Human-Guided Learning of Column Networks: Knowledge Injection for Relational Deep Learning

Mayukh Das
mayukh.das@samsung.com
Samsung R&D Institute, Bangalore

Devendra Singh Dhami
devendra.dhami@utdallas.edu
The University of Texas at Dallas

Yang Yu
yangyu@hlt.utdallas.edu
The University of Texas at Dallas

Gautam Kunapuli
gautam.kunapuli@verisk.com
Verisk Analytics, Inc.

Sriraam Natarajan
sriraam.natarjan@utdallas.edu
The University of Texas at Dallas

ABSTRACT

Recently, deep models have been successfully adopted in several applications, especially where low-level representations are needed. However, sparse, noisy samples and structured domains (with multiple objects and interactions) are some of the open challenges in most deep models. Column Networks, a deep architecture, can succinctly capture domain structure and interactions, but may still be prone to sub-optimal learning from sparse and noisy samples. Inspired by the success of human-knowledge guided learning in AI, especially in data-scarce domains, we propose Knowledge-augmented Column Networks that leverage human advice/knowledge for better learning with noisy/sparse samples. Our experiments demonstrate that our approach leads to either superior overall performance or faster convergence (i.e., both effective and efficient).

ACM Reference Format:
Mayukh Das, Devendra Singh Dhami, Yang Yu, Gautam Kunapuli, and Sriraam Natarajan. 2021. Human-Guided Learning of Column Networks: Knowledge Injection for Relational Deep Learning. In *8th ACM IKDD CODS and 26th COMAD (CODS COMAD 2021), January 2–4, 2021, Bangalore, India.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3430984.3431018

1 INTRODUCTION

The re-emergence of Deep Learning (Goodfellow et al., 2016) has found significant and successful applications in complex real-world tasks such as image (Krizhevsky et al., 2012), audio (Lee et al., 2009) and video processing.However, the combinatorial complexity of reasoning in relational domains over a large number of relations and objects has remained a significant bottleneck to overcome. Recent work in relational deep learning has sought to address this particular challenge (França et al., 2014, Kaur et al., 2017, Kazemi and Poole, 2018, Šourek et al., 2015). **Column Networks**, CLN, (Pham et al., 2017), a deep architecture composed of several (feedforward) interconnected mini-columns each of which represents an entity in the domain, is a particularly promising approach for several reasons - (1) hidden layers of a CLN share parameters, which restricts the parameter space from exploding with increasing depth, (2) as the depth increases, the CLN can begin to model feature interactions of considerable complexity and well as long range relational dependencies and (3) learning and inference are linear in the size of the network and the number of relations, which makes CLNs highly efficient. In brief CLNs can fundamentally represent relational structure in an implicit fashion, unlike other graph-centric deep models which learn numerical embeddings of relational structures. However, as our evaluation also illustrates, CLNs have not overcome the necessity to rely on vast amounts of data for optimal learning since it does not leverage any knowledge about the problem domain, similar to most deep architectures. This problem is even more critical in structured domains since only a small fraction of relationships are actually true rendering implicit sample sparsity.

It is well known that inductive bias is necessary for optimal generalization over new instances (Mitchell, 1980). One of the fundamental forms of inductive bias comes from knowledge of the target domain/task. While deep learning does incorporate domain knowledge (for example, through parameter tying, convolutions, attention mechanisms or denoising encoders) but they are limited in their scope and treatment of such knowledge. We are motivated to develop systems that can incorporate richer and more general forms of domain knowledge. Human experts can guide learning by providing *rules over training examples and features*. The earliest such approaches combined explanation-based learning (EBL-NN, (Shavlik and Towell, 1989)) or symbolic domain rules with ANNs (KBANN, (Towell and Shavlik, 1994)). Another natural way a human could guide learning is by expressing *preferences* and has been studied extensively within the preference-elicitation framework due to Boutilier et al. (2006). We are inspired by this form of knowledge as they have been successful within the context of inverse reinforcement learning (Kunapuli et al., 2013), imitation learning (Odom et al., 2015) and planning (Das et al., 2018).

These approaches span diverse machine learning formalisms, and they all exhibit the same remarkable behavior: **better generalization with fewer training examples** because they effectively exploit and incorporate domain knowledge as an inductive bias. This is the prevailing motivation for our approach: to develop a framework that **allows a human to guide deep learning** by incorporating rules and constraints that define the domain and its aspects. Incorporation of prior knowledge into deep learning has begun to receive interest recently, for instance, the recent work on incorporating prior knowledge of color and scene information into deep learning for image classification (Ding et al., 2018). However,

CODS COMAD 2021, January 2–4, 2021, Bangalore, India

ACM ISBN 978-1-4503-8817-7/21/01…$15.00
https://doi.org/10.1145/3430984.3431018

in many such approaches, the guidance is not through a human, but rather through a pre-processing algorithm to generate guidance. Our framework is much more general in that a human provides guidance during learning. Furthermore, the human providing the domain knowledge is not an AI/ML expert but rather a domain expert who provides rules naturally. We exploit the rich representation power of relational methods to capture, represent and incorporate such rules into relational deep learning models. Note that our focus is not combining logic and deep networks as several others have explored this connection for decades since the origin of neuro-symbolic reasoning to more recent ILP-based neural models (Kaur et al., 2017, Kazemi and Poole, 2018) . We use first-order logic as a representation language for human knowledge and employ it in the context of CLNs.

We make the following contributions: (1) we propose the formalism of Knowledge-augmented Column Networks, (2) we present, inspired by previous work (such as KBANN), an approach to inject generalized domain knowledge in a CLN and develop the learning strategy that exploits this knowledge, and (3) we demonstrate, across four real problems in some of which CLNs have been previously employed, the effectiveness and efficiency of injecting domain knowledge. Specifically, our results across the domains clearly show statistically superior performance with small amounts of data. As far as we are aware, this is the first work on human-guided CLNs.

2 BACKGROUND AND RELATED WORK

Using domain knowledge as inductive bias to accelarate learning has long been explored (Fung et al., 2003, Kunapuli et al., 2010, Le et al., 2006a, Odom and Natarajan, 2018, Towell and Shavlik, 1994). Fu et al., (1995) presents a unified view of different variations of knowledge-based neural networks. Such knowledge based learning has been proposed for support vector machines (Fung et al., 2003, Le et al., 2006b) in propositional cases and probabilistic logic models (Odom and Natarajan, 2018) for relational cases. (Towell and Shavlik, 1994) introduce the KBANN algorithm which compiles first order logic rules into a neural network and (Kunapuli et al., 2010) present the first work on applying knowledge, in the form of constraints, to the perceptron. The knowledge-based neural network framework has been applied successfully to various real world problems such as recognizing genes in DNA sequences (Noordewier et al., 1991), , robotic control (Handelman et al., 1990) and recently in personalised learning systems (Melesko and Kurilovas, 2018). Combining relational (symbolic) and deep learning methods has recently gained significant research thrust since relational approaches are indispensable in faithful and explainable modeling of implicit domain structure, which is a major limitation in most deep architectures in spite of their success. While extensive literature exists that aim to combine the two (Battaglia et al., 2016, Lodhi, 2013, Rocktäschel et al., 2014, Sutskever et al., 2009), to the best of our knowledge, there has been little or no work on incorporating knowledge in any such framework.

