Deep Neural Networks with Knowledge Instillation

Fan Yang* Ninghao Liu* Mengnan Du* Kaixiong Zhou* Shuiwang Ji* Xia Hu*

Abstract

Deep neural network (DNN) has become an effective computational tool because of its superior performance in practice. However, the generalization of DNN still largely depends on the training data, no matter in quantity or quality. In this paper, we propose a knowledge instillation framework, named NeuKI, for feed-forward DNN, aiming to enhance learning performance with the aid of knowledge. This task is particularly challenging due to the complicated nature of knowledge and numerous variants of DNN architectures. To bridge the gap, we construct a separate knowledge-DNN faithfully encoding the instilled knowledge for joint training. The core idea is to regularize the training of target-DNN with the constructed knowledge-DNN, so that the instilled knowledge can guide the model training. The proposed NeuKI is demonstrated to be applicable to both knowledge rules and constraints, where rules are encoded by structure and constraints are handled by loss. Experiments are conducted on several real-world datasets from different domains, and the results demonstrate the effectiveness of NeuKI in improving learning performance, as well as relevant data efficiency and model interpretability.

1 Introduction

In the past decade, deep neural networks (DNN) have achieved a huge success in various application domains, such as computer vision [1] and natural language processing (NLP) [2]. Following an end-to-end design [3], DNN is typically trained with label information under supervised settings, and seeks to avoid the explicit structure and feature engineering in data. With the current facilities of abundant labelled data and affordable computing resources, DNN somewhat balances well between the data efficiency and learning performance [4].

Meanwhile, it has been well-received that human knowledge has a great potential in facilitating real-world applications [5]. Analogous to the human's decision making, which involves both nurture (e.g., experiences) and nature (e.g., cognition), combining data with domain knowledge can be beneficial in enhancing learning performance. For example, in medical domains, the quantity and quality of the labelled data are usually very limited, because collecting valid samples for one disease could be labor-intensive and time-consuming [6]. To alleviate such problems, researchers usually employ structured approaches, such as graphical models [7], to improve the sample complexity with prior knowledge. This methodology naturally enlightens people to utilize human knowledge for further enhancement of DNN under similar scenarios in deep learning era.

To make use of human knowledge in DNNs, a straightforward way is to build a hybrid system which combines both expert systems and DNN models [8]. With this integration, people can directly inject domain knowledge into the expert system, so as to enhance DNN performance. Beyond this, constructing neural-symbolic system is another methodology to combine knowledge with DNN [9], which generally manipulates architectures from given knowledge to perform reasoning. In addition, researchers recently try to incorporate knowledge by adding logical constraints to DNN [10], aiming to integrate structured knowledge through model regularization.

While initial attempts have been made, effective knowledge incorporation in DNN is still considered to be challenging due to the following reasons. First, most of the human knowledge, such as domain theory, are structured with hierarchy, which makes it nontrivial to be encoded. For example, the distillation method [10], though powerful, has the limitation to reflect the knowledge structure. Second, human knowledge can be diversified with different forms, thus building a general framework for incorporation is hard. Existing method [11], for instance, is designed only for semantic constraints. Third, the target-DNN for integration can have different architectures. With the example of recent method CILP++ [9], it is simply applicable to multilayer perceptron (MLP) and fails for convolutional neural network (CNN), for knowledge incorporation.

To tackle the aforementioned challenges, we propose a general knowledge instillation framework, named **NeuKI**, to instill human knowledge into feed-forward DNN. Particularly, besides the target-DNN, we build another separate knowledge-DNN artificially. The knowledge-DNN is faithfully constructed to regularize the training of target-DNN, so that instilled knowledge

*Department of Computer Science & Engineering, Texas A&M University, College Station, Texas, USA. {nacoyang,nhliu43,dumengnan,zkxiong,sji,xiahu}@tamu.edu.

can affect the model prediction intentionally. NeuKI is demonstrated to be applicable to different forms of knowledge, including both structured rules and declarative constraints. Besides, NeuKI does not have additional requirements on the type of target-DNN as long as it is feed-forward, which makes it convenient to incorporate knowledge into different types of DNN. The main contributions of this paper are summarized as follows:

- We propose a general knowledge instillation framework NeuKI, for feed-forward DNN to incorporate knowledge through regularized model training.
- We design a general architecture for knowledge-DNN, which can incorporate different forms of knowledge, including structured rules and declarative constraints.
- We conduct experiments on several real-world datasets from different domains, demonstrating the effectiveness and superiority of NeuKI.

2 Preliminaries

In this section, we briefly introduce some basics within the context of this paper, and formulate the problem.

2.1 Human Knowledge Within this paper, the concept of knowledge specifically refers to those human knowledge (e.g., domain theory) which can assist people to make better predictions in practice. To enhance human's decision with knowledge, logic plays an important role for reasoning [12]. To this end, most of human knowledge is organized as either structured rules in first-order logic [13] or declarative constraints in propositional logic [14]. The general definitions of knowledge rules and knowledge constraints are given as below.

Definition 1 (Knowledge Rule): It is an explicit regulation formulated from structured human knowledge, which can be generally represented as:

$$\text{Rule } r : \mathcal{A} \longleftarrow \mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m, \quad (m \geq 0), \tag{2.1}$$

where $\mathcal{A}$ is the *head* and $\{\mathcal{A}_1, \cdots, \mathcal{A}_m\}$ is the *body* of r.

