilities in the output space to find satisfying cases, it randomly generates a set of assignments for each variable using the probability distribution of the neural network's output. The loss function is formed as:

$$L^S(\alpha, p) = \frac{\sum_{x^i \in X \wedge x^i \models \alpha} p\left(x^i \mid p\right)}{\sum_{x^i \in X} p\left(x^i \mid p\right)},$$

where X is the set of all possible assignments to all output variables, and x^i is one assignment. Here, α is one of the constraints.

To utilize the constraints during prediction, we use the following approaches.

Integer Linear Programming. (ILP) (Roth and Yih 2005) is used to formulate an optimization objective in which we want to find the most probable solution for $\log F(\theta)^\top y$, subject to the constraints. Here, y is the unknown variable in the optimization objective, and $F(\theta)$ is the network's output probabilities for each variable in y. The constraints on y are formulated as $\mathcal{C}\left(y\right) \leq 0$.

Task	Strong Baseline	Simple Baseline
Img Cls.	CNN + MLP	MLP
Hier. Img Cls.	Resnet18 + MLP	-
NLI	RoBERTa + MLP	-
Causal Rea.	RoBERTa + MLP	BERT + MLP
BIO Tagging	BERT + MLP	LSTM + MLP
NER	W2V + BERT + MLP	W2V + LSTM + MLP
Ari. Operation	CNN + MLP	-
BeliefNet.	RoBERTa + MLP	W2V + MLP
Sudoku∗	$(n * n * n)$ Vector	-

Table 1: Baselines for each task. The basic models we used are RoBERTa (Liu et al. 2019), BERT (Devlin et al. 2018), W2V (Mikolov et al. 2013), CNN (LeCun et al. 1998), and MLP. The simple baseline means fewer parameters. [KEYS: Cls.=classification, Hier.=Hierarchical, NLI= Natural Language Inference, Rea.=reasoning, Ari.=Arithmetic, BeliefNet=Belief Network, W2V=Word to Vec.]. ∗ For the Sudoku, the model is not a generalizable DNN and relies on the integration methods as an inference algorithm to solve one specific table.

Search and Dynamic Programming. for some of the proposed benchmarks, when applicable, we use the A^* search or Viterbi algorithm to choose the best output at prediction time given the generated probability distribution of the final trained network (Lu et al. 2021).

6 Experiments and Discussion

This section highlights our experimental findings using proposed baselines, tasks, and evaluation criteria. Details on experimental designs, training hyper-parameters, codes, models, and results can be found on our website[2]. The basic architectures for each task are shown in Table 1.

The results of the experiments are summarized in Table 3. The columns represent evaluation criteria, and the rows represent tasks and their baselines. Each task's model 'row' records the strong/simple baseline's performance without constraint integration, and below that, the improvements or drops in performance after adding constraints are reported. Here, we summarize the findings of these experiments by answering the following questions.

What are the key differences in the performance of inference-time and training-time integration? Notably, using ILP only in inference time outperformed other baselines in most of the tasks. However, it fails to perform better than the training-time integration methods when the base model is wildly inaccurate in generating the probabilities for the final decisions. This phenomenon happened in our experiments in the semi-supervised setting and can be seen when comparing rows [#44, #45] to #46. In this case, inference alone cannot help correct the model, and global constraints should be used as a source of supervision to assist with the learning process.

ILP performs better than the training-time methods when applied to simpler baselines (see column simple baseline performance). However, the amount of improvement does not differ significantly when applying ILP to the simpler baselines compared to the strong ones. Additionally, the training-time methods perform relatively better on simpler baselines than the strong ones (either the drop is less or the improvement is higher) (compare columns 'Strong Baseline' and 'Simple Baseline' for '+PD' and '+SampL' rows.)