Column networks transform relational structures into a deep architecture in a principled manner and are designed especially for collective classification tasks (Pham et al., 2017). The architecture and formulation of the column network are suited for adapting it to the advice framework. The GraphSAGE algorithm (Hamilton et al., 2017) shares similarities with column networks since both architectures operate by aggregating neighborhood information but differs in the way the aggregation is performed. Graph convolutional networks (Kipf and Welling, 2016) is another architecture that is very similar to the way CLN operates, again differing in the aggregation method. Diligenti et al., (2017) presents a method of incorporating constraints, as a regularization term, which are first order logic statements with fuzzy semantics, in a neural model and can be extended to collective classification problems. While it is similar in spirit to our proposed approach it differs in its representation and problem setup.

Several recent approaches aim to make deep architectures robust to label noise by either learning from easy samples with importance weights or by additional noise-adaptation layers or, may be, by regularization over virtual adversarial randomization (Goldberger and Ben-Reuven, 2017, Jiang et al., 2018, Miyato et al., 2018, Patrini et al., 2017).

While the above approaches enable effective learning of deep models in presence of noise, there are some fundamental differences with our problem setting.

(1) **[Type of noise]**: We aim to handle *systematic* noise (Odom and Natarajan, 2018) which is frequent in real-world due to cognitive bias or sample sparsity.
(2) **[Type of error]**: Systematic noise leads to generalization errors (see Example 1).
(3) **[Structured data]**: K-CLN works in the context of structured data (entities/relations). Though crucial, structured data is inherently sparse (most relations are false in the real world).
(4) **[Noise prior]**: Most noise handling approaches for deep models explicitly try to model the noise, which is impossible for systematic noise. K-CLN instead allows expert knowledge to guide the learner towards better generalization via an inductive bias.

Augmented learning with human knowledge has been proven to be an effective strategy in machine learning, probabilistic learning or sequential decision making, in presence of systematic noise (sparsity + sample bias + errors in data recording). Although, pseudo-labels introduced by Lee, (2013) are used for constructing efficient semi-supervised methods in deep learning, weak supervision is not always successful as it assumes presence of large amounts of data and certainly not the best approach with noisy data (since the pseudo-lables are derived from the fully observed label set where noise could propagate). Advice is typically provided **before** the data set is encountered i.e., by a domain expert and hence is independent of the fully labeled data (which can be noisy). Data programming (Ratner et al., 2016) can be viewed as constraining the data using weak labels and is orthogonal to our setting since which can be regarded as constraining the model or hypotheses space.

3 KNOWLEDGE-AUGMENTED COLUMN NETWORKS

3.1 Column Network: A brief background

Column Networks (Pham et al., 2017) allow for encoding interactions/relations between entities as well as the attributes of such

entities in a principled manner without explicit relational feature construction or vector embedding. This enables us to seamlessly transform a multi-relational knowledge graph into a deep architecture making them one of the robust *relational* deep models. Figure 1 illustrates an example column network, with respect to the knowledge graph on the left. Note how each entity forms its own column and relations are captured via the sparse inter-column connectors. Consider a graph $\mathcal{G} = (V, A)$, where $V = \{e_i\}_{i=1}^{|V|}$ is the set of vertices/entities. Without loss of generality, we assume only one entity type. A is the set of arcs/edges between two entities e_i and e_j denoted as $r(e_i, e_j)$. Note that the graph is multi-relational, *i.e.,* $r \in R$ where R is the set of relation types in the domain. To obtain the equivalent Column Network C from G, let x_i be the feature vector representing the attributes of an entity e_i and y_i its label predicted by the model[1]. h_i^t denotes a hidden node w.r.t. entity e_i at the hidden layer t ($t = 1, \ldots, T$ is the index of the hidden layers). The *context* between 2 consecutive layers captures the dependency of the immediate neighborhood (based on edges/inter-column connectors). For entity e_i, the context w.r.t. r and hidden nodes are computed as,

$$c_{ir}^t = \frac{1}{|\mathcal{N}_r(i)|} \sum_{j \in \mathcal{N}_r(i)} h_j^{t-1}; \tag{1}$$

$$h_i^t = g\left(b^t + W^t h_i^{t-1} + \frac{1}{z} \sum_{r \in R} V_r^t c_{ir}^t\right) \tag{2}$$

where $\mathcal{N}_r(i)$ are all the neighbors of e_i w.r.t. r in the knowledge graph $\mathcal{G}$. Note the absence of context connectors between h_2^t and h_4^t (Figure 1, *right*) since there does not exist any relation between e_2 and e_4 (Figure 1, *left*). The activation of the hidden nodes is computed as the sum of the bias, the weighted output of the previous hidden layer and the weighted contexts where $W^t \in \mathbb{R}^{K^t \times K^{t-1}}$ and $V_r^t \in R^{K^t \times K^{t-1}}$ are weight parameters and b^t is a bias for some activation function g. z is a pre-defined constant that controls the parameterized contexts from growing too large for complex relations. Setting z to the average number of neighbors of an entity is a reasonable assumption. The final output layer is a softmax over the pre-final layer, T, $P(y_i = \ell | h_i^T) = softmax\left(b_l + W_l h_i^T\right)$ where $\ell \in L$ is the label (L is the set of labels).

3.2 Problem Setting

For a clearer perspective of the problem we aim to address, let us consider the following example,

Example 1. *We wish to classify whether a published article is about carcinoid metastasis (Zuetenhorst and Taal, 2005) or is irrelevant, from a citation network, and textual features of articles. There are several challenges: (1) Data is implicitly sparse due to rarity of clinical studies, (2) Some articles may cite other articles about carcinoid and contain some textual features, but may actually address another topic and (3) Finally, the presence of systematic noise, introduced by the citation parser or uninformative abstracts.*

The above cases may lead to the model not being able to effectively capture certain dependencies, or converge slower, even if they are captured somewhere in the advanced layers of the deep network. Our approach attempts to alleviate this problem via augmented learning of Column Networks using human advice/knowledge. We formally define our problem in the following manner,

Given: A sparse multi-relational graph $\mathcal{G}$, attributes x_i of each entity (sparse or noisy) in $\mathcal{G}$, equivalent Column-Network C and access to a Human-expert
To Do: More effective and efficient collective classification by knowledge augmented training of $C(\theta)$, where $\theta = \langle \{W^t\}_1^T, \{V_r^t\}_{r \in R; t=1}^{t=T}, \{W_\ell\}_{\ell \in L} \rangle$ is the set of all the network parameters of C.

We develop ***K***nowledge-augmented ***C***o***L***umn ***N***etworks (K-CLN), that incorporates human-knowledge, for more effective and efficient learning from relational data (Figure 2 illustrates the overall architecture). While knowledge-based connectionist models are not entirely new, our formulation provides - (1) a principled approach for incorporating knowledge specified in an intuitive logic-based encoding/language (2) a deep model for collective classification in relational data.

3.3 Knowledge Representation

Any model specific encoding of domain knowledge, such as numeric constraints or modified loss functions etc., has limitations, namely (1) counter-intuitive to the humans since they are domain expert (2) the resulting framework is brittle and not generalizable. Consequently, we employ preferences (akin to IF-THEN statements) to capture human knowledge.

