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 is missing 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 et al., 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 et al., 2019), Knowledge graphs (Zheng and Kordjamshidi, 2022), Context-free grammars (Deutsch et al., 2019), Algebraic equations (Stewart and Ermon, 2017), or probabilistic relations (Constantinou et al., 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 et al., 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

[*] Work done during internship at Michigan State University

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 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 run-time, 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 et al., 2007; Richard-

[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.

son 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 et al., 2020). As of formal representations, knowledge integration has been addressed at both inference (Lee et al., 2019; Scholak et al., 2021; Dahlmeier and Ng, 2012) and training time (Hu et al., 2016; Nandwani et al., 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 et al., 2012).

These methods aim at finding a valid set of decisions given the constraints, while their objective is specified by the output of 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–we will see this later in our results too.

Prior research has investigated such integration by using variants of beam search (Hargreaves et al., 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 et al., 2012), finite-state/push-down Automata (Deutsch et al., 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 choose ILP as the current off-the-shelf tool performing a very efficient search and offering a natural way to integrate constraints as far as we can find a linear form for them (Faghihi et al., 2021; Kordjamshidi et al., 2015).

Training-Time Integration: Several recent techniques have been proposed for knowledge integration at training time (Nandwani et al., 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 et al., 2019; Asai and Hajishirzi, 2020), rule-regularized supervision (Hu et al., 2016; Guo et al., 2021), reenforcement 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 et al., 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 et al., 2021), program synthesize (Austin et al., 2021; Ellis et al., 2021), semantic parsing (Clarke et al., 2010; Lee et al., 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 runtime 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 et al., 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 et al., 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 et al., 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 includes 3993 questions. Following the work by (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^{ESIM}_{1000} (Minervini and Riedel, 2018a), 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, 2018a).

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 et al., 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:

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

"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 Constraints in (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 et al., 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:

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

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:

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

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 two approaches.

Primal-Dual (PD). This approach (Nandwani et al., 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 decision 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 $\mathcal{C}(y) \leq 0$.

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 hyperparameters, codes, models, and results can be found on our website[2]. The basic architectures for each task are shown in Table 1.

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

Task	Strong Baseline	Simple Baseline
Image Cls.	CNN + MLP	MLP
Hier. Image Cls.	Resnet18 + MLP	-
NLI	RoBERTa + MLP	-
Causal Rea.	RoBERTa + MLP	BERT + MLP
BIO Tagging	BERT + MLP	Bi-LSTM + MLP
NER	W2V + BERT + MLP	W2V + Bi-LSTM + MLP
Ari. Operation	CNN + MLP	-
BeliefNet Con.	RoBERTa + MLP	Word vectors + MLP
Sudoku	$(n * n * n)$ Vector directly learns probabilities	-

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, **Con.**=Consistency]. For the Sudoku, the model is not a generalizable DNN and the method uses the integration methods as an inference algorithm to solve one specific table.

	Mutual Excl.	Seq. structure	Lin. Const	Log. Const	Log Const + quantifier	Prog Const
Softmax	✓	✗	✗	✗	✗	✗
PD	✓	✓	✓	NC	NC	✗
SampL	✓	✓	✓	✓	✓	✓
ILP	✓	✓	✓	NC	NC	✗
A^* Search	✓	✓	NG	NG	NG	✗

Table 2: The limitation of integration methods based on different types of constraints. [**KEYS**: **NC**=Needs Conversion, **NG**=No Generalization, **Excl.**= Exclusivity, **Seq.**=Sequential, **Lin.**=Linear, **Log.**=Logical, **Const.**=Constraint, **Prog Const**=Any Constraints encoded as a program.]

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.)

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 the 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').