Knowledge rules are comprehended from right to left side: if all the body atoms are true, then the head atom is also true. Each atom (e.g., $\mathcal{A}$ and $\mathcal{A}_m$) in rule r is essentially a tuple $p(t_1, t_2, \cdots, t_n)$ $(n \geq 1)$, where p denotes an n-ary predicate and $t_1, t_2, \cdots, t_n$ are the corresponding input terms. When $n = 1$, the predicates in r can be regarded as the attributes of input term. Consider one knowledge rule of animal: $Is\_Carnivore(x) \leftarrow Is\_Mammal(x), Eat\_Meat(x)$, for example. $Is\_Carnivore(x)$ is the head atom, and $Eat\_Meat(x)$, $Is\_Mammal(x)$ are the body atoms, where x denotes the input instance of animal. This knowledge rule just indicates that, if an animal is mammal and eats meat, then this animal is a carnivore.

Definition 2 (Knowledge Constraint): It is generally represented as a declarative statement, which is built by atoms representing basic propositions.

Knowledge constraint is essentially a propositional formula, whose atoms may simply have the value *true* or *false*. Usually, atoms are combined into propositional formulas through various logical operators, such as conjunction ($\wedge$) and disjunction ($\vee$). For example, *"Lion is a mammal and eats meat"* is a typical knowledge constraint, where two atoms (i.e., "is a mammal" and "eats meat") are combined in conjunction with value *true*.

2.2 Feed-Forward DNN Feed-forward DNNs, such as MLP and CNN, are widely used neural models, where neurons are structured by layers and neuron connections only exist between different layers. In feed-forward DNNs, each neuron connection has an associated weight and is established without cycles. Typically, the first layer is *input*, the last layer is *output*, and others are named *hidden* layers. The neurons in the input provide information from outside. The hidden neurons are responsible for model computation. The neurons in the output generate predictions to outside. All information flows inside the model move in only one direction. Considering an input feature vector $\mathbf{x} = [x_1, x_2]$, the output of one hidden neuron h^o can be expressed as:

$$h^o = f(w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2), \tag{2.2}$$

where $f(\cdot)$ indicates the activation function, w_0 is the bias term and $\mathbf{w} = [w_1, w_2]$ denotes the connection weights. Given samples with features $\mathbf{x}$ and target $\mathbf{y}$, feed-forward DNN can learn the relationship between $\mathbf{x}$ and $\mathbf{y}$. The common training method is the back-propagation (BP) algorithm. With proper loss, optimizer and hyper-parameters, feed-forward DNN can work well for both classification and regression tasks.

2.3 Problem Statement We denote the target-DNN for incorporation as $\mathcal{N}^t_{\mathbf{W}}$, where $\mathbf{W}$ indicates the network parameters. With conventional training, $\mathbf{W}$ is iteratively updated to make correct predictions for the labels in training data $\mathcal{D}$. Assuming knowledge $\mathcal{K}$ is available for prediction, we aim to enhance the learning performance of $\mathcal{N}^t_{\mathbf{W}}$ with the aid of $\mathcal{K}$. To achieve this, $\mathcal{K}$ need to be integrated with the training process of $\mathcal{N}^t_{\mathbf{W}}$, so that the information in $\mathcal{K}$ could be effectively injected into $\mathbf{W}$. Overall, compared with the standard training simply with $\mathcal{D}$, our goal is to conduct a knowledge-regularized training for $\mathcal{N}^t_{\mathbf{W}}$ with both $\mathcal{D}$ and $\mathcal{K}$. In this paper, specifically, we refer this joint training process as the knowledge instillation.

3 Knowledge Instillation

$$L(\mathbf{z}^t, \mathbf{z}^k, \mathbf{y}, h) = \lambda_t \underbrace{\sum_{i \in \mathcal{M}} L_t\big(q_i(\mathbf{z}^t, h), y_i\big)}_{\text{loss for target-DNN}} + \lambda_k \underbrace{\sum_{j \in \mathcal{M}'} L_k\left(q_j(\mathbf{z}^k, h), y_j\right)}_{\text{loss for knowledge-DNN}} + \lambda_d \underbrace{\sum_{l \in \mathcal{M} \cap \mathcal{M}'} L_d\left(1 - q_l(\mathbf{z}^t, h),\ 1 - q_l(\mathbf{z}^k, h)\right)}_{\text{loss for knowledge consensus}} \tag{3.3}$$

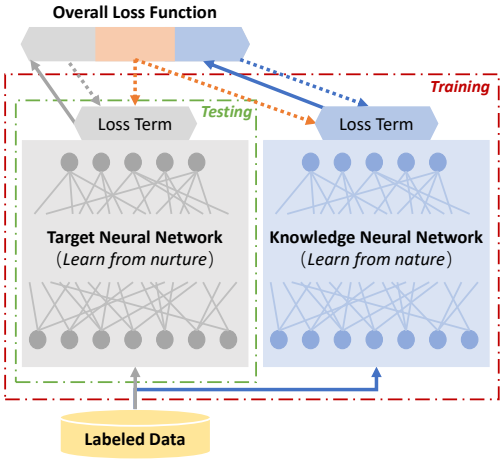


Figure 1: The overall framework of NeuKI.

In this section, we formally introduce NeuKI, illustrated by Fig. 1. Then, we generally design knowledge-DNN, and introduce how to instill rules and constraints.