[2]https://hlr.github.io/gluecons/

	Mut Excl.	Seq. Struc.	Lin. Const	Log. Const	Log + Quan	Prog Const
Softmax	✓	✗	✗	✗	✗	✗
PD	✓	✓	✓	NC	NC	✗
SampL	✓	✓	✓	✓	✓	✓
ILP	✓	✓	✓	NC	NC	✗
A^*	✓	✓	NG	NG	NG	✗

Table 2: The limitation of integration methods with respect to constraint types. [KEYS: NC= Needs Conversion, NG= No Generalization, Mut Excl.= Mutual Exclusivity, Seq.= Sequential, Struc.=Structure, Lin.= Linear, Log.= Logical, Const.= Constraint, Quan.=Quantifiers, Prog Const= Any Constraints encoded as a program, A^* is abbreviation for the A^* Search algorithm.]

How does the size of data affect the performance of the integration techniques? The integration methods are exceptionally effective in the low-data regime when the constraints come from external knowledge or structural information. This becomes evident when we compare the results of 'EMR' and 'BIO tagging' with the 'Self Consistency in Decision Dependency' tasks in column 'Low data/ Performance'. This is because such constraints can inject additional information into the models, compensating for the lack of training data. However, when constraints are built over the self-consistency of decisions, they are less helpful in low-data regimes (rows #12 to #29), though a positive impact is still visible in many cases. This observation can be justified since there are fewer applicable global constraints in-between examples in the low-data regime. Typically, batches of the full data may contain tens of relationships leading to consistency constraints over their output, while batches of the low data may contain fewer relationships. The same observation is also seen as batch sizes for training are smaller.

Does constraint integration reduce the constraint violation? Since our inference-time integration methods are searching for a solution consistent with the constraints, they always have a constraint violation rate of 0%. However, training-time integration methods cannot fully guarantee consistency. However, it is worth noting these methods have successfully reduced the constraint violation in our experiments even when the performance of the models is not substantially improved or is even slightly hurt (see rows #18 and #20, rows #24 and #26, and rows #30 to #32). In general, SampL had a more significant impact than PD on making models consistent with the available task knowledge (compare rows with '+PD' and '+SampL' in column 'Constraint Violation').