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

	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%	-	55.2
		9	+ ILP	↑2.88%	↑3.18%		↑1.90%	-	-	55.2
		10	+ SampL + ILP	↑2.42%	↑3.52%		↑3.82%	-	-	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%	1.17%	210	200
		31	+ PD	↓1.00%	↓0.30%		↑2.42%	0.94%	245	-
		32	+ SampL	↓0.30%	↑0.50%		↑3.36%	0.98%	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 to evaluate models' performance. The full data of the Sudoku task is a $9 * 9$ table. *: Constraint Violation is on the full data with the strong baselines. ms: Run-Time is computed per example/batch and is reported in milliseconds. ↑ indicates improvement over the initial Model performance. ↓ indicates a drop in the performance. 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]

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-raining 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.

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

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 et al., 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).

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

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck, and Sameer Singh. 2022. Pylon: A pytorch framework for learning with constraints. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 319–324. PMLR.

Saeed Amizadeh, Hamid Palangi, Alex Polozov, Yichen Huang, and Kazuhito Koishida. 2020. Neuro-symbolic visual reasoning: Disentangling. In *ICML*, pages 279–290. PMLR.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*, pages 936–945.

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. *arXiv preprint arXiv:2004.10157*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss markov random fields and probabilistic soft logic. *JMLR*, 18:1–67.

Marcus D Bloice, Peter M Roth, and Andreas Holzinger. 2020. Performing arithmetic using a neural network trained on digit permutation pairs. In *ISMIS*, pages 255–264. Springer.

Sebastian Borgeaud and Guy Emerson. 2020. Leveraging sentence similarity in natural language generation: Improving beam search using range voting. In *4th WNGT*, pages 97–109.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, et al. 2020. Toward trustworthy ai development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*.

Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, and Phil Blunsom. 2019. Make up your mind! adversarial generation of inconsistent natural language explanations. *arXiv preprint arXiv:1910.03065*.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine learning*, 88(3):399–431.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *14th CoNLL*, pages 18–27.

Anthony Costa Constantinou, Norman Fenton, and Martin Neil. 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.

Daniel Dahlmeier and Hwee Tou Ng. 2012. A beam-search decoder for grammatical error correction. In *EMNLP 2012*, pages 568–578.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4496–4505.

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

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad.

Luc De Raedt, Robin Manhaeve, Sebastijan Dumancic, Thomas Demeester, and Angelika Kimmig. 2019. Neuro-symbolic= neural+ logical+ probabilistic. In *NeSy'19 workshop @ IJCAI*.

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

Daniel Deutsch, Shyam Upadhyay, and Dan Roth. 2019. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd CoNLL*, pages 482–492.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. 2021. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN PLDI*, pages 835–850.

Hossein Rajaby Faghihi, Quan Guo, Andrzej Uszok, Aliakbar Nafar, and Parisa Kordjamshidi. 2021. Domiknows: A library for integration of symbolic domain knowledge in deep learning. In *EMNLP: System Demonstrations*, pages 231–241.

Hossein Rajaby Faghihi and Parisa Kordjamshidi. 2021. Time-stamped language model: Teaching language models to understand the flow of events. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4560–4570.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating nlp models via contrast sets. *ArXiv*, abs/2004.02709.

Quan Guo, Hossein Rajaby Faghihi, Yue Zhang, Andrzej Uszok, and Parisa Kordjamshidi. 2021. Inference-masked loss for deep structured output learning. In *29th IJCAI*, pages 2754–2761.

James Hargreaves, Andreas Vlachos, and Guy Emerson. 2021. Incremental beam manipulation for natural language generation. In *16th EACL*, pages 2563–2574.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *54th ACL*, pages 2410–2420.

Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. 2021. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. *Neurips*.

Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. 2021. Beliefbank: Adding memory to a pretrained language model for a systematic notion of belief. *arXiv preprint arXiv:2109.14723*.

Parisa Kordjamshidi, Dan Roth, and Hao Wu. 2015. Saul: Towards declarative learning based programming. In *2015 AAAI Fall Symposium Series*.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Celine Lee, Justin Gottschlich, and Dan Roth. 2021. Toward code generation: A survey and lessons from semantic parsing. *arXiv preprint arXiv:2105.03317*.

Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, and Jaime Carbonell. 2019. Gradient-based inference for networks with output constraints. In *AAAI*, volume 33, pages 4147–4154.

Jay Yoon Lee, Michael L Wick, Jean-Baptiste Tristan, and Jaime G Carbonell. 2017. Enforcing output constraints via sgd: A step towards neural lagrangian relaxation. In *AKBC@ NIPS*.

Rumeng Li, Xun Wang, and Hong Yu. 2020. Metamt, a meta learning method leveraging multiple domain data for low resource machine translation. In *AAAI*, volume 34, pages 8245–8252.

Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. 2019. A logic-driven framework for consistency of neural models. In *EMNLP-IJCNLP*, pages 3924–3935.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. In *NAACL*, pages 4288–4299.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. *NeurIPS*. Code is available.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pasquale Minervini and Sebastian Riedel. 2018a. Adversarially regularising neural nli models to integrate logical background knowledge. *arXiv preprint arXiv:1808.08609*.

Pasquale Minervini and Sebastian Riedel. 2018b. Adversarially regularising neural NLI models to integrate logical background knowledge. *CoRR*, abs/1808.08609.

Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. 2021a. Spartqa: A textual question answering benchmark for spatial reasoning. In *NAACL*, pages 4582–4598.

Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. 2021b. SPARTQA: A textual question answering benchmark for spatial reasoning. In *NAACL*, pages 4582–4598, Online. Association for Computational Linguistics.

Yatin Nandwani, Abhishek Pathak, and Parag Singla. 2019. A primal dual formulation for deep learning with constraints. *NeurIPS*, 32.

Adam Paszke and et al. Gross. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS 32*, pages 8024–8035. Curran Associates, Inc.

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

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML*, pages 737–744.

Dan Roth and Vivek Srikumar. 2017. Integer linear programming formulations in natural language processing. In *15th EACL: Tutorial Abstracts*.

Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. 2021. Rulebert: Teaching soft rules to pre-trained language models. In *EMNLP*, pages 1460–1476.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *EMNLP*, pages 9895–9901.

Moritz Schubotz, Philipp Scharpf, Kaushal Dudhat, Yash Nagar, Felix Hamborg, and Bela Gipp. 2018. Introducing mathqa: a math-aware question answering system. *IDD*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975.

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

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. WIQA: A dataset for "what if..." reasoning over procedural text. In *EMNLP*.

Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. 2019. Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394*.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. Joint constrained learning for event-event relation extraction. *CoRR*, abs/2010.06727.

Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. 2021. Deepstochlog: Neural stochastic logic programming. *arXiv preprint arXiv:2106.12574*.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. 2018. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, pages 5502–5511. PMLR.

Tsung-Yen Yang, Michael Y Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. 2021. Safe reinforcement learning with natural language constraints. *NeurIPS*, 34:13794–13808.

Chen Zheng and Parisa Kordjamshidi. 2022. Relevant commonsense subgraphs for "what if..." procedural reasoning. In *Findings of ACL 2022*, pages 1927–1933.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *EMNLP*, pages 1568–1575.

A Appendix

Here, we describe experimental details with the selected tasks and baselines. All the execution times are recorded on a machine with Intel Core i9-9820X (10 cores, 3.30 GHz) CPU and Titan RTX with NVLink as GPU.

A.1 Simple Image Classification

Task MNIST Binary is a simple and standard dataset in which handwritten numbers from 0 to 9 appear in the form of $28 * 28$ pixels of an image to be classified. However, a typical model used to solve it does not predict each label independently. Designing our model in a way that predicts each class individually and independently, in combination with a low data regime, will make this dataset somewhat challenging.

Experiments To solve MNIST classification, which is the benchmark for our simple image classification, we define one model per class. Each class has two CNN layers with kernel sizes of 5, followed by a linear layer. The images are normalized with the typical values of 0.1307 and 0.3081. The model trains with a batch size of 30 for three epochs with the Adam optimizer and the learning rate of $2e-3$. At the end of the final evaluation, we show the averaged F1 measure for all the classes. The results are shown in table 4.