3.1 Overall Framework Design We design NeuKI to instill knowledge specifically for feed-forward DNNs. To effectively integrate $\mathcal{K}$ with target-DNN $\mathcal{N}^t_{\mathbf{W}}$, we particularly design a separate knowledge-DNN $\mathcal{N}^k_{\mathbf{T}}$ as the model regularizer. $\mathcal{N}^t_{\mathbf{W}}$ and $\mathcal{N}^k_{\mathbf{T}}$ are jointly tuned for instillation during training, while only $\mathcal{N}^t_{\mathbf{W}}$ is kept for prediction during testing. Moreover, $\mathcal{N}^t_{\mathbf{W}}$ and $\mathcal{N}^k_{\mathbf{T}}$ have the same access to $\mathcal{D}$, but they are essentially different in design, where target-DNN is built based on particular task and knowledge-DNN is constructed by specific knowledge. The details of knowledge-DNN construction will be introduced in Sec. 3.2.

For $\mathcal{N}^t_{\mathbf{W}}$, the logit vector [15] can be indicated by $\mathbf{z}^t = [z^t_1, \cdots, z^t_M]$, considering M classes in total. The softmax function with temperature h is defined as:

$$q_i(\mathbf{z}^t, h) = \exp(z^t_i/h) \Big/ \sum\nolimits_{j=1}^{M} \exp(z^t_j/h), \tag{3.4}$$

where h controls the distribution softness. For $\mathcal{N}^k_{\mathbf{T}}$, since knowledge for instillation could be partial, the corresponding logit vector $\mathbf{z}^k$ may only cover part of the classes. We denote the original class set as $\mathcal{M}$ ($|\mathcal{M}| = M$), and the class set covered by knowledge as $\mathcal{M}'$ ($\mathcal{M}' \subseteq \mathcal{M}$). Through similar softmax function $q_i(\mathbf{z}^k, h)$ in Eq. 3.4, the probability distribution over $|\mathcal{M}'|$ classes would be obtained for $\mathcal{N}^k_{\mathbf{T}}$. To effectively regularize $\mathbf{W}$, we let the probability distribution from $\mathcal{N}^k_{\mathbf{T}}$ influence the distribution from $\mathcal{N}^t_{\mathbf{W}}$ accordingly. Thus, in the overall loss, a term on prediction difference between $\mathcal{N}^t_{\mathbf{W}}$ and $\mathcal{N}^k_{\mathbf{T}}$ is designed to achieve the knowledge regularization.

The overall loss of NeuKI can be formulated as Eq. 3.3, where $\lambda_t, \lambda_k, \lambda_d$ indicate the associated term weights, and $\mathbf{y} = [y_1, \cdots, y_M]$ corresponds to the labels in $\mathcal{D}$. The first term in Eq. 3.3 aims to measure the training loss of target-DNN and its specific form depends on the task we handle. For example, in a classification task, L_t could be the cross-entropy expressed as $L_t = -y_i \log(q_i(\mathbf{z}^t, 1)) - (1 - y_i) \log(1 - q_i(\mathbf{z}^t, 1))$. The second term is to measure the training loss of knowledge-DNN, and L_k could be selected as either conventional losses or some special knowledge losses, which will be introduced specifically in Sec. 3.4. The third term is the key for knowledge instillation, where we measure the consensus between target-DNN and knowledge-DNN. The intuition is that the predictions from $\mathcal{N}^t_{\mathbf{W}}$ and $\mathcal{N}^k_{\mathbf{T}}$ need to be as close as possible, since the training samples from real world ought to be matched with valid knowledge. Particularly, L_d is employed as the distance-based loss (e.g., cosine distance), measuring the classification difference with reverse probabilities[1].

As for the training of NeuKI, considering the convergence speed, we apply simultaneous stochastic gradient decent (SSGD) to optimize the framework for joint training, which has been proved to be efficient in [16]. With the overall loss in Eq. 3.3, the objective function for SSGD optimization can be further expressed as:

$$\min_{\mathbf{W}, \mathbf{T}} \quad L(\mathbf{z}^t, \mathbf{z}^k, \mathbf{y}, h) + \mu_t \Phi(\mathbf{W}) + \mu_k \Phi(\mathbf{T}), \tag{3.5}$$

where μ_t, μ_k denote the balance coefficients, and $\Phi(\cdot)$ indicates the L2 regularization. By optimizing Eq. 3.5, $\mathcal{N}^t_{\mathbf{W}}$ and $\mathcal{N}^k_{\mathbf{T}}$ are jointly tuned for knowledge instillation.

3.2 Knowledge Network Structure Constructing a faithful knowledge-DNN regarding to the instilled knowledge is pivotal for the proposed NeuKI. Inspired by neural-symbolic systems [17] and graph networks [18], we design a general knowledge-DNN to map human knowledge. Instead of conducting full connection in MLP or local connection in CNN, we create the

[1] Since $\mathcal{M}' \subseteq \mathcal{M}$, measuring difference from the reverse side can help incorporate classification probabilities from all M classes.

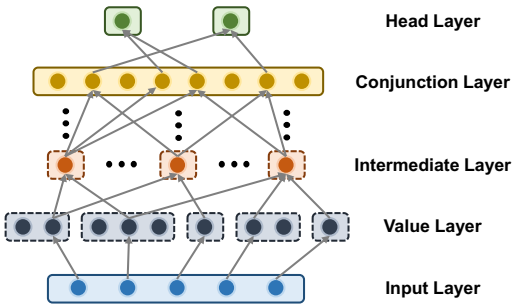


Figure 2: The structure of knowledge-DNN.

neuron links specifically according to the structured relations in knowledge. In this way, the relational inductive bias [18] from knowledge can be effectively encoded into the DNN architecture. The general structure of knowledge-DNN is shown in Fig. 2.

