Training Neural Networks with Classification Rules for Incorporating Domain Knowledge

Wenyu Zhang[a,1], Fayao Liu[a,1], Cuong Manh Nguyen[a], Zhong Liang Ou Yang[a,2], Savitha Ramasamy[a], Chuan-Sheng Foo[a,b]

[a] *Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), 1 Fusionopolis Way, #21-01, Connexis South Tower, Singapore*
[b] *Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), 1 Fusionopolis Way, #16-16, Connexis North Tower, Singapore*

Abstract

Neural networks are capable of learning complex concepts and tasks, given abundant training data. In real-world applications where data collection can be difficult, integrating domain knowledge into the model can reduce the burden on data requirements and allow human experts greater control over model decisions. This paper focuses on incorporating conditional statements for tabular data as classification rules, which have a simple structure and are easy to construct. We introduce a general rule loss constraint to guide neural network training in a model agnostic manner, and propose confidence learning to automatically weigh the contribution of multiple candidate rules. Experimental evaluation with three real-world datasets shows that the rule loss can substantially increase model performance, particularly when training data is limited.

Email addresses: zhang_wenyu@i2r.a-star.edu.sg (Wenyu Zhang), liu_fayao@i2r.a-star.edu.sg (Fayao Liu)

[1] These two authors contributed equally to this work.

[2] Work was done while Z. L. Ou Yang was with Institute for Infocomm Research, A*STAR, Singapore.

Keywords: Rule based systems, Knowledge based systems, Neural networks, Constrained learning

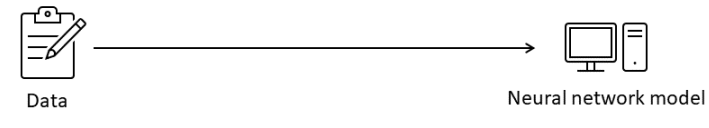
1. Introduction

Neural network models have demonstrated remarkable abilities in a variety of domains and applications. Given abundant training data, neural network models are capable of learning complex concepts and tasks. However, collecting data for training can be challenging, expensive or time-consuming in real-world applications. The ability to incorporate human expert knowledge in the model training process can reduce the burden on data requirements, as well as create a human-in-the-loop process where human experts have greater control over the decisions made by the learned model.

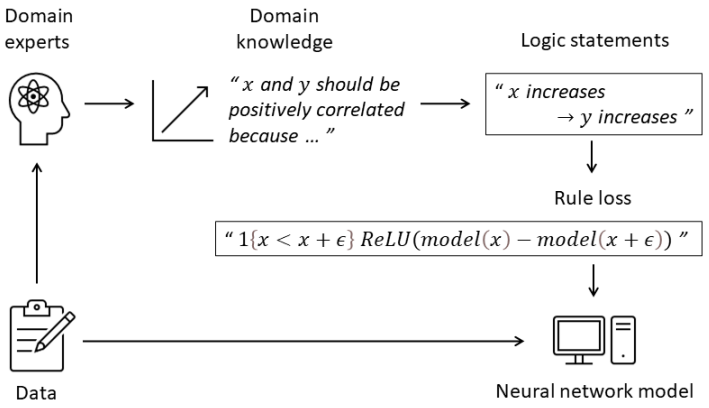
In the review by von Rueden et al. [1], prior domain knowledge can be incorporated through additional information such as algebraic and differential equations, logic rules, knowledge graphs, and simulation results. For example, classical physics knowledge can be incorporated into physics-informed neural networks by penalizing model outputs deviating from the partial differential equations describing the dynamical system [2]. In this paper, we focus on conditional statements for classification tasks with tabular data. We use simple conditional sentences of the form $A \rightarrow B$ ("if A, then B") to explicitly capture the relationship between characteristics of the input and the output class probability. We note that these If-Then statements are represented in formal logic systems such as propositional logic and first-order logic. In this work, we do not delve into the more expressive syntax of formal logic systems that include variables and quantifiers, which may be more diffi-

cult to construct for real-world systems and more complex for model learning. Examples of If-Then sentences in real-world applications include "if the delay in initiating the machine component is greater than threshold τ, then it is faulty", and "if a patient is older, then the patient is at greater risk of heart diseases". Due to the simplicity in the structure of these conditional sentences compared to the other forms of knowledge representation, we expect human domain experts to more readily construct these conditional statements from their prior knowledge and experience of the classification tasks. Rules can also be extracted from trained machine learning models [3, 4]. We assume the rules to be provided, with rule extraction considered outside the scope of this work. Rules with hard thresholding is the foundation of decision trees, which despite their simplicity, are still popular methods for tabular data. However, decision trees have a limited modeling capacity compared to neural networks [5, 6]. Our goal is to incorporate the conditional statements as classification rules to guide neural network training in a model agnostic manner. Figure 1 compares the conventional workflow where data is directly fed into neural network models for learning, versus the workflow in this work, where domain knowledge from human experts are formulated into conditional statements and then incorporated into the model learning objective.