Simple Baseline To create a simple baseline for this task, we flatten the images to form a vector of size $1 * 784$ instead of a $28 * 28$ image. Then we input this vector to a single-layer MLP. In this simple baseline, spatial data will be lost, but due to the simplicity of the dataset, the results are still near 90%.

A.2 Hierarchical Image Classification

Task The task here is to classify images of the size $32 * 32 * 3$ into 100 child classes and 20 parent classes to form a hierarchical structure. This dataset consists of 50k train examples and 10k test examples. We randomly select 10% of the data for the **low data regime**.

Experiments The main architecture for this task is ResNet50 with Two linear layers on top. We have designed two separate models for super-classes and sub-classes. We use the Adam optimizer with a learning rate of 0.001. We train the models for 20 epochs for the baseline. For primal-dual and sampling loss, we train 20 epochs after ten epochs of training normally. As for the evaluation metric, we use the average of the accuracy values of super-classes and sub-classes.

The simple Baseline for this task is ResNet18 which provides a smaller model compared to ResNet50 and leads to a drop in a few percentages of accuracy.

A.3 Causal Reasoning

Task We use WIQA benchmark (Tandon et al., 2019) to evaluate the causal reasoning QA task. In particular, the WIQA benchmark contains 29808 training samples, 6894 development samples, and 3993 test data samples. Formally, the task of the WIQA is to predict an answer a from a set of candidate answers $A \in [$ more, less, no effect $]$ given question q and a document $\mathcal{C}$ that contains sentences $\mathcal{C} = \{c_1, \ldots, c_n\}$. Each data sample has a fixed triplet format (q, C, a). We enforce the constraints of symmetry and transitivity with the models for predicting consistent answers to a set of triplets. For example, the symmetry constraint is defined as follows: $symmetric(x) \Rightarrow F(q, C) \wedge \neg F(\neg q, C)$ where q and $\neg q$ represent the question and its antonym, C represents the same document, and $\neg F$ is the opposite of the answer F.

Data Scale	Baseline	Baseline+ILP	SampleLoss	SampleLoss+ILP	primal-dual	primal-dual+ILP
100%	94.23	94.47	93.06	93.71	94.37	94.55
5%	88.78	90.48	91.97	93.18	93.18	93.18

Table 4: MNIST Binary results in the form of macro F1 over all classes with various methods and limited training data.

Models	in-para	out-of-para	no-effect	Test Acc
Majority	45.46	49.47	55.0	30.66
Adaboost	49.41	36.61	48.42	43.93
Decomp-Attn	56.31	48.56	73.42	59.48
BERT (Tandon et al., 2019)	79.68	56.13	89.38	73.80
RoBERTa (Tandon et al., 2019)	74.55	61.29	89.47	74.77
Logic-Guided (Asai and Hajishirzi, 2020)	-	-	-	78.50
RoBERTa + Sampling loss	77.19	63.28	90.25	77.65
RoBERTa + ILP	**78.55**	**65.12**	**90.02**	**78.98**
Human	-	-	-	96.33

Table 5: Model Comparisons on WIQA benchmark. The evaluation metric of WIQA test data includes four categories: in-paragraph, out-of-paragraph, no effect, and overall test accuracy.

NLI Consistency Rules
$\top \Rightarrow \text{ent}(h_1, h_1)$
$\text{con}(h_1, h_2) \Rightarrow \text{con}(h_2, h_1)$
$\text{ent}(h_1, h_2) \Rightarrow \neg\,\text{con}(h_2, h_1)$
$\text{neu}(h_1, h_2) \Rightarrow \neg\,\text{con}(h_2, h_1)$
$\text{ent}(h_1, h_2) \wedge \text{ent}(h_2, h_3) \Rightarrow \text{ent}(h_1, h_3)$