To make effective mappings, we design five different layers for knowledge-DNN. The input layer is designed to handle the predicates in $\mathcal{D}$. The value layer aims to associate input with specific values, since human knowledge could involve numerical conditions beyond binary ones. The intermediate neurons are arranged to mimic the knowledge structure, usually representing some high-level concepts. Through the conjunction layer, different pieces of knowledge are merged together. One conjunction neuron typically indicates an unique knowledge item, corresponding to one decision criterion. Head neurons represent all possible outcomes with knowledge. Note that the number of head neurons may not necessarily equal to the class number in $\mathcal{D}$, because the instilled knowledge could be incomplete.

Knowledge-DNN is basically a special MLP with sparser connections, encoding the relational interactions reflected by knowledge. Knowledge-DNN has better model interpretability compared with conventional DNNs, due to its intuitional network architecture. Essentially, the relational insights from knowledge-DNN are obtained by sacrificing its general computation capability. Although knowledge-DNN may not work well in general cases, it could be beneficial for the specific task with particular knowledge. The loss function of knowledge-DNN is left to be discussed in Sec. 3.4.

3.3 Instilling Knowledge Rules We now introduce how to handle the knowledge rules, indicated by Eq. 2.1. The overall steps include two parts: *rule representation* and *parameter initialization*. Without loss of generality, we consider the body atom as the input attribute, which is the 1-ary predicate for knowledge rules.

Rule Representation. Given a set of knowledge rules, we can map the corresponding relations to the architecture of knowledge-DNN by customizing neuron connections. Fig. 3 shows a simple example of rule representation by knowledge-DNN. Moreover, if the rule set contains different values for the same predicate, a value layer need to be employed to further discriminate. Also, if the rule set is organized hierarchically, intermediate layers need to be introduced for representation as well.

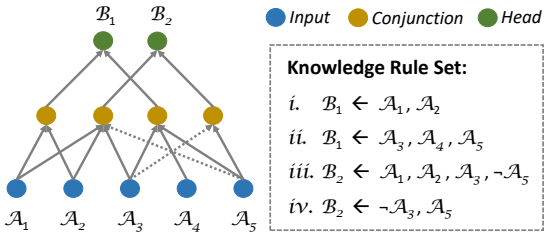


Figure 3: The example of rule representation.

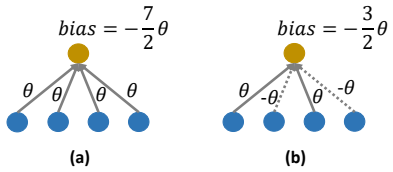


Figure 4: The examples of parameter initialization.

Parameter Initialization. To guarantee the faithfulness, we need to further initialize the parameters of knowledge-DNN, including bias and weight terms. By manual initialization, we aim to guide the knowledge-DNN to behave as rules define. Here, we employ a similar scheme used in [19]. Referring rules, we assign weight θ to each connection with positive attribute input, and assign $-\theta$ to connections with negative inputs. The bias of the aggregate neuron can be initialized based on the activation margin R and positive connection number P, which is indicated by:

$$\text{bias} = -(P - R) \cdot \theta, \quad (0 < R < 1). \tag{3.6}$$

To illustrate the points, we show two examples in Fig. 4, where both of them assume binary attribute input for simplicity. In both examples, R is set to be $\frac{1}{2}$. Besides, for some special knowledge rules whose atoms are arranged in disjunction manner, we can set the corresponding P as 1, since the aggregate neuron would be activated if any of the input attribute is true.

Furthermore, it is remarkable to note that the knowledge-DNN in NeuKI is applicable to different types of knowledge rules, including *deterministic*, *probabilistic* and *fuzzy* rules. For deterministic ones, the weights could be stored in constant tensors simply with value θ, $-\theta$ and 0, which would not be updated by the BP algorithm. Under this scenario, the designed knowledge-DNN is reduced to a decision tree with certainty. For probabilistic and fuzzy rules, we could use variable tensors to store the weights, and further tune the weights through BP process. The normalized weights in knowledge-DNN can indicate the conditional firing probability as well as fuzzy truth value.

3.4 Instilling Knowledge Constraints Declarative constraint is another important carrier for knowledge. By employing special losses, knowledge-DNN is also capable of incorporating constraints. With NeuKI, referring Eq. 3.3, we aim to encode constraints through L_k. Specifically, let L_k^α denotes the knowledge loss on α, where α is a declarative statement for constraint. Then, L_k^α measures the satisfaction extent for α, and $L_k^\alpha = 0$ when α is perfectly satisfied. With similar principles, the knowledge loss on complex formulas with conjunction and disjunction, can be further indicated by:

$$L_k^{\alpha\wedge\beta} = L_k^\alpha + L_k^\beta \quad \text{and} \quad L_k^{\alpha\vee\beta} = L_k^\alpha \cdot L_k^\beta, \tag{3.7}$$

where $L_k^{\alpha\wedge\beta} = 0$ holds only when $L_k^\alpha = L_k^\beta = 0$, and $L_k^{\alpha\vee\beta} = 0$ holds for either $L_k^\alpha = 0$ or $L_k^\beta = 0$. As for the loss on formulas with negation, it could be handled by transformation with De Morgan's laws. Recent work [11, 20] have already conducted some initial exploration on the functional form of L_k^α, and effectively encoded some specific semantic constraints. Here, we will not investigate the general form of L_k^α, but instead discuss the differences for constraint instillation between NeuKI and other existing work.