In this work, we constrained the learning of neural network parameters by imposing a rule loss representing each conditional statement. In particular, for a rule $A \rightarrow B$, a penalty is applied for violating the consequent B when the antecedent A is true. We introduce a general form of the rule loss which encapsulates more specific forms from previous studies [7, 8]. While other



(a) Conventional



(b) Injecting If-Then rules

Figure 1: Process of training neural network models

studies have allowed only a single rule [7, 8, 9, 10], which can be impractical and requires manual rule selection, we propose to use confidence learning to automatically weigh the contribution of multiple candidate rules. We demonstrate in Section 4 the advantage of incorporating rules with three real-world datasets, and achieved performance gains especially when training data is limited. The results from this work reflect the potential of leveraging simple forms of prior domain knowledge to address data shortage issues in training neural networks.

2. Related Work

2.1. Integrating domain knowledge

Previous studies have incorporated human expert knowledge through different forms of knowledge representation such as algebraic and differential equations to model physical systems [2, 11], logic rules to reflect formal logic dependencies between variables [7, 8], and knowledge graphs typically constructed from databases to describe the relationships and structures of entities [12]. Abu-Mostafa [13] expressed knowledge as input-output examples to guide model training. Due to the rich semantic and logic structures in languages, natural language processing (NLP) is a popular domain for demonstrating the power of incorporating knowledge. Gan et al. [14] proposed to inject legal knowledge as a set of first-order logic rules into neural network models to enhance the logical reasoning capability for judgment prediction, with Yu et al. [15] and Wang & Pan [16] incorporating rules on the relationships between entities in information extraction. We refer readers to reviews [1, 17] for details on different types of knowledge representations and how they are integrated into model learning.

2.2. Encoding logical constraints

Recent studies in neural-symbolic AI have aimed to explicitly integrate complex logical constraints with machine learning methods. Some methods use neural circuits and logic programming languages for symbolic processing [18, 19]. These can be applied to non-differentiable discrete constraints but are computationally intensive. As an example, Yang et al. [20] estimated the gradients of discrete functions and enabled learning with discrete constraints

using gradient descent. There are also studies that have algorithmically translated a set of rules to generate a neural network for training [3, 21, 22], with that of de Penning et al. further growing and adapting the network in an online learning setting [4]. Neurons have been constructed to represent elements such as logical gates in a clause for logical neural networks [23] and in recurrent neural networks with external memory structures [24]. Sen et al. extended the logical neural networks' ability from inducing rules in Boolean Logic to first-order logic [25], and applied logical neural networks to knowledge base completion [26]. Riemann et al. proposed the use of neural logic rule layers as neural network modules that can represent logical rules [27]. Existing works on incorporating logical constraints typically evaluate on synthetic tasks or tasks with well-defined relationships between entities (e.g. knowledge base completion). Approaches for leveraging prior domain knowledge can be very task-specific. We are not aware of any published studies that have used the same tasks as those in our work.

2.3. Learning with loss constraints

A common and straightforward strategy to incorporate rules into neural networks without changing the network structure is to formulate the rules as differentiable constraints, and to add these constraints to the learning objective [28]. Our work falls in this category. Rule violations are penalized through loss functions. Most existing methods only deal with a single rule [7, 8, 9, 10]. Seo et al. [7] utilized an additional module in the neural network to produce outputs following the specified rule, and Hu et al. [10] used a teacher network for knowledge distillation. Ahmed et al. [9] introduced a PyTorch package to facilitate the integration of declarative knowledge into

neural network models through a constraint function. Physics-informed neural networks [29, 30] have been trained to simulate physics models described by partial differential equations for applications such as fluid dynamics, and Chen et al. [31] added geometric constraints to obey rules for 3D transformations. Fischer et al. [32] and Chen et al. [31] incorporated multiple rules but with all the rules being given the same importance, and Wang & Pan [16] applied a data-independent weight for each rule by gradient descent to minimize task loss. Our proposed confidence scores for multiple candidate rules are data-dependent and additionally regularized by prior distributions, which allow domain knowledge to guide the estimated importance of each rule. While neural network training is known to be data-intensive, we demonstrate through experiments that our incorporation of rules is especially effective in data-scarce settings.