Definition 1. *A preference is a modified Horn clause,* $\wedge_{\mathsf{k},\mathsf{x}} \mathtt{Attr}_\mathsf{k}(\mathsf{E_x}) \wedge \ldots \wedge_{\mathsf{r} \in \mathsf{R},\mathsf{x},\mathsf{y}} \mathsf{r}(\mathsf{E_x}, \mathsf{E_y}) \Rightarrow [\mathtt{label}(\mathsf{E_z}, \ell_1) \uparrow; \mathtt{label}(\mathsf{E_k}, \ell_2) \downarrow]$ *where* $\ell_1, \ell_2 \in L$ *and the* $\mathsf{E_x}$ *are variables over entities,* $\mathtt{Attr}_\mathsf{k}(\mathsf{E_x})$ *are attributes of* E_x *and* r *is a relation.* $\uparrow$ *and* $\downarrow$ *indicate the preferred non-preferred labels respectively. Quantification is implicitly* $\forall$ *and hence dropped. We denote a set of preference rules as* $\mathfrak{P}$.

Note that we can always, either have just the preferred label in head of the clause and assume all others as non-preferred, or assume the entire expression as a single literal. Intuitively a rule can be interpreted as conditional rule, **IF [conditions hold] THEN label ℓ is preferred**. A preference rule can be partially instantiated as well, *i.e.*, or more of the variables may be substituted with constants.

Example 2. *For the prediction task mentioned in Example 1, a possible preference rule could be,*

$$\mathtt{hasWord}(\mathsf{E_1}, \text{“}\mathtt{AI}\text{”}) \wedge \mathtt{hasWord}(\mathsf{E_2}, \text{“}\mathtt{domain}\text{”}) \wedge$$
$$\mathtt{cites}(\mathsf{E_2}, \mathsf{E_1}) \Rightarrow \mathtt{label}(\mathsf{E_2}, \text{“}\mathtt{irrelevant}\text{”}) \uparrow$$

Intuitively, this rule denotes that an article is not a relevant clinical work to carcinoid metastasis if it cites an 'AI' article and contains the word "domain", since it is likely to be another AI article that uses carcinoid metastatis as an evaluation domain.

3.4 Knowledge Injection

Given that knowledge is provided as *partially-instantiated* preference rules $\mathfrak{P}$, more than one entity may satisfy a preference rule. Also, more than one preference rules may be applicable for a single entity. The main intuition is that we aim to consider the error of

[1]Note that since in our formulation every entity is uniquely indexed by i, we use e_i and i interchangeably

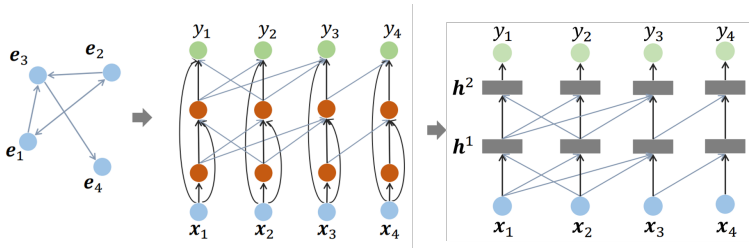

Figure 1: Original Column network [diagram src: (Pham et al., 2017)]

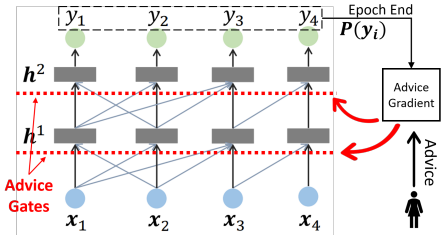

Figure 2: K-CLN architecture

the trained model w.r.t. both the data and the advice. Consequently, in addition to the *"data gradient"* as with original CLNs, there is a *"advice gradient"*. This gradient acts a feedback to augment the learned weight parameters (both column and context weights) towards the direction of the *advice gradient*. It must be mentioned that not all parameters will be augmented. Only the parameters w.r.t. the entities and relations (contexts) that satisfy $\mathfrak{P}$ should be affected. Let $\mathcal{P}$ be the set of entities and relations that satisfy the set of preference rules $\mathfrak{P}$. The hidden nodes (equation 1) can now be expressed as,

$$h_i^t = g\left(b^t + W^t h_i^{t-1} \Gamma_i^{(W)} + \frac{1}{z}\sum_{r\in R} V_r^t c_{ir}^t \Gamma_{ir}^{(c)}\right)$$
$$\text{s.t. } \Gamma_i, \Gamma_{i,r} = \begin{cases} 1 & \text{if } i, r \notin \mathcal{P} \\ \mathcal{F}(\alpha \nabla_i^{\mathfrak{P}}) & \text{if } i, r \in \mathcal{P} \end{cases} \tag{3}$$

where $i \in \mathcal{P}$ and $\Gamma_i^{(W)}$ and $\Gamma_{ir}^{(c)}$ are advice-based soft gates with respect to a hidden node and its context respectively. $\mathcal{F}()$ is some gating function, $\nabla_i^{\mathfrak{P}}$ is the *"advice gradient"* and α is the trade-off parameter explained later. The key aspect of soft gates is that they attempt to enhance or decrease the contribution of particular edges in the column network aligned with the direction of the *"advice gradient"*. We choose the gating function $\mathcal{F}()$ as an exponential $[\mathcal{F}(\alpha\nabla_i^{\mathfrak{P}}) = \exp(\alpha\nabla_i^{\mathfrak{P}})]$. The intuition is that soft gates are natural, as they are multiplicative and a positive gradient will result in $\exp(\alpha\nabla_i^{\mathfrak{P}}) > 1$ increasing the value/contribution of the respective term, while a negative gradient results in $\exp(\alpha\nabla_i^{\mathfrak{P}}) < 1$ pushing them down. We now present the *"advice gradient"* (the gradient with respect to preferred labels).

Proposition 1. *Under the assumption that the loss function with respect to advice / preferred labels is a log-likelihood, of the form* $\mathcal{L}^{\mathfrak{P}} = \log P(y_i^{(\mathfrak{P})}|h_i^T)$, *then the advice gradient is,* $\nabla_i^{\mathfrak{P}} = I(y_i^{(\mathfrak{P})}) - P(y_i)$, *where* $y_i^{(\mathfrak{P})}$ *is the preferred label of entity and* $i \in \mathcal{P}$ *and* I *is an indicator function over the preferred label. For binary classification, the indicator is inconsequential but for multi-class scenarios it is essential (*$I = 1$ *for preferred label* ℓ *and* $I = 0$ *for* $L \setminus \ell$*).*

Since an entity can satisfy multiple advice rules we take the *most* preferred label, *i.e.*, we take the label $y_i^{(\mathcal{P})} = \ell$ to the preferred label if ℓ is given by most of the advice rules that e_j satisfies. In case of conflicting advice (i.e. different labels are equally advised), we simply set the advice label to be the label given by the data, $y_i^{(\mathfrak{P})} = y_i$ (Proof in supplementary appendix). As illustrated in the K-CLN architecture (Figure 2), at the end of every epoch of training the *advice gradients* are computed and soft gates are used to augment the value of the hidden units (Equation 3).