	Tasks	#	Models	Strong Baseline Performance	Simple Baseline Performance	Low data Size	Low data Performance	Constraint Violation*	Run-Time^{ms} Training	Run-Time^{ms} Inference
Classification with Label Dependency	Simple Img Cls^{F1}	1	Model	94.23%	87.34%	5%	88.78%	7.17%	34	27.5
		2	+ PD	↑0.14%	↓1.14%		↑4.40%	8.32%	36.6	-
		3	+ SampL	↓1.17%	↑0.49%		↑3.19%	9.04%	39	-
		4	+ ILP	↑0.24%	↑1.60%		↑1.70%	-	-	31.5
		5	+ SampL + ILP	↓0.52%	↑2.02%		↑4.40%	-	-	-
		6	+ PD + ILP	↑0.32%	↑0.39%		↑4.40%	-	-	-
	Hierarchical Img Cls^{F1}	7	Model	58.03%	52.54%	10%	31.33%	39.26%	55.3	48.43
		8	+ SampL	↑0.39%	↑0.54%		↑2.18%	36.57%	58.2	-
		9	+ ILP	↑2.88%	↑3.18%		↑1.90%	-	-	55.2
		10	+ SampL + ILP	↑2.42%	↑3.52%		↑3.82%	-	58.2	55.2
Self Consistency in Decision Dependency	Causal^{A} Reasoning	12	Model	74.77%	73.80%	30%	60.49%	8.60%	104	46.2
		13	+ PD	↑2.17%	↑1.98%		↑1.10%	11.36%	118.1	-
		14	+ SampL	↑2.54%	↑2.17%		↑1.63%	4.37%	119.4	-
		15	+ ILP	↑4.03%	↑4.51%		↑1.88%	-	-	59.2
		16	+ SampL + ILP	↑4.15%	↑4.25%		↑2.11%	-	-	-
		17	+ PD + ILP	↑3.60%	↑4.30%		↑1.76%	-	-	-
	NLI^{A}	18	Model	74.00%	-	10%	68.65%	9.48%	29.2	10.7
		19	+ PD	↑0.25%	-		↑3.25%	7.26%	31.7	-
		20	+ SampL	↑0.55%	-		↑0.95%	5.00%	29.8	-
		21	+ ILP	↑8.90%	-		↑7.75%	-	-	14.3
		22	+ SampL + ILP	↑8.20%	-		↑7.05%	-	-	-
		23	+ PD + ILP	↑8.75%	-		↑10.1%	-	-	-
	Belief^{F1} Network	24	Model	94.90%	84.46%	25%	94.36%	0.22%	8.3	7.57
		25	+ PD	↑0.94%	↑0.87%		↓0.49%	0.16%	23.59	-
		26	+ SampL	↓0.29%	↓0.95%		↓3.03%	0.01%	8.5	-
		27	+ ILP	↑0.21%	↓0.10%		↓0.97%	-	-	11
		28	+ SampL + ILP	↑1.10%	↓3.19%		↓2.31%	-	-	-
		29	+ PD + ILP	↑2.68%	↑1.60%		↑0.51%	-	-	-
Consistency with EK	EMR^{F1}	30	Model	90.15%	85.22%	20%	82.00%	20.32%	210	200
		31	+ PD	↓1.00%	↓0.30%		↑2.42%	16%	245	-
		32	+ SampL	↓0.30%	↑0.50%		↑3.36%	16.7%	280	-
		33	+ ILP	↑3.02%	↑4.10%		↑8.86%	-	-	226
		34	+ SampL + ILP	↑2.40%	-		↑7.83%	-	-	-
		35	+ PD + ILP	↑1.64%	-		↑8.15%	-	-	-
Structural Consistency	BIO^{F1} Tagging	36	Model	89.56%	82.77%	30%	75.36%	2.19%	361.2	263.2
		37	+ PD	↑0.97%	↑0.04%		↑1.25%	0.99%	389.1	-
		38	+ SampL	↓0.17%	↑0.73%		↑2.62%	0.16%	429.8	-
		39	+ ILP	↑0.61%	↑2.96%		↑3.01%	-	-	312
		40	+ SampL + ILP	↑0.08%	↑2.43%		↑2.80%	-	-	-
		41	+ PD + ILP	↑1.07%	↑1.83%		↑2.73%	-	-	-
		42	+A^* search	↑0.59%	↑2.97%		↑3.03%	-	-	-
Constraints in (Un/Semi)Supervision	Arithmetic Supervision for Digit Classification^{A}	43	Model	9.01%	-	5%	10.32%	96.92%	13.6	-
		44	+ PD	↑89.39%	-		↑85.01%	3.18%	197	-
		45	+ SampL	↑89.55%	-		↑85.60%	2.86%	90.2	-
		46	+ ILP	↓2.11%	-		0.00%	-	-	-
		47	+Supervised	↑89.53%	-		↑84.30%	2.86%	12.5	-
	Sudoku^{CS}	48	PD	96.00%	-	$6*6$ Table	100%	3.7%	-	-
		49	SampL	87.00%	-		100%	18.88%	-	-
		50	ILP	100%	-		100%	-	-	-

Table 3: Impact of constraint integration. F1, A, and CS are F1-measure, accuracy, and constraint satisfaction metrics, respectively, showing model performance. The full data of the Sudoku task is a $9 * 9$ table. *: The 'Constraint Violation' values are calculated for the strong baselines trained with full data. ms: Run-Time is computed per example/batch and is reported in milliseconds. ↑ indicates improvement and ↓ indicates a drop in the performance compared to the initial Model. Run times are recorded on a machine with Intel Core i9-9820X (10 cores, 3.30 GHz) CPU and Titan RTX with NVLink as GPU. [KEYS: EK= external knowledge]