3. Proposed Method

3.1. Notations

We denote total N observed samples as $\left\{\left(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\right)\right\}_{n=1}^{N}$, where for the n-th sample, $\boldsymbol{x}^{(n)}$ and $\boldsymbol{y}^{(n)}$ are the predictor and response, respectively. $\boldsymbol{y}^{(n)}$ is the one-hot vector of true class label in L classes. Samples are independently and identically distributed as $(\boldsymbol{x}, \boldsymbol{y}) \sim P(\mathcal{X}, \mathcal{Y})$.

A data encoder neural network model $f(\cdot)$ parameterized by Θ yields soft labels (a vector of estimated class probabilities $P(\mathcal{Y}|x)$) as $\boldsymbol{s} = f(\boldsymbol{x}; \Theta)$. The final predicted class is the one with the highest probability in $\boldsymbol{s}$ i.e. $\arg\max_i \boldsymbol{s}_i$.

3.2. Rule Loss

We consider the If-Then rule of the form $A(\boldsymbol{x}) \rightarrow B(f(\boldsymbol{x};\Theta))$, where $A(\boldsymbol{x})$ and $B(f(\boldsymbol{x};\Theta))$ are the antecedent and consequent statements of $\boldsymbol{x}$ and $f(\boldsymbol{x};\Theta)$, respectively. Then, the general form of the rule loss is

$$L_{rule}(\Theta;\boldsymbol{x}) = \mathbb{1}\{A(\boldsymbol{x})\} * penalty_{rule}(\Theta;\boldsymbol{x}) \tag{1}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function with $\mathbb{1}\{A(\boldsymbol{x})\} = 1$ if $A(\boldsymbol{x})$ is true and 0 otherwise, and $penalty_{rule}(\Theta;\boldsymbol{x})$ is the penalty for violating $A(\boldsymbol{x}) \rightarrow B(f(\boldsymbol{x};\Theta))$. Overall, Eq. 1 means that a penalty $penalty_{rule}(\Theta;\boldsymbol{x})$ is applied for rule violation when $A(\boldsymbol{x})$ is true. While the exact form of the rule loss depends on specific applications, the general form of Eq. 1 covers a variety of existing rules in the literature where statements $A(\boldsymbol{x})$ and $B(f(\boldsymbol{x};\Theta))$ are well-defined by target values or thresholds [8] or where statements are more expressive [7]. For example, for the rule "if $\boldsymbol{x} = \boldsymbol{a}$, then $f(\boldsymbol{x};\Theta) = \boldsymbol{b}$", we can set $\mathbb{1}\{A(\boldsymbol{x})\} = \mathbb{1}\{\boldsymbol{x} = \boldsymbol{a}\}$ and the penalty as $penalty_{rule}(\Theta;\boldsymbol{x}) = dist(f(\boldsymbol{x};\Theta),\boldsymbol{b})$ for some non-negative distance function $dist(\cdot)$ such that $dist(f(\boldsymbol{x};\Theta),\boldsymbol{b}) = 0$ if and only if $f(\boldsymbol{x};\Theta) = \boldsymbol{b}$. For the correlation rule "if $\boldsymbol{x}$ increases, then $f(\boldsymbol{x};\Theta)$ increases", we can represent the formulation in [7] as $\mathbb{1}\{A(\boldsymbol{x})\} = \mathbb{1}\{\boldsymbol{x} < \tilde{\boldsymbol{x}}\}$ and the penalty as $penalty_{rule}(\Theta;\boldsymbol{x}) = \text{ReLU}(f(\boldsymbol{x};\Theta) - f(\tilde{\boldsymbol{x}};\Theta))$ where $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$ for some perturbation $\boldsymbol{\epsilon}$. The specific choice of $dist(\cdot)$ or ϵ depends on the application, learning task and other loss terms in the model training objective. In our experiments, we formulated $penalty_{rule}(\Theta;\boldsymbol{x})$ based on the form of the cross-entropy classification loss for the aircraft system anomaly detection task, and an existing perturbation strategy in [7] for the credit risk prediction task and the cardiovascular classification task.

When the penalty function is differentiable with respect to Θ, the rule loss can be optimized together with the task loss when training neural network models with common gradient-based methods. This allows practitioners to readily supplement model learning with prior domain knowledge.