Table 6: The constraints of the NLI task. $\text{con}(X, Y)$, where X is the hypothesis and Y is the premise is *True* if the X contradicts Y. ent and neu represent the entailment and undetermined relationship respectively (Minervini and Riedel, 2018a). $ESIM$ (Minervini and Riedel, 2018b)

Experiments We use the large-scaled pre-trained language model, RoBERTa, as the backbone architecture that initially followed (Asai and Hajishirzi, 2020). Besides, we use 2 linear layers of MLP to predict the causal reasoning answer. We keep 128 tokens as the max length for the question and 256 tokens as the max length for the paragraph in each data sample. We set the batch size of the data to 8 and train the model using 10 epochs. The learning rate of our model is $1e-2$. The model is optimized by Adam optimizer. Table 5 shows the model performance on the WIQA benchmark compared to other strong baseline architectures. After integrating the constraints of symmetry and transitivity, we observe that the RoBERTa architecture using sampling loss improves 2.88% over the RoBERTa baseline model. Moreover, the RoBERTa architecture, including ILP as inference, has a 4.21% improvement.

A.4 Natural Language Inference

Task

Natural Language Inference (NLI) is the task of evaluating a hypothesis given a premise. The constraints applicable to this task are summarized in Table 6.

Experiments

For this task, we use a pre-trained RoBERTa base model with two layers of Multilayer perceptron(MLP) to predict a hypothesis given a premise. To encode the input pair of (Hypothesis, Premise), we concatenate them together and use pre-trained RoBERTa. Here, we use an AdamW optimizer with a $1e-5$ learning rate along with cross-entropy loss and set the batch size of the data to 16. The 10% and 100% of training data from SNLI (Bowman et al., 2015) are used to train the model and train each model for 5 epochs. We use two benchmarks to evaluate the models, SNLI for standard evaluation and A^{ESIM}_{1000} for constraint-focused evaluation. The results from training are shown in table 7. According to the table, our experiments show no significant improvement from training using either Primal-Dual (PD) or Sampling loss alone. However, ILP as an inference-time integration significantly improves the accuracy on A^{ESIM}_{1000} around $6-8\%$ on both Primal-Dual and Sampling-loss. We also observe the same effectiveness for using ILP on the model trained with only 10% of data. Furthermore, Primal-Dual and Sampling-Loss do not help to reduce constrain violations due to the low violation rate on the baseline (5% violations). Sampling Loss takes around $31.25ms$ for each sample. Meanwhile, Primal-Dual takes up to $43.12ms$ more than baseline. ILP during inference

adds around $4ms$ for each sample in each training method. The best method for this task is the combination of Sampling Loss + ILP.

A.5 Belief Network Consistency

Task In this task, each example has an entity and the corresponding fact. We concatenate the entity with the facts to form the input text for the models. Each fact can be either $True$ or $False$.

The Train set, of the size $1.8k$, is small compared to the test set, of size $20k$, by design. Also, one-third of the Train set is set aside as the dev set.

Experiments The model we use is RoBERTa-base (Liu et al., 2019) topped with two linear layers. In our model, all but the last two transformer layers are frozen. The length of the transformer input is 64. Since this dataset is easy to solve, we limit the transformer size to make the model less complex. The batch size is 128 to incorporate as many constraints between entities as possible. Here, we use the Adam optimizer with a learning rate of $2e-4$ and train the models for 15,and 30 epochs for the data sizes 100%,and 25%, respectively. The results for various methods are shown in table 8, in which the primal-dual + ILP method outperforms other methods.

Baseline Simple To create the simple baseline, we use the Word2vec (Spacy small) to obtain the representation for each sentence of size 96 and input it to a single layer of MLP. The training parameters are the same as the baseline model except for the learning rate, which is changed to $2e-3$.