How do the integration methods perform on simpler baselines? According to our experiments, there is a significant difference between the performance of the integration methods applied to simple and strong baselines when the source of constraint was external (BIO tagging, EMR, Simple Image Cls, and Hierarchical Image Cls tasks). Moreover, we find that ILP applied to a simple baseline can sometimes achieve a better outcome than a strong model without constraints. This is, in particular, seen in the two cases of EMR and Causal Reasoning, where the difference between the simple and strong baselines is in using a pre-trained model. Thus, explicitly integrating knowledge can reduce the need for pre-training. In such settings, constraint integration compensates for pre-training a network with vast amounts of data for injecting domain knowledge for specific tasks. Additionally, the substantial influence of integration methods

on simple baselines compared to strong ones in these specific tasks indicates that constraint integration is more effective when knowledge is not presumably learned (at some level) by available patterns in historical data used in the pre-training of large language models.

How much time overhead is added through the integration process? While the inference-time method (ILP) has a computational overhead during inference, we have shown that this overhead can be minimized if a proper tool is used to solve the optimization problem (here, we use Gurobi[3]. It should be noted that training-time integration methods do not introduce additional overhead during inference; however, they typically have a high computational cost during training. In the case of our baselines, SampL has shown to be relatively more expensive than PD. This is because SampL has an additional cost for forming samples and evaluating the satisfaction of each sample.

What is the effect of combining inference-time and training-time integration methods? Our results show that combining inference-time and training-time methods mainly yields the highest performance on multiple tasks. For example, the performance on the NLI task on low-data can yield over 10% improvement with the combination of PD and ILP, while ILP on its own can only improve around 7%. The rationale behind these observations needs to be further investigated. However, this can be attributed to better local predictions of the training-time integration methods that make the inference-time prediction more accurate. A more considerable improvement is achieved over the initial models when these predictions are paired with global constraints during ILP (see rows #16, #28, #29, and #41).

What type of constraints can be integrated using each method? Table 2 summarizes the limitations of each constraint integration method to encode a specific type of knowledge. We have included "Softmax" in this table since it can be used to support mutual exclusivity directly in DNN. However, "Softmax" or similar functions are not extendable to more general forms of constraints. SampL is the most powerful method that is capable of encoding any arbitrary program as a constraint. This is because it only needs to evaluate each constraint based on its satisfaction or violation. A linear constraint can be directly imposed by PD and ILP methods. However, first-order logic constraints must be converted to linear constraints before they can be directly applied. Still, PD and ILP methods fail to generalize to any arbitrary programs as constraints. The A^* search can generally be used for mutual exclusivity and sequential constraints, but it cannot provide a generic solution for complex constraints as it requires finding a corresponding heuristic. (Chang, Ratinov, and Roth 2012) show A^* with constraints can be applied under certain conditions and when the feature function is decomposable.

7 Conclusion and Future Work

This paper presented a new benchmark, GLUECons for constraint integration with deep neural networks. GLUECons contains nine different tasks supporting a range of applications in natural language and computer vision domains. Given this benchmark, we evaluated the influence of the constraint integration methods beyond the tasks' performance by introducing new evaluation criteria that can cover the broader aspects of the effectiveness and efficiency of the knowledge integration. We investigated and compared methods for integration during training and inference. Our results indicate that, except in a few cases, inference-time integration outperforms the training-time integration techniques, showing that training-time integration methods have yet to be fully explored to achieve their full potential in improving the DNNs. Our experiments show different behaviors of evaluated methods across tasks, which is one of the main contributions of our proposed benchmark. This benchmark can serve the research community around this area to evaluate their new techniques against a set of tasks and configurations to analyze multiple aspects of new techniques. In the future, we plan to extend the tasks of this benchmark to include more applications, such as spatial reasoning over natural language (Mirzaee et al. 2021b), visual question answering (Huang et al. 2021), procedural reasoning (Faghihi and Kordjamshidi 2021; Dalvi et al. 2019), and event-event relationship extraction (Wang et al. 2020).