3.3. Confidence Learning for Multiple Rules

In practice, a human practitioner can readily construct multiple candidate rules that can potentially apply to a system, but may find it difficult to manually select the best rule or set of rules without further data analysis. We propose to use confidence learning to automatically weigh each of total J candidate rules (Figure 2), where the confidence score of a rule j in [0,1] reflects the model's estimated contribution of rule j to the system. A score of 0 implies the system is not dependent on rule j, 1 implies rule j is the single rule contributing to the system, and a value in $(0, 1)$ implies there are multiple rules contributing to the system at varying extents. In our experiments in Section 4, the confidence encoder is composed of two multi-layer perceptron (MLP) layers with ReLU activation functions, and another linear layer followed by a mean pooling and a softmax layer to generate the confidence probability score. The overall objective is

$$\begin{aligned} L(\Theta, \Psi; \boldsymbol{x}, \boldsymbol{y}) = L_{task}(\Theta; \boldsymbol{x}, \boldsymbol{y}) + \alpha L_{reg}(\Theta, \Psi) + \\ \lambda \sum_{j=1}^{J} g_j(\Psi; \boldsymbol{x}) L_{rule_j}(\Theta; \boldsymbol{x}) + \beta KL(g(\Psi; \boldsymbol{x}), \boldsymbol{w}_{prior}) \end{aligned} \tag{2}$$

where rule selection is guided by a prior distribution $\boldsymbol{w}_{prior} \in [0, 1]^J$ with $\sum_{j=1}^{J} \boldsymbol{w}_{prior,j} = 1$, and where $g(\cdot)$ is the confidence encoder network parameterized by Ψ that yields confidence scores $\boldsymbol{w} = g(\boldsymbol{x}; \Psi) \in [0, 1]^J$ and

$\sum_{j=1}^{J} \boldsymbol{w}_j = \sum_{j=1}^{J} g_j(\Psi; \boldsymbol{x}) = 1$. The KL-divergence regularization term $KL(g(\Psi; T\boldsymbol{x}), \boldsymbol{w}_{prior})$ controls how close the learned confidence scores are to the prior. The task loss $L_{task}(\Theta; \boldsymbol{x}, \boldsymbol{y})$ and regularization term $L_{reg}(\Theta, \Psi)$ on the magnitude of learned model parameters depend on the dataset, task and model choice. For our experiments, we use the conventional cross-entropy loss as the task loss for classification, and the L_2-norm on model parameters as the regularization term. The rule loss term $L_{rule_j}(\Theta; \boldsymbol{x})$ is formulated according to the choice of rules for each application, as described in Section 3.2.

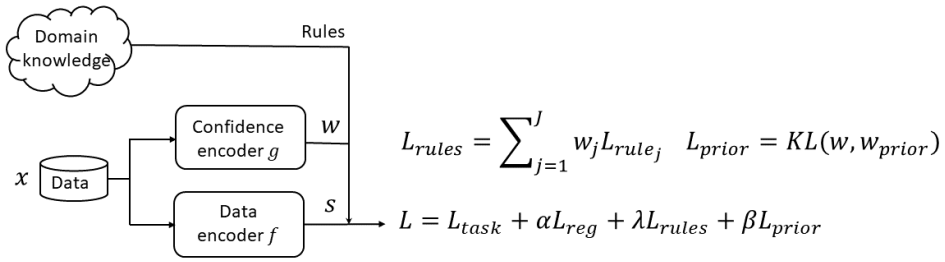


Figure 2: Domain knowledge provides If-Then rules that are simple to collect and implement. For multiple candidate rules, the contribution w of each rule is learned by a confidence neural network encoder g and guided by a prior distribution w_{prior}. The overall objective is the weighted sum of the task loss L_{task}, parameter regularization L_{reg}, the rules loss L_{rules}, and the distance measure between w and its prior.

4. Experiments and Results

We evaluated our formulation of the rule loss with three real-world datasets for a range of applications: aircraft system anomaly detection, credit risk prediction, and cardiovascular classification. We used a single rule in Section 4.1 and 4.2, and multiple candidate rules in Section 4.3. We varied the amount of available training data to demonstrate the effectiveness of injecting If-Then

rules when learning with scarce data. We set default hyperparameters as $\lambda = 1$ and $\beta = 1$ such that the task and rule loss are assigned the same importance. For the aircraft system anomaly detection task, we regularize model parameters to prevent overfitting due to the small number of observed anomalies. We set $\alpha = 0.5$, a conventional value for L_2 regularization to simplify the derivative of the term for backpropagation.

4.1. Anomaly Detection for Aircraft Thrust Reverser System

4.1.1. Dataset

We evaluated our formulation of the rule loss with a real-world dataset for anomaly detection on an aircraft system (the engines and braking systems) used for deceleration during landing. Fully-functioning and healthy engines and brakes are critically required for safe aircraft landings, and a more accurate anomaly detection model can better assist human operators in deciding when the aircraft's engines and brakes should undergo maintenance (Figure 3).