A.6 Named Entity and Relation Extraction

Task

The named entity and relation extraction task is designed to evaluate the models' performance in detecting the entity types and their relationships in a document. To simplify the task setting, we ignore the span recognition task and consider phrases as given inputs. We further limit the candidates for relationship classification and only classify given pairs of entities based on the relationship that exists between them. Respectively, there is no 'None' class for the relationship classification task. The types of entities are 'person', 'organization', 'location', and 'other'. The possible types of relationships are 'live-in', 'work-for', 'located-in', 'kill', and 'orgbase-on'. Table 10 summarizes the existing relationship constraints between relation types and entity types. Other constraints are the mutual exclusivity between different named entity types for each mention or formally defined as:

$$\forall x \in entities$$
$$\text{IF } \text{Type}_\text{A}(x) \Rightarrow \neg \vee_{Type_B \in \text{entity\_types}}^{Type_B \neq \text{Type1}} \text{Type}_\text{B}(x),$$

where $\text{Type}_\text{A} \in$ entity_types and $\text{Type}_\text{Y}(x)$ is $True$ if x is of type Y. A similar mutual exclusivity constraint is also defined over the relationship types for each pair of entities.

Experiments

In the base model, we represent each token in the document using a pre-trained BERT-base model (Devlin et al., 2018)and a pre-trained Word2Vec model (Honnibal and Montani, 2017). We concatenate these two representations and take the mean over the tokens of each span to represent it. Then we feed these vectors to boolean classifiers for each entity class. To classify the relationships, we concatenate the representations of two entities and feed them to the boolean classifiers for each relationship type. As the CONLL2003 dataset does not provide train/validation/test splits, we randomly create those sets and further subsample the training set with 20% of the data to generate a low-resource setting. To create a simpler model as a baseline, we remove the BERT-base representation and add a Bi-LSTM layer after getting the representations from the Word2Vec from Spacy. For all experiments, the learning rate is set to $1e-2$, and they are reported on the best performing model on the dev set (in terms of macro-F1) as we trained the models for 200 epochs. All models are optimized by Adam optimizer.
Baselines: Pre-trained Bert + Spacy W2V + 1 Layer MLP **Baseline Simple:** Spacy W2V + LSTM + 1 Layer MLP

The detailed results of our baselines performance on both entity and relationship detection tasks are listed in Table 9. The results indicate the constraint integration methods are majorly biased toward a better relationship extraction. This is perhaps due to the fact that the relationship pairs have been limited to the set of known pairs, and the base performance of the relationship classifier is higher than the entity recognizer.

A.7 BIO Tagging

Task

we select the CONLL-2003 Shared Task benchmark to evaluate the BIO tagging task. The CONLL benchmark contains 14987 training samples, 3466 development samples, and 3684 test samples, including 9 tagging labels. The evaluation metrics in this task include Precision, Recall, and F1. Formally, the process of computing the score for a partial tagging sequence

Models	SNIL small training	SNIL large training	A_{1000}^{ESIM} small training	A_{1000}^{ESIM} large training
$ESIM$	-	87.25	-	60.78
$ESIM^{AR}$	-	87.55	-	73.32
DomiKnowS without constraints	88.16	89.65	69.90	71.60
DomiKnowS with constraints + ILP	88.65	90.11	77.35	80.25
Primal-Dual	88.38	90.04	70.05	73.50
Primal-Dual + ILP	88.38	90.04	**79.90**	79.05
Sampling Loss	88.19	90.11	68.65	74.55
Sampling Loss + ILP	88.26	**90.26**	76.30	**82.20**

Table 7: NLI accuracy results with various methods and limited training data

Model	25%	100%
Base Domiknows	94.36	94.90
Base Domiknows + ILP	93.39	95.11
Sample Loss	91.33	94.61
Sample Loss + ILP	92.05	96.0
primal dual	93.87	95.84
primal dual + ILP	**95.43**	**96.22**

Table 8: BeliefBank F1-measure results with various methods and limited training data. Since the test data size is large in this case, every small improvement is notable.

in the training time is calculated as follows:

$$f(y_{[1,...,T]}, W) = \sum_{i=1}^{T} logp(y_i|W),$$

where T is the sequence length, W is the learnable weight. We use $A*$ search over tag prefixes during the inference time to integrate constraints in the BIO tagging outputs. To integrate BIO constraints, Specifically, we add the constraint function $c \in C$, where C={'B-*' $\prec$ 'I-*', 'O' $\not\prec$ 'I-*' } , $\prec$ represents the 'before' relation, and $\not\prec$ represents the 'not before' relation. Formally, the process of a constrained tagging sequence in the inference time is computed as follows:

$$f(y_{[1,...,T]},W) = \sum_{i=1}^{T} logp(y_i|W) - \sum_{c \in C} c(y_{[1,...,T]}, W),$$

where T is the sequence length and W is the learnable weight.