[3]https://www.gurobi.com/

Acknowledgments

This project is supported by National Science Foundation (NSF) CAREER award 2028626 and partially supported by the Office of Naval Research (ONR) grant N00014-20-1-2005. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation nor the Office of Naval Research.

References

Ahmed, K.; Li, T.; Ton, T.; Guo, Q.; Chang, K.-W.; Kordjamshidi, P.; Srikumar, V.; Van den Broeck, G.; and Singh, S. 2022. PYLON: A PyTorch framework for learning with constraints. In *NeurIPS 2021 Competitions and Demonstrations Track*, 319–324. PMLR.

Amizadeh, S.; Palangi, H.; Polozov, A.; Huang, Y.; and Koishida, K. 2020. Neuro-Symbolic Visual Reasoning: Disentangling. In *ICML*, 279–290. PMLR.

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2017. Guided Open Vocabulary Image Captioning with Constrained Beam Search. In *EMNLP*, 936–945.

Asai, A.; and Hajishirzi, H. 2020. Logic-guided data augmentation and regularization for consistent question answering. *arXiv preprint arXiv:2004.10157*.

Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le, Q.; et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *JMLR*, 18: 1–67.

Bloice, M. D.; Roth, P. M.; and Holzinger, A. 2020. Performing Arithmetic Using a Neural Network Trained on Digit Permutation Pairs. In *ISMIS*, 255–264. Springer.

Borgeaud, S.; and Emerson, G. 2020. Leveraging Sentence Similarity in Natural Language Generation: Improving Beam Search using Range Voting. In *4th WNGT*, 97–109.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, 632–642. Lisbon, Portugal: Association for Computational Linguistics.

Brundage, M.; Avin, S.; Wang, J.; Belfield, H.; Krueger, G.; Hadfield, G.; Khlaaf, H.; Yang, J.; Toner, H.; Fong, R.; et al. 2020. Toward trustworthy AI development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*.

Camburu, O.-M.; Shillingford, B.; Minervini, P.; Lukasiewicz, T.; and Blunsom, P. 2019. Make up your mind! adversarial generation of inconsistent natural language explanations. *arXiv preprint arXiv:1910.03065*.

Chang, M.-W.; Ratinov, L.; and Roth, D. 2012. Structured learning with constrained conditional models. *Machine learning*, 88(3): 399–431.

Clark, P.; Tafjord, O.; and Richardson, K. 2020. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*.

Clarke, J.; Goldwasser, D.; Chang, M.-W.; and Roth, D. 2010. Driving semantic parsing from the world's response. In *14th CoNLL*, 18–27.

Constantinou, A. C.; Fenton, N.; and Neil, M. 2016. Integrating expert knowledge with data in Bayesian networks: Preserving data-driven expectations when the expert variables remain unobserved. *Expert systems with applications*, 56: 197–208.

Dahlmeier, D.; and Ng, H. T. 2012. A beam-search decoder for grammatical error correction. In *EMNLP 2012*, 568–578.

Dalvi, B.; Tandon, N.; Bosselut, A.; Yih, W.-t.; and Clark, P. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *(EMNLP-IJCNLP)*, 4496–4505.

Dash, T.; Chitlangia, S.; Ahuja, A.; and Srinivasan, A. 2022. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1): 1–15.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI*, volume 7, 2462–2467. Hyderabad.

De Raedt, L.; Manhaeve, R.; Dumancic, S.; Demeester, T.; and Kimmig, A. 2019. Neuro-symbolic= neural+ logical+ probabilistic. In *NeSy'19 workshop @ IJCAI*.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Deutsch, D.; Upadhyay, S.; and Roth, D. 2019. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd CoNLL*, 482–492.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ellis, K.; Wong, C.; Nye, M.; Sablé-Meyer, M.; Morales, L.; Hewitt, L.; Cary, L.; Solar-Lezama, A.; and Tenenbaum, J. B. 2021. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN PLDI*, 835–850.