Proof for Proposition 1 Most advice based learning methods formulate the effect of advice as a constraint on the parameters or a regularization term on the loss function. We consider a regularization term based on the advice loss $\mathcal{L}^{(\mathfrak{P})} = \log P(y_i = y_i^{(\mathfrak{P})}|h_i^T)$ and we know that $P(y_i|h_i^T) = \text{softmax}(b_\ell + W_\ell h_i^T)$. We consider $b_\ell + W_\ell h_i^T = \Psi_{(y_i, h_i^T)}$ in its functional form following prior non-parametric boosting approaches (Odom et al., 2015). Thus $P(y_i = y_i^{(\mathfrak{P})}|h_i^T) = \exp(\Psi_{(y_i^{(\mathfrak{P})}, h_i^T)})/\sum_{y'\in L}\exp(\Psi_{(y', h_i^T)})$. A functional gradient w.r.t. Ψ of $\mathcal{L}^{(\mathfrak{P})}$ yields,

$$\nabla_i^{\mathfrak{P}} = \frac{\partial \log P(y_i = y_i^{(\mathfrak{P})}|h_i^T)}{\partial \Psi_{(y_i^{(\mathfrak{P})}, h_i^T)}} = I(y_i^{(\mathfrak{P})}) - P(y_i) \tag{4}$$

Alternatively, assuming a squared loss such as $(y_i^{(\mathfrak{P})} - P(y_i))^2$, would result in an advice gradient of the form $2(y_i^{(\mathfrak{P})} - P(y_i))(1 - P(y_i))P(y_i)$. We observe that in a functional form the advice gradient is the difference between *the true label distribution and the predicted distribution* (or some function of that difference), *irrespective of the the type of loss* we choose to optimize. As illustrated in the K-CLN architecture (in main paper), at the end of every epoch of training the *advice gradients* are computed and soft gates are used to augment the value of the hidden units as shown in the main section,

$$\Gamma_i, \Gamma_{i,r} = \begin{cases} 1 & \text{if } i, r \notin \mathcal{P} \\ \mathcal{F}(\alpha \nabla_i^{\mathfrak{P}}) & \text{if } i, r \in \mathcal{P} \end{cases}$$

Proposition 2. *Given that the loss function* $\mathcal{H}_i$ *of original CLN is cross-entropy (binary or sparse-categorical for the binary and multi-class prediction cases respectively) and the objective w.r.t. advice is log-likelihood, the functional gradient of the modified objective is,*

$$\nabla(\mathcal{H}_i') = (1-\alpha)\left(y_i I - P(y_i|h^T)\right) + \alpha\left(I_i^{\mathfrak{P}} - P(y_i^{\mathfrak{P}}|h^T)\right)$$
$$= (1-\alpha)\nabla_i + \alpha\nabla_i^{\mathfrak{P}} \tag{5}$$

where $0 \le \alpha \le 1$ *is the trade-off parameter between the effect of data and effect of advice,* I_i *and* $I_i^{\mathfrak{P}}$ *are the indicator functions on the label w.r.t. the data and the advice respectively and* ∇_i *and* $\nabla_i^{\mathfrak{P}}$ *are the gradients, similarly, w.r.t. data and advice respectively.*

Proof for Proposition 2: The original objective function (*w.r.t.* data) of CLNs is cross-entropy. For clarity, let us consider the binary

prediction case, where the objective function is now a binary cross-entropy of the form, $\mathcal{H} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(P(y_i)) + (1-y_i)\log(1-P(y_i))$.

Ignoring the summation for brevity, for every entity i, $\mathcal{H}_i = y_i \log(P(y_i)) + (1-y_i)log(1-P(y_i))$. Extension to the multi-label prediction case with a sparse categorical cross-entropy is straightforward and is an algebraic manipulation task. Now, from Proposition 1, the loss function *w.r.t.* advice is the log likelihood of the form, $\mathcal{L}^{\mathfrak{P}} = \log P(y_i^{\mathfrak{P}}|h^T)$. Thus the modified objective is,

$$\begin{aligned}\mathcal{H}_i' =&(1-\alpha)\left[y_i \log\left(P(y_i)\right) + (1-y_i)\log\left(1-P(y_i)\right)\right] \\ &+ \alpha \log(P(y_i^{\mathfrak{P}}))\end{aligned} \tag{6}$$

where α is the trade-off parameter. $P(y) = P(y|h^T)$ can be implicitly understood. Now we know from Proposition 1 that the distributions, $P(y_i)$ and $P(y_i^{\mathfrak{P}})$, can be expressed in their functional forms, given that the activation function of the output layer is a *softmax*, as $P(y_i) = \exp\left(\Psi_{(y_i,h_i^T)}\right)/\sum_{y'\in L}\exp\left(\Psi_{(y',h_i^T)}\right)$. Taking the functional (partial) gradients (*w.r.t.* $\Psi_{(y_i,h_i^T)}$ and $\Psi_{(y_i^{\mathfrak{P}},h_i^T)}$) of the modified objective function (Equation 6), followed by some algebraic manipulation we get,

$$\begin{aligned}\nabla(\mathcal{H}_i') =&(1-\alpha)[y_i I_i - y_i P(y_i) - P(x_i) + y_i P(y_i)] + \alpha(I_i^{\mathfrak{P}} - P(y_i^{\mathfrak{P}})) \\ =&(1-\alpha)\left(y_i I - P(y_i)\right) + \alpha\left(I_i^{\mathfrak{P}} - P(y_i^{\mathfrak{P}})\right)\end{aligned}$$

Hence, it follows from Proposition 2 that the data and the advice balances the training of the K-CLN network parameters $\theta^{\mathfrak{P}}$ via the trade-off hyperparameter α. When data is noisy (or sparse with negligible examples for a region of the parameter space) the advice/knowledge (if correct) induces a bias on the output distribution towards the correct label. Even if the advice is incorrect, the network still tries to learn the correct distribution to some extent from the data (if not noisy). The contribution of the effect of data vs effect of advice will primarily depend on α. If both data and human advice are sub-optimal, correct label distribution is not learnable.

3.5 The Algorithm

Algorithm 1 outlines all the key steps. KCLN(), the main procedure [**lines: 1-14**], trains a Column Network using both the data (the knowledge graph $\mathcal{G}$) and the human advice (set of preference rules $\mathfrak{P}$). It returns a K-CLN $C^{\mathfrak{P}}$ where $\theta^{\mathfrak{P}}$ are the network parameters, which are initialized to any arbitrary value (0 in our case; [**line: 3**]). Our gating functions are piece-wise/non-smooth and apply only to the subspace entities, features and relations that satisfy the preference rules. So, as a pre-processing step, we create tensor masks that compactly encode such a subspace via the procedure CREATEMASK() [**line: 4**].