We collected operational data from a fleet of 19 aircraft, with data for each aircraft recorded for multiple flights. In total, this dataset contains 5703 flight recordings. Raw data from the upstream and downstream monitoring of related components (engines and brakes) is logged, and we extracted 16 features from the raw data. The dataset has a healthy class, and two classes of anomalies (Table 1): (1) Type A anomalies are easy to detect and can be fully detected by the status of binary features being "off" and indicating the failure to deploy any of the deceleration subsystems; (2) Type B anomalies are more challenging to detect and are the ones of interest in this application.

Table 1: Sample sizes of the aircraft dataset.

Data split	**Healthy**	**Anomaly**	
		Type A	Type B
Train	3925	44	11
Test	1683	20	20

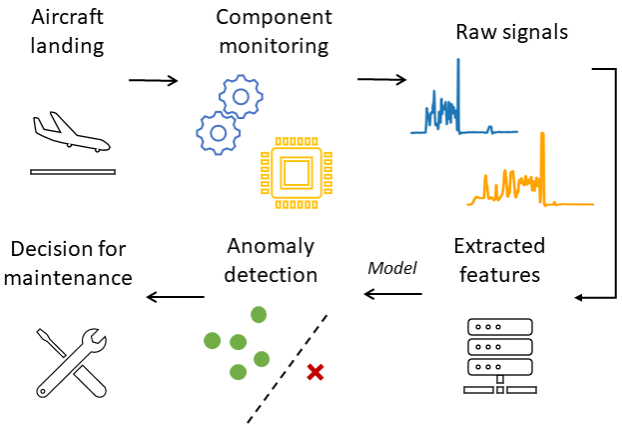


Figure 3: Anomaly detection for aircraft system

4.1.2. Implementation Details

We trained the samples with binary labels indicating if the samples are anomalies. We used a simple single-layer perceptron (SLP) since the features have already been extracted based on domain knowledge to synthesize potentially useful information. Due to class imbalance, we loaded data from two data loaders: one for healthy samples and one for anomalous samples with mini-batch size $min(256, \text{class sample size})$, and used a class-weighted cross-entropy

$$L_{task}^{AD}(\Theta; \boldsymbol{x}, \boldsymbol{y}) = -\boldsymbol{cy} \cdot \log(\boldsymbol{s}) \tag{3}$$

where $\boldsymbol{c}_\ell = \frac{\#\text{samples}/L}{\#\text{samples in class } \ell}$ and $\boldsymbol{s} = f(\boldsymbol{x}; \Theta)$ is the estimated class probabilities. The network is trained with an Adam optimizer using a learning rate of 0.1 for 1000 iterations, and checkpointed at the lowest training loss. The dataset has a single rule informed by domain knowledge: "if $\boldsymbol{x}_i > \tau$, then the sample is more likely to be an anomaly" for feature $\boldsymbol{x}_i$ denoting the desynchronization time between multiple subsystems and a threshold τ. We did not implement confidence learning in this case with a single rule, and formulated the rule loss as

$$L^{AD}_{rule}(\Theta; \boldsymbol{x}) = -\mathbb{1}\{\boldsymbol{x}_i > \tau\} * \log(f(\boldsymbol{x}; \Theta)). \tag{4}$$

Since $f(\boldsymbol{x}; \Theta)$ is the predicted probability that the sample $\boldsymbol{x}$ is an anomaly, the rule loss encourages a high predicted probability when the feature $\boldsymbol{x}_i$ crosses the domain-specific threshold τ. To understand the formulation of this rule loss with respect to the general form $\mathbb{1}\{A(\boldsymbol{x})\} * penalty_{rule}(\Theta; \boldsymbol{x})$ in Equation 1, we note that for the rule "if $\boldsymbol{x}_i > \tau$, then the sample is more likely to be an anomaly", the antecedent statement of $\boldsymbol{x}$ is $\boldsymbol{x}_i > \tau$, hence the corresponding indicator function $\mathbb{1}\{A(\boldsymbol{x})\}$ in Equation 1 is $\mathbb{1}\{\boldsymbol{x}_i > \tau\}$. The penalty function can be set as $penalty_{rule}(\Theta; \boldsymbol{x}) = dist(f(\boldsymbol{x}; \Theta), 1)$ for some non-negative distance function $dist(\cdot)$ such that $dist(f(\boldsymbol{x}; \Theta), 1) = 0$ if and only if $f(\boldsymbol{x}; \Theta) = 1$. The negative log function i.e. $penalty_{rule}(\Theta; \boldsymbol{x}) = -\log(f(\boldsymbol{x}; \Theta))$ fits the criteria as $-\log(f(\boldsymbol{x}; \Theta)) \geq 0$ for $f(\boldsymbol{x}; \Theta) \in [0, 1]$, and $-\log(f(\boldsymbol{x}; \Theta)) = 0$ if and only if $f(\boldsymbol{x}; \Theta) = 1$. Moreover, the negative log function is also present in the cross-entropy classification loss in Equation (3). Using the same function in the task and rule loss avoids imbalanced learning due to magnitude differences in loss term values.