Faghihi, H. R.; Guo, Q.; Uszok, A.; Nafar, A.; and Kordjamshidi, P. 2021. DomiKnowS: A Library for Integration of Symbolic Domain Knowledge in Deep Learning. In *EMNLP: System Demonstrations*, 231–241.

Faghihi, H. R.; and Kordjamshidi, P. 2021. Time-Stamped Language Model: Teaching Language Models to Understand The Flow of Events. In *NAACL*, 4560–4570.

Gardner, M.; Artzi, Y.; Basmova, V.; Berant, J.; Bogin, B.; Chen, S.; Dasigi, P.; Dua, D.; Elazar, Y.; Gottumukkala, A.; Gupta, N.; Hajishirzi, H.; Ilharco, G.; Khashabi, D.; Lin, K.; Liu, J.; Liu, N. F.; Mulcaire, P.; Ning, Q.; Singh, S.; Smith, N. A.; Subramanian, S.; Tsarfaty, R.; Wallace, E.; Zhang, A.; and Zhou, B. 2020. Evaluating NLP Models via Contrast Sets. *ArXiv*, abs/2004.02709.

Guo, Q.; Faghihi, H. R.; Zhang, Y.; Uszok, A.; and Kordjamshidi, P. 2021. Inference-masked loss for deep structured output learning. In *29th IJCAI*, 2754–2761.

Hargreaves, J.; Vlachos, A.; and Emerson, G. 2021. Incremental Beam Manipulation for Natural Language Generation. In *16th EACL*, 2563–2574.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing Deep Neural Networks with Logic Rules. In *54th ACL*, 2410–2420.

Huang, J.; Li, Z.; Chen, B.; Samel, K.; Naik, M.; Song, L.; and Si, X. 2021. Scallop: From Probabilistic Deductive Databases to Scalable Differentiable Reasoning. *Neurips*.

Kassner, N.; Tafjord, O.; Schütze, H.; and Clark, P. 2021. BeliefBank: Adding Memory to a Pre-Trained Language Model for a Systematic Notion of Belief. *arXiv preprint arXiv:2109.14723*.

Kordjamshidi, P.; Khashabi, D.; Christodoulopoulos, C.; Mangipudi, B.; Singh, S.; and Roth, D. 2016. Better call Saul: Flexible Programming for Learning and Inference in NLP. In *COLING*, 3030–3040. Osaka, Japan: COLING.

Kordjamshidi, P.; Roth, D.; and Wu, H. 2015. Saul: Towards declarative learning based programming. In *2015 AAAI Fall Symposium Series*.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25.

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

Lee, C.; Gottschlich, J.; and Roth, D. 2021. Toward Code Generation: A Survey and Lessons from Semantic Parsing. *arXiv preprint arXiv:2105.03317*.

Lee, J. Y.; Mehta, S. V.; Wick, M.; Tristan, J.-B.; and Carbonell, J. 2019. Gradient-based inference for networks with output constraints. In *AAAI*, volume 33, 4147–4154.

Lee, J. Y.; Wick, M. L.; Tristan, J.-B.; and Carbonell, J. G. 2017. Enforcing Output Constraints via SGD: A Step Towards Neural Lagrangian Relaxation. In *AKBC@ NIPS*.

Li, R.; Wang, X.; and Yu, H. 2020. MetaMT, a meta learning method leveraging multiple domain data for low resource machine translation. In *AAAI*, volume 34, 8245–8252.

Li, T.; Gupta, V.; Mehta, M.; and Srikumar, V. 2019. A Logic-Driven Framework for Consistency of Neural Models. In *EMNLP-IJCNLP*, 3924–3935.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Lu, X.; West, P.; Zellers, R.; Le Bras, R.; Bhagavatula, C.; and Choi, Y. 2021. NeuroLogic Decoding:(Un) supervised Neural Text

Generation with Predicate Logic Constraints. In *NAACL*, 4288–4299.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and Raedt, L. D. 2018. DeepProbLog: Neural Probabilistic Logic Programming. *NeurIPS*. Code is available.