Despite the specific functional form of L_k^α, existing work [11, 20] all treat L_k^α as an additional regularization term for training. Instead of appending L_k^α at the end of loss, NeuKI directly employs L_k^α to train a separate model (i.e., knowledge-DNN) for constraint instillation. There are some nice properties of doing this: (1) The auxiliary training of knowledge-DNN provides another view from the constraint perspective, which potentially improve the network convergence; (2) The knowledge consensus term L_d, referring Eq. 3.3, enables the knowledge reciprocity from different tasks (i.e., label prediction and constraint satisfaction), which is beneficial for performance enhancement. Thus, we note that NeuKI is more advantageous in handling knowledge constraints compared with existing efforts.

4 Experiments

In this section, we evaluate NeuKI on several real-world datasets from different domains. With the experiments, we aim to answer three key research questions:

- How effective is NeuKI in improving learning performance of the target-DNN with human knowledge?
- How does the data efficiency change by knowledge instillation with NeuKI regarding to the target-DNN?
- How is the model interpretability of the target-DNN after instilling specific human knowledge with NeuKI?

4.1 Experimental Settings

Table 1: Dataset statistics in experiments.

Datasets	#Instance	#Class	Domain
ASD	704	2	Clinical Diagnosis
SGS	3,190	3	Life Science
MR	10,662	2	Sentiment Analysis
IMDB	50,000	2	Sentiment Analysis

4.1.1 Datasets We employ four real-world datasets to evaluate NeuKI. The relevant knowledge depends on the specific task, which is formulated from different perspectives. The data statistics are given in Table 1.

- **Autistic Spectrum Disorder (ASD)**: It is a recent dataset on ASD classification for adults, which is used to evaluate NeuKI on knowledge rules;
- **Splice-junction Gene Sequences (SGS)**: It involves the sequence data with imperfect domain theory, which is used to evaluate rule instillation as well.
- **Movie Review (MR)**: It is a benchmark dataset for sentiment classification, which is used to evaluate NeuKI on knowledge constraint instillation.
- **IMDB**[2]: It is another large-scale benchmark dataset for binary sentiment classification, which is also used for evaluation of constraint instillation.

4.1.2 Alternatives and Baselines We employ the following alternatives for knowledge instillation.

- **KBANN** [19]: This is one of the earliest knowledge-based models, which is designed by manipulating MLP architecture based on knowledge rules.
- **Rule-p/q** [10]: This is a recent method to incorporate rules for DNN, which is developed by iterative projections between student (p) and teacher (q).
- **Semantic Loss** [11]: This method is designed to incorporate semantic constraints for DNN, which appends a knowledge loss term for regularization.

Besides, we also include some DNN baselines as follows.

- **MLP** [21]: The basic DNN for prediction. We use it as the baselines for most of the tasks in experiments.
- **CNN-seq** [22]: The CNN designed for sequence classification, which is our baseline for SGS data.
- **CNN-text** [23]: The CNN designed for text classification, which is our baseline for MR and IMDB data.
- **LSTM** [24]: The recurrent model for text classification, which is the baseline for MR and IMDB data.

[2] http://ai.stanford.edu/~amaas/data/sentiment/

4.2 Experimental Results In the following, we present and discuss the experiment results of NeuKI. Sec. 4.2.1 and Sec. 4.2.2 study NeuKI on learning performance with knowledge rules. Then, in Sec. 4.2.3, we focus on the data efficiency with NeuKI. Further, model interpretability aided by NeuKI is discussed in Sec. 4.2.4. Finally, we investigate the NeuKI with knowledge constraints in Sec. 4.2.5. For all the experiments, DNNs are implemented with Keras[3], and we set $h = 1$, $\lambda_t = \lambda_k = 1$, $\mu_t = \mu_k = 10^{-3}$.

4.2.1 Learning with Extracted Rules. In this part, we first evaluate the learning performance of target-DNN with NeuKI on a clinical application. Particularly, we employ the ASD dataset, and split the data where 80% is for training and 20% is for testing. Considering the data type (i.e., categorical) and prediction task (i.e., classification), we employ target-DNN $\mathcal{N}^t_{\mathbf{W}}$ as an MLP model with one hidden layer. L_t, L_k take the binary cross-entropy loss, and L_d is implemented as the Euclidean distance. Besides, we set $\lambda_d = 1.6$ for overall training. Since there is no available knowledge directly applied to ASD dataset, we use the extracted rules as potential probabilistic knowledge for classification. Specifically, we use WEKA 3.8[4] to mine relevant knowledge rules from ASD dataset, and the corresponding rules are listed as below. In the mined rule set, the head atoms indicate the predicted label (i.e., "YES" and "NO"), and the body atoms correspond to the attributes of each patient (e.g., "A1_Score").