The network $C^{\mathfrak{P}}(\theta^{\mathfrak{P}})$ is then trained through multiple epochs till convergence [**lines: 6-12**]. At the end of every epoch the output probabilities and the gradients are computed and stored in a shared data structure [**line: 11**] to be accessed in the next epoch. Training is largely similar to original CLN with *two key modifications* [**line: 9**] - **(1)** Equation 3 is the modified expression for hidden units. **(2)** The data trade-off $1-\alpha$ augments the original loss and the advice trade-off α, is used to compute the gates. Procedure CREATEMASK()

Algorithm 1 K-CLN: **K**nowledge-augmented **Co**L**l**umn **N**etworks

1: **procedure** KCLN(Knowledge graph $\mathcal{G}$, Column network $C(\theta)$, Advice $\mathfrak{P}$, Trade-off α)
2: K-CLN $C^{\mathfrak{P}}(\theta^{\mathfrak{P}}) \leftarrow C(\theta)$ ▷ modified hidden units Eqn 3
3: Initialize $\theta^{\mathfrak{P}} \leftarrow \{0\}$ ▷ initialize parameters of K-CLN
4: $\mathcal{M}^{\mathcal{P}} = \langle \mathcal{M}^W, \mathcal{M}^c, \mathcal{M}^{label}\rangle \leftarrow$ CREATEMASK($\mathcal{G}, \mathfrak{P}$) ▷ mask ∀ entities/relations/labels ∈ $\mathcal{P}$
5: Initial gradients $\forall i\ \nabla_{i,0}^{\mathfrak{P}} = 0$; $i \in \mathcal{P}$
6: **for** epochs k=1 to convergence **do** ▷ convergence criteria same as original CLN
7: Get advice gradients $\nabla_{i,(k-1)}^{\mathfrak{P}}$ for prev. epoch $k-1$
8: Gates $\Gamma_i^{\mathfrak{P}}, \Gamma_{i,r}^{\mathfrak{P}} \leftarrow \exp\left(\alpha\nabla_i^{\mathfrak{P}} \times \mathcal{M}_i^{\mathcal{P}}\right)$
9: Train $C^{\mathfrak{P}}$ using Equation 3; Update $\theta^{\mathfrak{P}}$
10: Compute $\forall i\ P(y_i)$ from $C^{\mathfrak{P}}$ ▷ for current epoch k
11: Store $\forall i\ \nabla_{i,k}^{\mathfrak{P}} \leftarrow I(y_i^{(\mathfrak{P})}) - P(y_i)$ ▷ Obtain $I(y_i^{(\mathfrak{P})}) \Leftarrow \mathcal{M}^{label}$
12: **end for**
13: **return** K-CLN $C^{\mathfrak{P}}$
14: **end procedure**

15: **procedure** CREATEMASK(Knowledge graph $\mathcal{G}$,Advice $\mathfrak{P}$)
16: $\mathcal{M}^W[D \times |O|] \leftarrow \emptyset$ ▷ D: feature length; $|O|$: # entities where $\mathcal{G} = (O, R)$
17: $\mathcal{M}^c[|O| \times |O|] \leftarrow \emptyset$; $\mathcal{M}^{label}[|O| \times L] \leftarrow \emptyset$ ▷ $\mathcal{M}^W$: entity; $\mathcal{M}^c$: context & $\mathcal{M}^{label}$: label mask
18: **for** each preference $p \in \mathfrak{P}$ **do**
19: **if** $\forall i \in O \wedge \forall r \in R : i$ and r satisfies p **then**
20: $\mathcal{M}^W[x, i] \leftarrow 1$ ▷ x is the feature affected by p
21: $\mathcal{M}^c[i, j] \leftarrow 1$ ▷ $r = \langle i, j\rangle \in R; j \neq i; j \in O$
22: $\mathcal{M}^{label}[i, \ell] \leftarrow 1$; s.t. LABELOF($i|p$) = ℓ
23: **end if**
24: **end for**
25: **return** $\langle \mathcal{M}^W, \mathcal{M}^c, \mathcal{M}^{label}\rangle$
26: **end procedure**

[**lines: 15-27**] constructs the tensor mask(s) over the space of entities, features and relations/contexts that are required to compute the gates (as seen in **line: 8**). There are *3 key components* of the advice mask. They are - **(1)** Entity mask $\mathcal{M}^W$ (#entities × #features), indicates entities and relevant features are affected by the advice, **(2)** Context mask $\mathcal{M}^c$ (#entities × #entities), indicates the contexts that are affected (relations are directed, so it is asymmetric), **(3)** Label mask $\mathcal{M}^{label}$, indicates the preferred label of the affected entities, in a one-hot encoding. The masks are iteratively computed for every preference [**lines: 19-25**]. This includes satisfiability checking, $p \in \mathfrak{P}$ [**line: 20**], which is achieved via subgraph matching on the knowledge graph $\mathcal{G}$ (preference rule ≡ subgraph template) ((Das et al., 2019, 2016)). The components $\mathcal{M}^W$ and $\mathcal{M}^c$ are used in gate computation in main procedure and $\mathcal{M}^{label}$ is used for the indicator $I_i^{\mathfrak{P}}$ in the advice gradient.

4 EXPERIMENTS

We investigate the following questions via our evaluation,

(1) Can K-CLNs learn efficiently with noisy sparse samples i.e., performance?
(2) Can K-CLNs learn effectively with noisy sparse samples i.e., speed of learning?

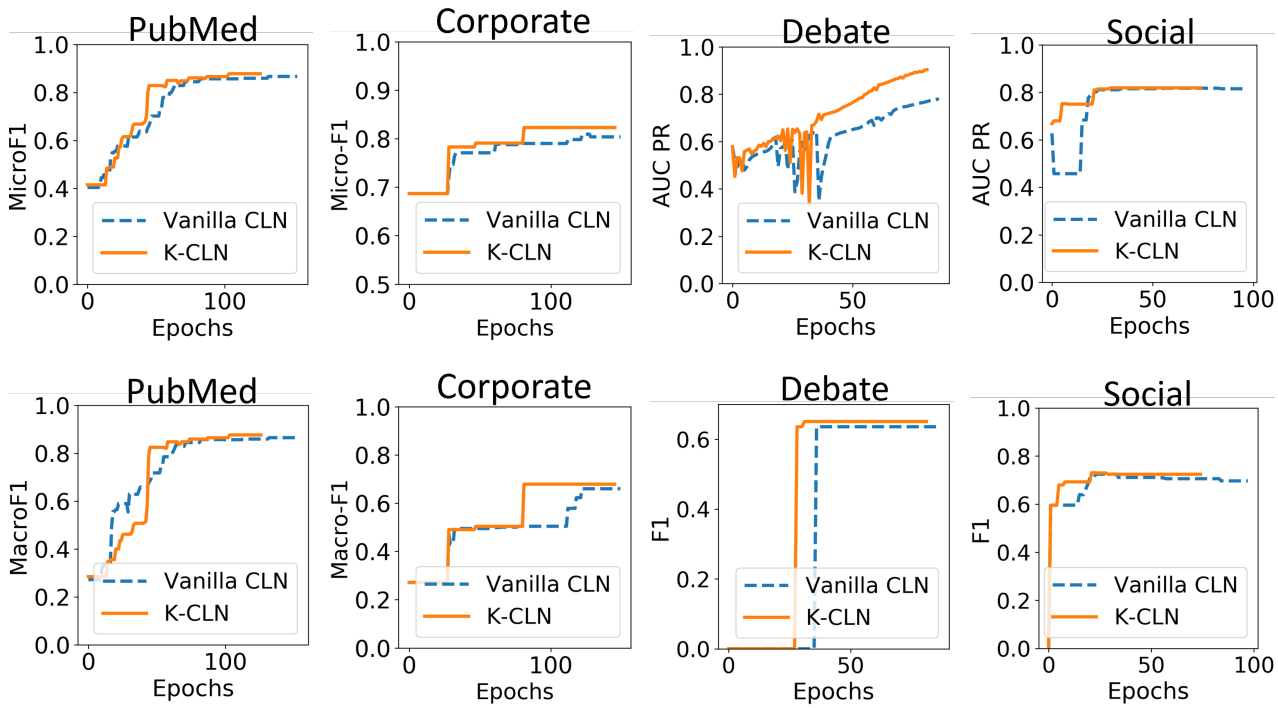

Figure 3: Performance w.r.t. epochs. Macro/Micro-F1 for multi-class problems and F1/AUC for binary class.