Mathews, S. M. 2019. Explainable artificial intelligence applications in NLP, biomedical, and malware classification: a literature review. In *Intelligent computing:proceedings of the computing conference*, 1269–1292. Springer.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Minervini, P.; and Riedel, S. 2018. Adversarially regularising neural NLI models to integrate logical background knowledge. *arXiv preprint arXiv:1808.08609*.

Mirzaee, R.; Faghihi, H. R.; Ning, Q.; and Kordjamshidi, P. 2021a. SPARTQA: A Textual Question Answering Benchmark for Spatial Reasoning. In *NAACL*, 4582–4598.

Mirzaee, R.; Rajaby Faghihi, H.; Ning, Q.; and Kordjamshidi, P. 2021b. SPARTQA: A Textual Question Answering Benchmark for Spatial Reasoning. In *NAACL*, 4582–4598. Online: Association for Computational Linguistics.

Nandwani, Y.; Pathak, A.; and Singla, P. 2019. A primal dual formulation for deep learning with constraints. *NeurIPS*, 32.

Paszke, A.; and Gross, e. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *NeurIPS 32*, 8024–8035. Curran Associates, Inc.

Richardson, M.; and Domingos, P. 2006. Markov logic networks. *Machine learning*, 62(1): 107–136.

Roth, D.; and Srikumar, V. 2017. Integer Linear Programming formulations in Natural Language Processing. In *15th EACL: Tutorial Abstracts*.

Roth, D.; and Yih, W. 2005. Integer Linear Programming Inference for Conditional Random Fields. In *ICML*, 737–744.

Saeed, M.; Ahmadi, N.; Nakov, P.; and Papotti, P. 2021. RuleBERT: Teaching Soft Rules to Pre-Trained Language Models. In *EMNLP*, 1460–1476.

Sang, E. F.; and De Meulder, F. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Scholak, T.; Schucher, N.; and Bahdanau, D. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *EMNLP*, 9895–9901.

Schubotz, M.; Scharpf, P.; Dudhat, K.; Nagar, Y.; Hamborg, F.; and Gipp, B. 2018. Introducing mathqa: a math-aware question answering system. *IDD*.

Speer, R.; Chin, J.; and Havasi, C. 2016. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *CoRR*, abs/1612.03975.

Stewart, R.; and Ermon, S. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Tandon, N.; Dalvi, B.; Sakaguchi, K.; Clark, P.; and Bosselut, A. 2019. WIQA: A dataset for "What if..." reasoning over procedural text. In *EMNLP*.

von Rueden, L.; Mayer, S.; Beckh, K.; Georgiev, B.; Giesselbach, S.; Heese, R.; Kirsch, B.; Pfrommer, J.; Pick, A.; Ramamurthy, R.; et al. 2019. Informed Machine Learning–A Taxonomy and Survey of Integrating Knowledge into Learning Systems. *arXiv preprint arXiv:1903.12394*.

Wang, H.; Chen, M.; Zhang, H.; and Roth, D. 2020. Joint Constrained Learning for Event-Event Relation Extraction. *CoRR*, abs/2010.06727.

Winters, T.; Marra, G.; Manhaeve, R.; and De Raedt, L. 2021. Deepstochlog: Neural stochastic logic programming. *arXiv preprint arXiv:2106.12574*.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and Broeck, G. 2018. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, 5502–5511. PMLR.

Yang, T.-Y.; Hu, M. Y.; Chow, Y.; Ramadge, P. J.; and Narasimhan, K. 2021. Safe reinforcement learning with natural language constraints. *NeurIPS*, 34: 13794–13808.

Zheng, C.; and Kordjamshidi, P. 2022. Relevant CommonSense Subgraphs for "What if..." Procedural Reasoning. In *Findings of ACL 2022*, 1927–1933.

Zoph, B.; Yuret, D.; May, J.; and Knight, K. 2016. Transfer Learning for Low-Resource Neural Machine Translation. In *EMNLP*, 1568–1575.