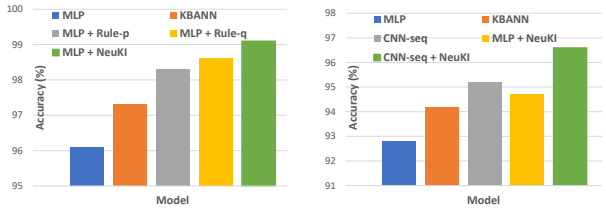
```
Classification Rules for ASD Dataset ---
NO←A9_Score=0,result=4; NO←result=6;
NO←A5_Score=0,A6_Score=0,A9_Score=0;
NO←result=5; NO←A4_Score=0; NO←A7_Score=0,jundice=no;
YES←A4_Score=1,A5_Score=1,A10_Score=1;
YES←A1_Score=1,A3_Score=1; YES←A5_Score=1,A6_Score=1;
```

Relevant experiment results are shown in Fig. 5(a). Among all the experiments, each method is run ten times with different random seeds, and the final results are reported based on the average. From the results, we see that all knowledge-based models can somewhat improve the accuracy with the aid of the extracted knowledge rules. Besides, NeuKI outperforms all other alternatives and the baseline, which directly demonstrates the effectiveness of NeuKI in enhancing learning performance with the aid of knowledge rules.

4.2.2 Learning with Domain Theory. In this part, we evaluate NeuKI with specific domain theory, which is typically a set of rules organized hierarchically.

[3]https://keras.io/

[4]https://www.cs.waikato.ac.nz/ml/weka/downloading.html



(a) ASD Dataset. (b) SGS Dataset.

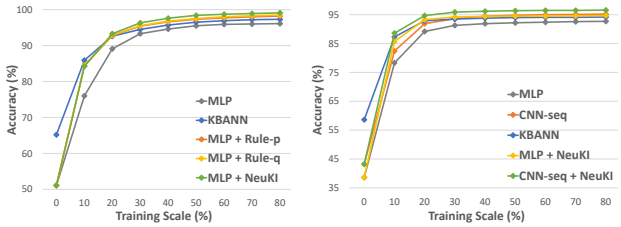
Figure 5: Classification accuracy with knowledge rules.

Specifically, we employ SGS dataset, and split the data for 10-fold cross validation. Since the prediction task is the sequence classification, we employ both MLP and CNN-seq as baselines. Also, since Rule-p/q method is not applicable to hierarchical rules, the alternative we compare with only contains KBANN. In this task, the employed MLP consists of two hidden layers, and the employed CNN-seq contains one convolutional layer. The hyper-parameters for the overall loss is the same as those in the ASD prediction task. As for the knowledge rules of this task, we refer to literature [25] and present the rules as follows. These rules are also considered to be probabilistic ones due to their uncertain nature in life science. In the rule set, the head atoms correspond to high-level concepts (e.g., "pyramidine-rich") or predicted labels (e.g., "EI"), and the body atoms indicate the relative location to reference point (e.g., @3) or the corresponding DNA symbols (e.g., 'TAG'). The standard notations of all possible combinations with nucleotides are introduced in [26].

```
Domain Theory for SGS Dataset ---
EI ← @-3'MAGGTRAGT', ¬EI-stop;
EI-stop←@-3'TAA';EI-stop←@-4'TAA';EI-stop←@-5'TAA';
EI-stop←@-3'TAG';EI-stop←@-4'TAG';EI-stop←@-5'TAG';
EI-stop←@-3'TGA';EI-stop←@-4'TGA';EI-stop←@-5'TGA';
IE ← pyramidine-rich, @-3'YAGG', ¬IE-stop;
pyramidine-rich ← 6 of (@-15'YYYYYYYYYY');
IE-stop←@1'TAA'; IE-stop←@2'TAA'; IE-stop←@3'TAA';
IE-stop←@1'TAG'; IE-stop←@2'TAG'; IE-stop←@3'TAG';
IE-stop←@1'TGA'; IE-stop←@2'TGA'; IE-stop←@3'TGA';
```

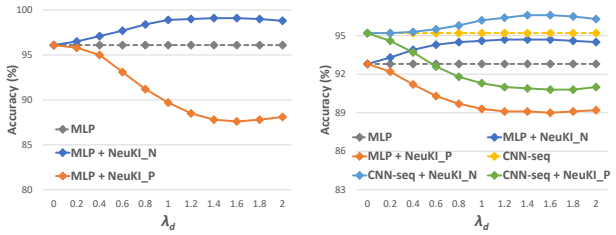
Fig. 5(b) shows the results on SGS dataset. According to the results, we note that CNN-seq performs much better than MLP for the sequence classification task, and it even outperforms the alternative knowledge-based model KBANN. As for NeuKI, it works well on both MLP and CNN-seq architecture, and enables a higher accuracy compared with all other baselines and alternatives accordingly. This set of experiments demonstrate that NeuKI is also capable of instilling domain theory with knowledge rules to further enhance the learning performance of the target-DNN.

4.2.3 Data Efficiency with Rules. In this part, we evaluate the data efficiency of target-DNN with



(a) ASD Dataset. (b) SGS Dataset.

Figure 6: Data efficiency comparison.



(a) ASD Dataset. (b) SGS Dataset.

Figure 7: Model interpretability illustration.