Experiments

We separately apply BI-LSTM and BERT as backbone following a 2-layers of MLP to predict the sequence tagging for each sentence. We select the cross entropy as the loss function to train the model. The model is optimized by Adam optimizer while the learning rate of our model is $1e-3$. We set the batch size of the data to 64 and train the model using 20 epochs. Moreover, in the inference time, we use $A*$ search and ILP over tag prefixes to integrate constraints on the BIO tagging outputs. We use both small data scales (30%, 60%) and full data (100%) to train and evaluate the model performance. The results are shown in Table 11. The model takes 20 epochs after initializing the parameters with the BIO model. In the inference time, the model integrating $A*$ decoding has a 2.25% improvement on small data and 0.74% improvement on full data compared to the baseline. Furthermore, including ILP as inference on the top of the model improves 0.72% over the baseline model on full data. At the same time, we evaluate the time complexity between different models. The baseline model takes $1.01ms$ for each sample. The model integrating $A*$ constraints takes $1.32ms$, while the model integrating ILP constraints takes $1.34ms$. It shows that the inference time complexity has not increased significantly.

A.8 Arithmetic Operation with Supervision Task

The goal of the MNIST Arithmetic task is to train an MNIST digit classifier with the only supervision being the sum of digit pairs. Constraints for such a task would be to produce predictions whose sum matches the given sum:

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

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.

Experiments

Constraint violation or satisfaction corresponds to the rate at which the constraints for each sample are violated or satisfied, averaged across all samples. The

	25% training data			100% training data		
	Entity F1	Relation F1	Overall F1	Entity F1	Relation F1	Overall F1
Base Model	79.86	83.68	82	88.91	91.14	90.15
Base Model + ILP	82.14	**97.84**	**90.86**	**91.77**	**97.84**	**95.14**
Base Model + PD	82.12	86.25	84.42	81.3	93.56	88.12
Base Model + Sampling	**83.94**	86.51	85.36	85.90	93	89.85

Table 9: The results of applying constraints on the CONLL dataset for the named entity and relationship extraction task.

Relation Type	Argument 1	Argument 2
Live In	Person	Location
Org Base	Org	Location
Work For	Person	Org
Kill	Person	Person
Located In	Location	Location

Table 10: The domain knowledge about the relationship between relation types and entity types in the CONLL dataset.

Data	Models	Precision	Recall	F1
30%	Bi-LSTM	78.86	76.34	77.58
	Bi-LSTM+BIO+A∗	81.75	78.01	79.84
	Bi-LSTM+BIO+ILP	81.68	78.21	79.65
	BERT	84.12	78.43	81.17
	BERT+BIO+A∗	81.87	86.50	84.12
	BERT+BIO+ILP	81.76	86.68	84.15
60%	Bi-LSTM	78.03	80.46	79.93
	Bi-LSTM+BIO+A*	79.92	83.55	81.70
	Bi-LSTM+BIO+ILP	79.83	83.55	81.65
	BERT	87.80	87.76	87.78
	BERT+BIO+A*	93.21	84.08	88.40
	BERT+BIO+ILP	93.24	84.06	88.41
100%	Bi-LSTM	86.25	84.68	85.46
	Bi-LSTM+BIO+A*	87.26	85.17	86.20
	Bi-LSTM+BIO+ILP	87.28	85.14	86.18
	BERT	93.02	88.87	90.90
	BERT+BIO+A*	**93.64**	89.77	91.66
	BERT+BIO+ILP	93.59	**89.87**	**91.70**

Table 11: Model Comparisons on CONLL-2003 BIO-tagging test benchmark. A* is A* decoding.

constraint satisfaction rate and accuracy rate are comparable between each setting, with a slight drop in percentage points due to the fact that an image pair may predict correctly 1 out of 2 digits but still not fully satisfy the constraints.
The full results for accuracy and constraint violation are in Table 12 and Table 13.