4.1.3. Results

We evaluated the performance in detecting Type B anomalies using F1-score with top-K evaluation and the area under the precision-recall curve (AUPRC). In the top-K evaluation, the K test samples with highest anomaly probability are predicted to be anomalous, with K being the ground-truth number of anomalies. We compared our results with those of baseline methods which do not incorporate rules e.g. k-nearest neighbor (KNN), random forest, AdaBoost, gradient boosting, and SLP. We run each experiment for 10 random seeds.

Table 2: Anomaly detection performance (%) on the Aircraft dataset.

Method	F1	AUPRC
KNN	5.00 ± 0.00	2.12 ± 0.00
Random Forest	24.00 ± 2.00	25.82 ± 3.08
AdaBoost	30.00 ± 0.00	28.16 ± 1.13
Gradient Boosting	33.50 ± 7.76	32.22 ± 6.67
SLP	46.50 ± 3.20	49.31 ± 1.35
Rule	**49.00** ± **2.00**	**50.05** ± **1.33**
Δ	**+2.50**	**+0.74**

The "Rule" method achieved the best average performance over other baselines, with an F1-score of 49.00 and AUPRC of 50.05 (Table 2). Incorporating the rule marks an improvement over the use of SLP by an F1-score of 2.50 and AUPRC of 0.74. Standard deviations of the performance metrics are large due to the small number of anomalies in the test set.

4.2. Credit Risk Prediction

4.2.1. Dataset

The German credit risk dataset [3] consists of 1000 records of bank customer data, each composed of 20 numerical or categorical features. The task is to predict the credit risk of a customer. After feature selection and preprocessing [4], the resulting dataset has a feature dimension of 17. We randomly sample 40% of the data as the test set and 10% as the validation set. The remaining 50% is used for training. We used the F1-score and AUPRC as the evaluation metrics because the dataset is highly imbalanced (number of samples for bad credit is $\sim$5 times the number of samples for good credit in the training set).

4.2.2. Implementation Details

The rule we considered for this dataset is that if the checking account balance is low, then the probability of having bad credit rapidly increases [33]. We first normalized the checking amount balance to [0,1] where a higher value indicates lower balance. For a given datapoint $\boldsymbol{x}$, let x_j denote the particular feature at which we want to enforce the rule loss (i.e., checking account balance here) and τ_j denote the threshold of the rule condition. We then defined the rule loss based on a perturbation strategy [7], which was achieved by perfoming a random pertubation, i.e., adding a random value within (0,1) to x_j to get $\tilde{\boldsymbol{x}}$. This rule loss is formulated as:

$$L_{rule}^{CR}(\Theta; \boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathbb{1}(x_j > \tau_j) \cdot \max(f(\boldsymbol{x}; \Theta) - f(\tilde{\boldsymbol{x}}; \Theta) + \delta, 0), \tag{5}$$

[3] https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[4] https://www.kaggle.com/datasets/uciml/german-credit

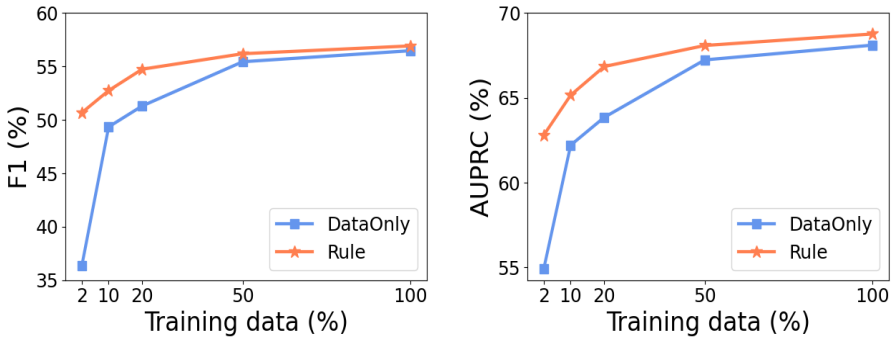


Figure 4: Classification performance on the German credit risk dataset with respect to different training data percentages. Left: F1-score; right: AUPRC.