NeuKI in both ASD and SGS dataset. To test the data efficiency, we train each DNN with different data scale regarding to the whole training set, and observe the corresponding accuracy. In particular, under the same scale, the model with higher test accuracy is regarded to be more efficient in training. Fig 6 shows the experiment results for both datasets. For ASD dataset, in Fig 6(a), we note that NeuKI and other alternatives are more efficient than the baseline MLP, especially when the training data is largely limited to 10% or 20%. Compared with alternatives, the data efficiency of NeuKI is competitive, and can slightly outperform them when the training scale is over 20%. Besides, for the scenario where training scale is extremely small (under 10%), we also note that KBANN is the best performer, which may benefit from its specific structure and initialization. In Fig 6(b), we have some similar observations for SGS dataset. With NeuKI, the target-DNN (i.e., MLP or CNN-seq) acquires enhanced data efficiency compared with baselines accordingly, and slightly outperforms the alternative KBANN when the training scale is over 20%. From this set of results, we demonstrate that the data efficiency of the target-DNN can be improved by knowledge instillation with NeuKI.

4.2.4 Model Interpretability with Rules. We investigate the model interpretability of target-DNN with NeuKI in this part. Conventional DNN typically lacks interpretability, because what they learn highly depends on the training data itself. However, with the aid of NeuKI, we can obtain some model insights for target-DNN according to the knowledge we instill. Specifically, to evaluate the model interpretability, we intentionally perturb the instilled knowledge and further observe the classification changes. If the target-DNN could alter its predictions in the perturbation direction, the instilled knowledge is then shown to be effective in reflecting the model mechanism, which can be regarded as a lens to the target-DNN. In particular, we make perturbations on both the extracted rules in ASD dataset and the domain rules in SGS dataset. The perturbation principle is to reverse all the rules whose head atom involves prediction labels. For example, if the head atom of one knowledge rule in ASD dataset is "YES", then we would reverse it to "NO" as our perturbative rule. In SGS dataset, along with this principle, we don't modify those knowledge rules with high-level concepts.

In the experiments, we observe the performance of target-DNN under three different scenarios, i.e., without knowledge rules, with normal rules and with perturbative rules. We use λ_d, referring Eq. 3.3, to control the knowledge regularization strength. The results are shown in Fig. 7, where NeuKI_N and NeuKI_P respectively indicate the instillation with normal and perturbative rules. According to the results in both datasets, we note that instilling perturbative rules is detrimental to the accuracy of target-DNN, which demonstrates the fact that perturbative rules potentially alternate some predictions which are correctly classified without rules. Besides, with the increase of λ_d, NeuKI_N can help improve the accuracy of target-DNN where the best improvements are reached at $\lambda_d = 1.4$ or $\lambda_d = 1.6$ depending on the task. Similarly, within some extent, the larger λ_d will also enable NeuKI_P to depress the learning performance significantly. With this set of experiments, we note that the model interpretability of target-DNN is improved due to the instilled knowledge with NeuKI.

4.2.5 Learning with Knowledge Constraints. In this part, we evaluate the effectiveness of NeuKI on knowledge constraints. To be specific, we employ the *exactly-one* constraint, which is evaluated in [11], for classification tasks. The declarative statement α for the constraint can be stated as: the network output for classification tasks should be mutually exclusive, i.e., exactly one binary indicator must be true. Accordingly, the semantic knowledge loss L_k^α is indicated by [11]:

$$L_k^\alpha = -\log \sum_{i=1}^{M} q_i \prod_{j=1, j\neq i}^{M} (1 - q_j), \tag{4.8}$$

where M is the number of classes, and q_i denotes the probability of class i. Particularly, in our experiments, we focus on the sentiment classification task, and use MR as well as IMDB dataset for evaluation.

The experiment results are shown in Table 2. As for the CNN-text_rand and CNN-text_static, they indicate

Table 2: Accuracy with knowledge constraints.

Model	*Accuracy (%)*	
	MR	IMDB
LSTM	78.8	84.1
CNN-text_rand	76.1	83.4
CNN-text_static	81	86.9
LSTM + Semantic Loss	78.8	84.2
CNN-text_rand + Semantic Loss	76.3	83.9
CNN-text_static + Semantic Loss	81.3	87.6
CNN-text_rand + NeuKI	**76.6**	**84.1**
CNN-text_static + NeuKI	**81.5**	**88.2**

two different modes of the baseline CNN-text, referring the notations in [23]. According to the results, we note that both the semantic loss and NeuKI help improve the accuracy, with the aid of the exactly-one knowledge constraint. Since this constraint is not strong, we can only observe a limited improvement with knowledge instillation. Besides, with the benefits of auxiliary training and knowledge reciprocity, NeuKI obtains slightly higher enhancement accordingly, compared with the alternative. This set of experiments directly demonstrate the effectiveness of NeuKI in instilling knowledge constraints, and show the advantages of NeuKI for constraint instillation over the existing method.

4.3 Discussion With conducted experiments, we give a further discussion about the target-DNN and instilled knowledge. Although some promising results are shown, we still need to point out that human knowledge cannot always guarantee the improvement despite the knowledge correctness. The key lies in the matching extent between knowledge and data. Take one of our initial experiments on the lung cancer dataset (not reported in this paper) for example. Smoking and air pollution have been generally considered as two important reasons for lung cancer. However, after we instill such knowledge, we obtain the worse performance than baselines. A quick overview for the data help us find the answer, which shows that there does exist many patients who never smoke and inhale polluted air. This example just demonstrates the importance of knowledge matching with regards to data. Thus, to better enhance the performance, the instilled knowledge can be partial, but it need to be matched with specific data. This actually explains why we employ WEKA to extract knowledge in Sec. 4.2.1, instead of using some social theory, because we need the exact knowledge for ASD dataset rather than the general knowledge for autistic disorder.