(3) How does quality of advice affect the performance of K-CLN i.e., reliance on robust advice?

We compare against the original Column Networks architecture with no advice *(Vanilla CLN indicates the original Column Network architecture (Pham et al., 2017))* as a baseline. Our intention is to show how advice/knowledge can guide model learning towards better predictive performance and efficiency, in the context of collective classification using Column Networks. Also, our problem setting is distinct from most existing noise robust deep learning approaches. Thus, we restrict our comparisons to the original work.

System: K-CLN extends original CLN architecture, which uses *Keras* as the functional deep learning API with a *Theano* backend for tensor manipulation. We extend this system to include: (1) advice gradient feedback at the end of every epoch, (2) modified hidden layer computations and (3) a pre-processing wrapper to parse the advice/preference rules and create appropriate tensor masks. Since it is not straightforward to access final layer output probabilities from inside any hidden layer using keras, we use *Callbacks* to write/update the predicted probabilities to a shared data structure at the end of every *epoch*. This data structure is then fed via inputs to the hidden layers. Each mini-column with respect to an entity is a dense network of 10 hidden layers with 40 hidden nodes in each layer (similar to the most effective settings outlined in (Pham et al., 2017)). The *advice masks* encode $\mathcal{P}$, *i.e.*, the set of entities and contexts where the gates are applicable (Algorithm 1).

Domains: We evaluate our approach on **four relational** domains – *Pubmed Diabetes* and *Corporate Messages*, (multi-class), and *Internet Social Debates* and *Social Network Disaster Relevance* (binary). *Pubmed Diabetes*[2] is a citation network for predicting whether a peer-reviewed article is about *Diabetes Type 1, Type 2 or none*, using textual features (TF-IDF vectors) from 19717 pubmed abstracts and 44, 338 citation relationships among them. Here articles are entities with 500 bag-of-words features (TF-IDF word vectors). *Internet Social Debates*[3] is for predicting stance ('for'/'against') about a debate topic from online posts on social debates. It contains 6662 posts (entities) characterized by TF-IDF vectors and ~ 25000 relations of 2 types, 'sameAuthor' and 'sameThread'. *Corporate Messages*[4] is an intention prediction data set of 3119 flier messages sent by corporate groups in the finance domain, with 1, 000, 000 *sameSourceGroup* relations. We predict the intention of the message *(Information, Action or Dialogue)*. Finally, *Social Network Disaster Relevance* is a relevance prediction data set of 8000 *Twitter* posts, curated and annotated by crowd with relevance scores. Besides textual ones, we use score features and 35k relations among tweets

Domains	#Entities	#Relations	#Features	Target type
Pubmed	19717	44, 338	500	Multi-class
Corporate	3119	~ 1, 000, 000	750	Multi-class
Debates	6662	~ 25000	500	Binary
Disaster	8000	35000	504	Binary

Table 1: Evaluation domains and their properties

[2]https://linqs.soe.ucsc.edu/data
[3]http://nldslab.soe.ucsc.edu/iac/v2/
[4]https://www.figure-eight.com/data-for-everyone/

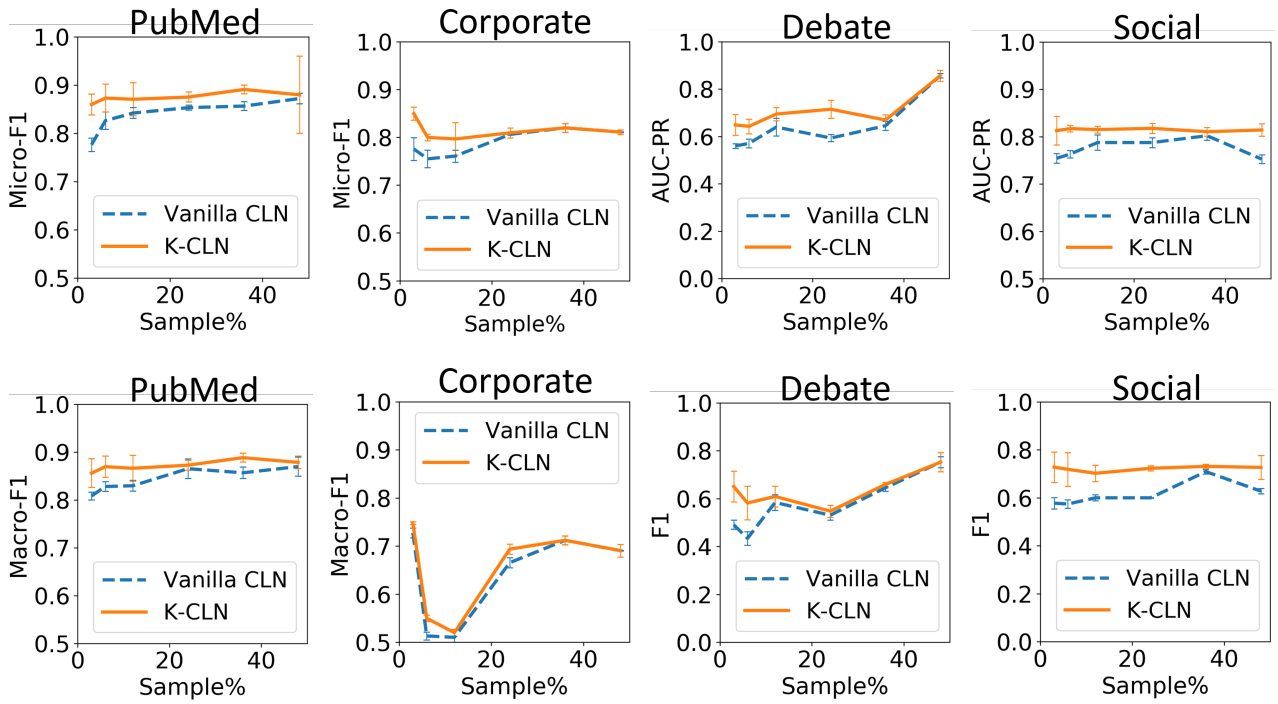

Figure 4: Performance w.r.t. varying samples. Sample size is varied from 3% to 48% of total.

(of types *'same author'* and *'same location'*). Table 1 outlines the important aspects of the 4 domains.