where δ is a margin hyper-parameter controlling how strictly the rule is enforced. Applying Eq. 5 means that when the condition $x_j > \tau_j$ is satisfied, the predicted probability of the example having bad credit should increase when x_j increases, i.e., $f(\tilde{\boldsymbol{x}}; \theta)$ should be higher than $f(\boldsymbol{x}; \theta)$. Otherwise, we enforced a loss of $\max(f(\boldsymbol{x}; \Theta) - f(\tilde{\boldsymbol{x}}; \Theta) + \delta, 0)$ to penalize such a violation. In our implementation, we set the rule threshold τ_j as 0.75. As there is only a single rule available for this dataset, we did not perform confidence learning. For the task loss, we used a class-weighted cross-entropy loss (Eq. 3).

4.2.3. Results

Our baseline is the multi-layer perception (MLP) neural network without incorporating rules ("DataOnly"). The performance of the neural network with respect to different training data percentages averaged over 20 runs (Figure 4) indicates that adding rules consistently improves the classification performance in terms of both F1-score and AUPRC. With less training data available, the performance boost is more significant. This suggests that our

technique for incorporating rules is more useful when limited labeled training data is available. This finding is notable, as data annotation is known to be a pain-point when using deep learning methods for real-world applications.

4.3. Cardiovascular Classification

4.3.1. Dataset

The cardiovascular classification dataset [5] consists of 70,000 records of patient data, each containing 11 features and a binary label indicating the presence or absence of cardiovascular disease. The data is evenly distributed, i.e., half of the patients have cardiovascular disease, and the other half does not. The task is to classify the data based on the 11 categorical or numerical features. To demonstrate the utility of incorporating rules in a scenario with scarce training data, we randomly sampled 80% of the data as test examples. From the remaining 20% data, we further randomly sampled the same amount of data for training and validation. The validation data were used for checkpointing models and choosing the best learning rate. We applied the same procedure to all neural network-based methods in our evaluation to ensure fair comparisons.

4.3.2. Implementation Details

According to the domain knowledge from medical studies [34, 35], factors that are known to be strongly associated with cardiovascular disease are higher systolic blood pressure (ap_hi), age and obesity measured by body mass index (bmi). Therefore, the rules we considered are:

[5] https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

a) If $ap\_hi > 129.5$, then the probability of getting cardiovascular disease greatly increases.

b) If $age > 50$, then the probability of getting cardiovascular disease greatly increases.

c) If $bmi > 25$, then the probability of getting cardiovascular disease greatly increases.

For each rule, we defined the rule loss in the same way as the credit risk prediction task (Sec. 4.2.2), i.e., we performed a random perturbation over the normalized feature x_j of $\boldsymbol{x}$ to obtain $\tilde{\boldsymbol{x}}$. The rule loss was then formulated in the same manner as Eq. 5. The rule thresholds τ_j for the three rules were normalized from 129.5, 50 and 25 for the features ap_hi, age, bmi respectively in the same way as we normalized the features. We applied confidence learning in this case with 3 candidate rules. For the confidence score priors, we used the winner-take-all strategy to assign 1 to the most confident Rule #a according to [34] and 0 for other rules. We performed ablation studies using the different confidence priors given in Sec. 4.3.4. For the task loss, we used the standard binary cross-entropy loss since the data distribution is balanced in the training set.

4.3.3. Results

Our baseline is the multi-layer perception (MLP) neural network without incorporating rules ("DataOnly"). For our method, we report the results of using a single rule (Rule #a) and a simple baseline that uses average weights for each rule without confidence learning, denoted as "Rule (multi_avg)".

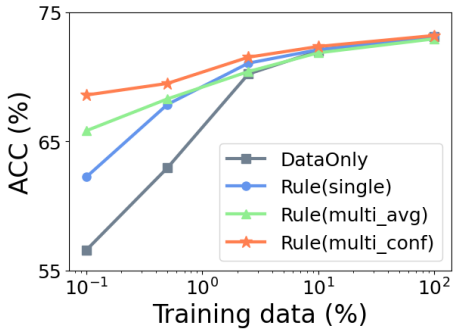


Figure 5: Classification performance on the Cardiovascular dataset with respect to different training data percentages.

Our full method with multiple rules and confidence learning is reported here as "Rule (multi_conf)".

Table 3: Classification accuracies (%) on the Cardiovascular dataset with respect to different training data percentages.