5 Related Work

Combining human knowledge with learning models has been regarded as a significant problem for a long while [27]. So far, various efforts have been paid to enhance the generalization and interpretability of DNN with the aid of knowledge. We categorize those work into two lines, and introduce some representative ones.

The first line of efforts lies in how to utilize the structured rules for DNN. Recently, [10] proposed a distillation framework to transfer the knowledge in rules into the parameters of DNN, which is applicable to diversified DNN architectures. Focusing on image classification tasks, the authors in [28] designed a method to incorporate prior deterministic knowledge rules with semantic based regularization. In addition, the structured rules formulated by relation graphs were studied in [29], with the goal of enhancing accuracy in object classification. Moreover, in [30], the authors employed the local patterns and the rule-modulated map to integrate the structured knowledge, aiming to strengthen conventional DNN with long-term dependencies.

The second line focuses on the knowledge constraint integration with DNN. The authors in [11] designed a general semantic loss function to reflect the constraint satisfaction, which regularizes the learning process of DNN based on human knowledge. In [20], the authors proposed a way to translate knowledge constraints into a differentiable loss term with desirable mathematical properties, and further optimized DNN with constraint regularization. Authors of [31] studied the constrained CNN for image segmentation tasks, where a novel loss function was employed to tune the models with set of linear constraints on outputs. As for work [32], the authors directly imposed hard constraints on DNN with a computationally feasible way, and showed promising results aided by knowledge constraints.

6 Conclusion and Future Work

In this paper, we propose a general knowledge instillation framework, named NeuKI, to incorporate human knowledge for enhancement of feed-froward DNN. In particular, a separate knowledge-DNN is faithfully designed to encode knowledge, which is jointly trained with target-DNN for knowledge regularization. By customizing the architecture as well as the loss of knowledge-DNN, NeuKI can sufficiently handle both structured rules and declarative constraints. Experiments on several real-world datasets demonstrate the improved learning performance with NeuKI, as well as the enhanced data efficiency and model interpretability. Promising future extensions of this work may include instilling more general knowledge and enabling reasoning capability of DNN with the aid of human knowledge.

Acknowledgement This work is, in part, supported by DARPA (N66001-17-2-4031, FA8750-17-2-0116), NSF (IIS-1657196, IIS-1750074) and JP Morgan. The opinions and/or findings shown in this paper do not represent the official views of any funding agency.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[2] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the conference on EMNLP*, pages 1422–1432, 2015.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[4] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[5] Juan Carlos Alvarado-Pérez, Diego H Peluffo-Ordónez, and Roberto Therón. Bridging the gap between human knowledge and machine learning. *The Advances in Distributed Computing and Artificial Intelligence Journal*, 4(1):54–64, 2015.

[6] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 2008.

[7] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[8] Murat Karabatak and M Cevdet Ince. An expert system for detection of breast cancer based on association rules and neural network. *Expert systems with Applications*, 36(2):3465–3469, 2009.

[9] Manoel VM França, Gerson Zaverucha, and Artur S d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning*, 94(1):81–104, 2014.

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

[11] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5502–5511, 2017.

[12] Timothy J Ross. *Fuzzy logic with engineering applications*. John Wiley & Sons, 2005.

[13] Raymond R Smullyan. *First-order logic*, volume 43. Springer Science & Business Media, 2012.

[14] Hans Kleine Büning and Theodor Lettmann. *Propositional logic: deduction and algorithms*, volume 48. Cambridge University Press, 1999.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint:1503.02531*, 2015.

[16] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2018.

[17] Artur S d'Avila Garcez, Krysia B Broda, and Dov M Gabbay. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2012.

[18] Peter W Battaglia, Jessica B Hamrick, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint:1806.01261*, 2018.

[19] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[20] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, et al. Dl2: Training and querying neural networks with logic. 2018.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[22] Ngoc Giang Nguyen, Vu Anh Tran, Duc Luu Ngo, et al. Dna sequence classification by convolutional neural network. *Journal of Biomedical Science and Engineering*, 9(05):280, 2016.

[23] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on EMNLP*, pages 1746–1751, 2014.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[25] Tania A Baker, James D Watson, Stephen P Bell, A Gann, MA Losick, and R Levine. *Molecular biology of the gene*. Benjamin-Cummings, 2003.

[26] IUB Nomenclature Committee et al. Ambiguity codes. *Europ. J. Biochem*, 150:1–5, 1985.

[27] Ashwin Srinivasan, S Muggleton, and RD King. Comparing the use of background knowledge by inductive logic programming systems. In *Proceedings of the 5th Workshop on ILP*, pages 199–230, 1995.

[28] Soumali Roychowdhury, Michelangelo Diligenti, and Marco Gori. Image classification using deep learning and prior knowledge. In *Workshops from AAAI Conference on Artificial Intelligence*, 2018.

[29] Jia Deng, Nan Ding, Yangqing Jia, et al. Large-scale object classification using label relation graphs. In *ECCV*, pages 48–64, 2014.

[30] Hu Wang. Renn: Rule-embedded neural networks. *arXiv preprint:1801.09856*, 2018.

[31] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE ICCV*, pages 1796–1804, 2015.

[32] Pablo Márquez-Neila, Mathieu Salzmann, et al. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint:1706.02025*, 2017.