Metrics: Following (Pham et al., 2017), we report macro-F1 and micro-F1 scores for the multi-class problems, and F1 scores and AUC-PR for the binary ones. Macro-F1 computes the F1 score independently for each class and takes the average whereas a micro-F1 aggregates the contributions of all classes to compute the average F1 score. Due to space limitation, we show only the Micro-F1 and the AUC-PR results *(complete set of plots - in supplementary appendix)*. For all experiments we use 10 hidden layers and 40 hidden units per column in each layer. All results are averaged over 5 runs.

Human Advice: K-CLN is designed to handle arbitrarily complex expert advice encoded as preference rules. However, even with some relatively simple preference rules K-CLN is more effective in sparse samples. *Eg: In Pubmed, the most complex preference rule is,* $\texttt{HasWord}(e_1, \text{'fat'}) \wedge \texttt{HasWord}(e_1, \text{'obese'}) \wedge \texttt{Cites}(e_2, e_1) \Rightarrow [\texttt{label}(e_2, \texttt{type}_2) \uparrow]$. Note how a simple rule, indicating an article citing another one discussing obesity is likely to be about Type2 diabetes, proved to be effective. Knowledge from real physicians can thus, be extremely effective. In *Disaster Relevance* even advice rules without domain expertise, such as *a tweet is likely to be NOT about disaster, if posted by the same user who usually posts non-disaster tweets.* Advice rules for experiments in other domains are designed in a similar fashion as well, *i.e.* via simple inspection and understanding of the domains. Another notable aspect is that, sub-optimal advice may lead to a wrong direction of the *Advice Gradient*. However, our soft gates do not alter the loss, but instead are akin to attention mechanisms that promote/demote the contribution of nodes/contexts. The trade-off parameter α balances the effect of advice and data during training.

4.1 Results

Efficiency (Q1): We present the aforementioned metrics with **varying sample size** and with **increasing epochs** and compare our model against *Vanilla CLN*. We split the data sets into a training set and a hold-out test set with 60%-40% ratio. For varying epochs we only learn on 40% of our pre-split training set (*i.e.*, 24% of the complete data) to train the model and test on the hold-out test set. Figure 3 shows that, although both K-CLN and Vanilla CLN converge to the same predictive performance (Micro-F1 for PubMed & Corporate and AUC-PR for the rest), K-CLN converges **significantly faster** (less epochs). Also. for the *corporate* and the *debate*, K-CLN not only **converges faster but also has a better predictive performance** than Vanilla CLN (Figure 3). We have similar observations in Macro-F1 results (see appendix) These results show that K-CLNs learn more *efficiently* with noisy sparse samples thereby answering **(Q1)** affirmatively.

Effectiveness (Q2): The intuition is, *domain knowledge should guide the model to learn better when the training data is sparse.* Thus they are trained on gradually varying sample sizes from 5% of the training data (3% of the complete data) till 80% of the training data (48% of complete data) and tested on the hold-out test set. Figure 4 presents the performance results with varying sample sizes for all data sets. K-CLN outperforms Vanilla CLN across all sample sizes, on both metrics, which suggests that the advice is relevant throughout the training phase with varying sample sizes.

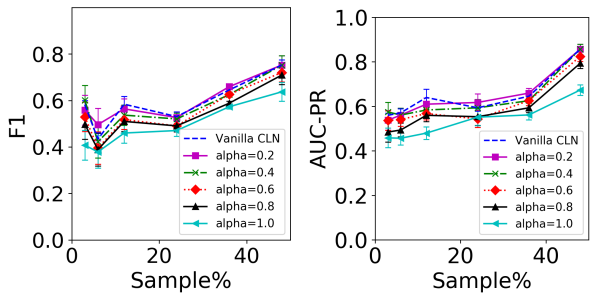

Figure 5: Bad+*alpha*: Performance, F1 *(Left)* and AUC-PR *(Right)* on Debates w/ varying sample sizes & w/ varying *trade-off parameter* α (on advice grad). Note, advice here is incorrect/sub-optimal. $\alpha = 0$ has the identical performance Vanilla CLN.

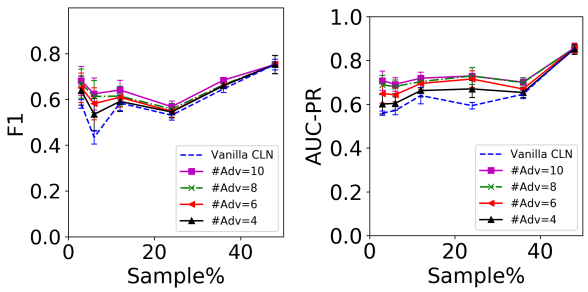

Figure 6: Good+#Advice: Performance, F1 *(Left)* and AUC-PR *(Right)* on Debates w/ varying sample sizes & w/ varying *Number of Advice Rules*. Advice here is good/optimal. $\#Adv = 0$ same as vanilla CLN. $\#Adv = 6$ coincides with our results of KCLN (Fig 3 & 4)

For *Corporate Messages*, K-CLN outperforms with small number of samples, gradually converging to a similar prediction performance with larger samples. However, we have observed that, for Macro-F1, the performance is similar for both, although K-CLN performs better with very small samples. This could happen, in multi-class prediction, when the advice may not apply to some of the classes, while applying to the rest, effectively averaging out in Macro-F1. Thus K-CLN outperforms CLN, with noisy sparse samples **(Q2)**.

Robustness (Q3): An obvious question is – *how robust is our learning system to that of noisy/incorrect advice?* Conversely, *how does the choice of α affect the quality of the learned model?* To answer these, we consider the **Internet Social Debates** domain by augmenting the learner with incorrect advice. The incorrect advice is created by changing the preferred label of the advice rules to incorrect values (as per our understating). Also, recall that the contribution of advice is dependent on the trade-off parameter α, which controls the robustness of K-CLN to advice quality. Hence, we experimented with different values of α $(0.2, 0.4, \ldots, 1.0)$, across varying sample sizes. Figure 5 shows how with higher α values the performance deteriorates due to the effect of noisy advice. $\alpha = 0$ is equivalent to no-advice/Vanilla CLN. Note, with reasonably low values of $\alpha = 0.2, 0.4$, the performance does not deteriorate. Thus with reasonably low values of α K-CLN is robust to quality of advice **(Q3)**. We picked one domain to present robustness but observed similar behavior in all domains.

5 DISCUSSION

It is difficult to quantify correctness or quality of human advice unless, absolute ground truth is accessible in some manner. We evaluate on sparse samples of real data sets with no availability of gold standard labels. We have provided the most relevant/useful advice in the experiments aimed at answering (Q1) and (Q2) as indicated in the experimental setup. We emulate noisy advice (for Q3) by flipping/altering the preferred labels of the original advice rules. We have shown theoretically (Prop. 1 & 2) that the robustness of K-CLN depends on the advice trade-off parameter α that controls the contribution of the data versus the advice towards effective training. We postulate that even in presence of noisy advice, the data (if not noisy) is expected to contribute towards effective learning with a weight of $(1 - \alpha)$. Of course, if both the data and advice are noisy the concept is not learnable (as with any algorithm).