For the digit classifiers, we use a simple LeNet-style CNN architecture (LeCun et al., 1998). The "explicit sum" model, in order to compute the summation probability distribution $P(S = s)$, sums the Softmax

Setting	Small Data	Large Data
Digit Labels	94.62	98.54
Primal-Dual	95.33	98.40
Sampling Loss	**95.92**	98.56
Semantic Loss	95.12	**98.62**
Explicit Sum	94.93	98.55
Explicit Sum + ILP	94.93	98.55
Baseline	10.32	9.01
Baseline + ILP	10.32	6.90

Table 12: Accuracy on various constraints/data settings for the MNIST Arithmetic task.

Setting	Small Data	Large Data
Digit Labels	10.24	2.86
Primal-Dual	12.52	3.18
Sampling Loss	7.96	2.86
Semantic Loss	9.44	2.76
Explicit Sum	9.78	2.88
Baseline	**94.50**	**96.92**

Table 13: Constraint violation rate for the MNIST Arithmetic task in the small and large data setting.

distributions of the two digits $P(D_1)$ and $P(D_2)$. i.e.

$$P(S = s) = \sum_{k=max(0,s-9)}^{min(s,9)} P(D_1 = k)P(D_2 = s - k)$$

We compare Primal-Dual and Sampling Loss with three other methods: 1) A baseline model consisting of a two-layer MLP on top of the digit classifier logits, supervised on the summation values instead of using constraints. 2) Training with digit labels directly (i.e. regular digit classification). 3) Explicitly specifying the summation constraints through the model architecture by directly computing the predicted summation value from the digit classifier logits (the "explicit sum" model). On both the small and large data settings, training with Primal-Dual, Sampling Loss, or Semantic Loss using constraints as the only source of supervision performs almost identically to both direct supervision on digit labels and the explicit sum model.

On the other hand, training with neither direct supervision nor constraints performs around random guessing in both the small and large training data cases. ILP does not improve the "explicit sum" case as it already achieves a high rate of constraint satisfaction, and it hurts the baseline case slightly as the baseline model probabilities are not informative. Similar results can be seen when looking at constraint satisfaction scores in the large data setting. In the small data setting, there's slightly more variability with Primal-Dual reaching a 12.52% constraint violation rate and Sampling Loss having just a 7.96% rate of constraint violation.

A.9 Sudoku

Task

The task of Sudoku is to predict the missing numbers from a $n * n$ Sudoku table such that the final table is valid according to the following rule: "There should not be two cells from each row/block/column, with the same value" or formally defined as:

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

where x and y are variables regarding the cells of the table, $i \in [0, n]$ for a $n*n$ Sudoku, $digit(x, i)$ is $True$ only if the value of x is predicted to be i.

For this task, we use an incomplete $9 * 9$ Sudoku as the harder task (large-data) and $6 * 6$ Sudoku representing a simpler task (low-data).

Experiments

Here, we use a simple learnable vector of size $n*n*n$, which stores the probability of each n assignments given all the cells of the table, i.e. if 0.6 is present in (1, 1, 2) index of the vector, the probability of row 1, column 1 being number 2 is predicted to be 60%. We train this vector using two available sources of supervision: 1) The input of the Sudoku through a CrossEntropy loss function. 2) The indirect loss function based on constraints when the input is masked. All models have been trained with a learning rate of $1e-1$ and use the SGD optimizer. Since the objective is to complete the Sudoku with correct numbers, we continue the training until the goal is accomplished or the 250 epochs have passed.