The experiments *w.r.t.* **Q3** (Figure 5) empirically support our theoretical analysis. We found that when $\alpha \leq 0.5$, K-CLN performs well even with noisy advice. In the earlier experiments where we use potentially good advice, we report the results with $\alpha = 1$, since the advice gradient is piecewise (affects only a subset of entities/relations). So it is reasonable to assign higher weight to the advice and the contribution of the entities and relations/contexts affected by it, given the advice is noise-free. Also, note that the drop in performance towards very low sample sizes (in Figure 5) highlights how learning is challenging in the noisy-data and noisy-advice scenario and aligns with our general understanding of most human-in-the-loop/advice-based approaches in AI. Trade-off between data and advice via a weighted combination of both is a well studied solution (Odom and Natarajan, 2018) and, hence, we adapt the same. Again, Figure 6 illustrates the effect of amount advice (on Debates) when advice is optimal. Earlier experiments in Fig 3 & 4 where done with 6 advice rules. Note. although increasing # rules beyond 6 helps in low sample sizes, the difference is not significant since the most important rules were given first.

6 CONCLUSION

We considered the problem of providing guidance for CLNs. Specifically, we assume that the advice givers as true domain experts and not CLN experts and we developed a formulation based on *preferences*. This formulation allowed for natural specification of guidance. We derived the gradients based on advice and outlined the integration with the original CLN formulation. Our experimental results across different domains clearly demonstrate the effectiveness and efficiency of the approach, specifically in knowledge-rich, data-scarce problems. Tracking the expertise of humans to infer advice quality as well as exploring other types of advice including feature importance, qualitative constraints, privileged information, etc. are potentially interesting future direction. Scaling our approach to web-scale data is a natural extension. Finally, extending the idea to other deep models remains an interesting direction for future research.

ACKNOWLEDGMENTS

MD, GK & SN gratefully acknowledge the support of CwC Program Contract W911NF-15- 1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). SN also acknowledges the NSF grant IIS-1836565 and AFOSR award FA9550-18-1-0462. DSD acknowledges the National Institute of Health (NIH) grant no. R01 GM097628. Any opinions, findings and conclusion or recommendations are those of the authors and do not necessarily reflect the view of the DARPA, ARO, AFOSR or the US government.

REFERENCES

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. 2016. Interaction networks for learning about objects, relations and physics. In *NIPS*.

Darius Braziunas and Craig Boutilier. 2006. Preference elicitation and generalized additive utility. In *AAAI*.

Mayukh Das, Devendra Singh Dhami, Gautam Kunapuli, Kristian Rao Kersting, and Sriraam Natarajan. 2019. Fast Relational Probabilistic Inference and Learning: Approximate Counting via Hypergraphs. In *AAAI*.

Mayukh Das, Phillip Odom, Md. Rakibul Islam, Janardhan Rao Doppa, Dan Roth, and Sriraam Natarajan. 2018. Preference-Guided Planning: An Active Elicitation Approach. In *AAMAS*.

Mayukh Das, Yuqing Wu, Tushar Khot, Kristian Kersting, and Sriraam Natarajan. 2016. Scaling Lifted Probabilistic Inference and Learning Via Graph Databases. In *SDM*.

Michelangelo Diligenti, Marco Gori, and Claudio Sacca. 2017. Semantic-based regularization for learning and inference. *AIJ* (2017).

Xintao Ding, Yonglong Luo, Qingde Li, Yongqiang Cheng, Guorong Cai, Robert Munnoch, Dongfei Xue, Qingying Yu, Xiaoyao Zheng, and Bing Wang. 2018. Prior knowledge-based deep learning method for indoor object recognition and application. *Systems Science & Control Engineering* (2018).

M. V. M. França, G. Zaverucha, and A. S. d'Avila Garcez. 2014. Fast relational learning using bottom clause propositionalization with artificial neural networks. *MLJ* (2014).

LiMin Fu. 1995. Introduction to knowledge-based neural networks. *Knowledge-Based Systems* (1995).

Glenn M Fung, Olvi L Mangasarian, and Jude W Shavlik. 2003. Knowledge-based support vector machine classifiers. In *Advances in neural information processing systems*.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

DA Handelman, Stephen H Lane, and Jack J Gelfand. 1990. Integrating neural networks and knowledge-based systems for intelligent robotic control. *IEEE Control Systems Magazine* (1990).

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*.

Navdeep Kaur, Gautam Kunapuli, Tushar Khot, Kristian Kersting, William Cohen, and Sriraam Natarajan. 2017. Relational Restricted Boltzmann Machines: A Probabilistic Logic Learning Approach. In *ILP*.

Seyed Mehran Kazemi and David Poole. 2018. RelNN: A Deep Neural Model for Relational Learning. In *AAAI*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

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

Gautam Kunapuli, Kristin P Bennett, Richard Maclin, and Jude W Shavlik. 2010. The adviceptron: Giving advice to the perceptron. In *ANNIE*.

Gautam Kunapuli, Phillip Odom, Jude W Shavlik, and Sriraam Natarajan. 2013. Guiding autonomous agents to better behaviors through human advice. In *ICDM*.

Quoc V Le, Alex J Smola, and Thomas Gärtner. 2006a. Simpler knowledge-based support vector machines. In *ICML*.

Quoc V Le, Alex J Smola, and Thomas Gärtner. 2006b. Simpler knowledge-based support vector machines. In *Proceedings of the 23rd international conference on machine learning*.

Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*.

Huma Lodhi. 2013. Deep relational machines. In *ICONIP*.

Jaroslav Melesko and Eugenijus Kurilovas. 2018. Semantic Technologies in e-Learning: Learning Analytics and Artificial Neural Networks in Personalised Learning Systems. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*.

Tom M Mitchell. 1980. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey.

Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *TPAMI* (2018).

Michiel O Noordewier, Geoffrey G Towell, and Jude W Shavlik. 1991. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in neural information processing systems*.

P. Odom, T. Khot, R. Porter, and S. Natarajan. 2015. Knowledge-Based Probabilistic Logic Learning. In *AAAI*.

Phillip Odom and Sriraam Natarajan. 2018. Human-Guided Learning for Probabilistic Logic Models. *Frontiers in Robotics and AI, section Computational Intelligence* (2018).

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*.

Trang Pham, Truyen Tran, Dinh Q Phung, and Svetha Venkatesh. 2017. Column Networks for Collective Classification.. In *AAAI*.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *NIPS*.

Tim Rocktäschel, Matko Bošnjak, Sameer Singh, and Sebastian Riedel. 2014. Low-dimensional embeddings of logic. In *ACL 2014 Workshop on Semantic Parsing*.

Jude W Shavlik and Geoffrey G Towell. 1989. Combining explanation-based learning and artificial neural networks. In *Proceedings of the sixth international workshop on Machine learning*. Elsevier.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan R Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*.

Geoffrey G Towell and Jude W Shavlik. 1994. Knowledge-based artificial neural networks. *Artificial intelligence* (1994).

Gustav Šourek, Vojtech Aschenbrenner, Filip Železny, and Ondřej Kuželka. 2015. Lifted Relational Neural Networks. In *NIPS Workshop on Cognitive Comput.: Integr. Neural & Symbolic Approaches*.

Johanna M Zuetenhorst and Babs G Taal. 2005. Metastatic carcinoid tumors: a clinical review. *The Oncologist* (2005).