Training (%)	0.1%	0.5%	2.5%	10%	100%
MLP	56.57 ± 4.78	62.93 ± 5.97	70.20 ± 0.60	71.98 ± 0.56	73.07 ± 0.21
Rule (single)	62.23 ± 6.60	67.84 ± 2.69	71.04 ± 1.03	72.08 ± 0.26	73.14 ± 0.11
Rule (multi_avg)	65.81 ± 1.53	68.57 ± 1.31	70.39 ± 0.87	71.84 ± 0.85	72.92 ± 0.13
Rule (multi_conf)	**68.57 ± 1.42**	**69.07 ± 2.45**	**71.50 ± 0.45**	**72.33 ± 0.25**	**73.18 ± 0.12**
Δ_{acc}	**+12.0**	**+6.14**	**+1.30**	**+0.35**	**+0.11**

Table 4: Validation errors on the Cardiovascular dataset with respect to different training data percentages.

Training (%)	0.1%	0.5%	2.5%	10%	100%
MLP	0.553 ± 0.190	0.587 ± 0.067	0.579 ± 0.043	0.563 ± 0.018	**0.548 ± 0.006**
Rule (single)	0.413 ± 0.183	0.585 ± 0.066	0.574 ± 0.040	0.564 ± 0.018	0.549 ± 0.006
Rule (multi_avg)	0.416 ± 0.217	0.572 ± 0.066	**0.571 ± 0.037**	0.564 ± 0.019	0.553 ± 0.006
Rule (multi_conf)	**0.402 ± 0.186**	**0.569 ± 0.076**	**0.573 ± 0.033**	**0.562 ± 0.018**	**0.548 ± 0.005**

Our results averaged over 10 runs indicate that incorporating rules generally improves the performance over the baseline methods in all the settings (Table 3, Figure 5). The improvement is more marked in cases with small training data, and Rule with confidence learning attains a 12.0% accuracy boost over MLP at the 0.1% training data setting. Our finding is consistent with the results given in Figure 4, and demonstrates the value of incorporating rules in scenarios with limited training data. We further report the validation errors (mean±std) in Table 4, which shows similar trends as in Table 3. Our method that incorporates rules with confidence learning generally achieves the lowest validation errors at all settings.

4.3.4. Ablation Study

We conducted ablation studies and analyses on the cardiovascular classification dataset with 0.05% training data.

Using different priors. We first studied the effect of using different priors for confidence learning with multiple candidate rules. In our experiment, we used the 'winner-take-all strategy to assign 1 to the most confident rule and 0 to the rest (denoted as "One-hot"). An alternative was to simply use equally weighted or average confidence scores, meaning that we would not have any preference or knowledge about the confidence of each rule (denoted as "Average"). We also studied the effect of excluding and including the KL-divergence loss. In our results (Table 5), "Fixed" denotes the confidence scores being fixed without learning, "Learn w/o KL loss" denotes learning the confidence scores without the KL-divergence loss, "Learn w KL loss (Full model)" denotes our confidence learning method that includes the KL divergence loss. We can see that all evaluated models outperform the MLP

baseline (with classification accuracy 56.57±4.78) by considerable margins, which highlights the effectiveness of incorporating multiple rules. Our full model with the "One-hot" prior achieves the best performance. It is also worth noting that without the KL-divergence loss, the model will simply assign the largest weight to the rule with the smallest rule loss, regardless of priors. This is validated by the fact that the same performance is achieved by using "Average" and "One-hot" priors for the "Learn w/o KL loss" model.

Table 5: Classification accuracy (%) on the Cardiovascular dataset for ablation study on using different priors for confidence learning.

Prior	Average	One-hot
Fixed	65.81±1.53	63.75±7.03
Learn w/o KL loss	64.25±4.89	64.25±4.89
Learn w KL loss (Full model)	66.74±1.38	68.57±1.42

Table 6: Classification accuracy (%) on the Cardiovascular dataset for ablation study on the rule margin parameter δ (bigger δ indicates more strict rules).

MLP	56.57 ± 4.78
Rule, $\delta = 0$	63.39 ± 3.48
Rule, $\delta = 0.001$	64.66 ± 3.40
Rule, $\delta = 0.005$	67.96 ± 0.95
Rule, $\delta = 0.01$	**68.57 ± 1.42**
Rule, $\delta = 0.05$	**68.86 ± 1.67**
Rule, $\delta = 0.1$	67.06 ± 2.55

Impact of δ. In the design of the rule loss, the margin parameter δ controls how strict the constraint is enforced. We studied the effect of varying δ on classification performance (Table 6). Even with $\delta = 0$, incorporating

rules outperforms the MLP baseline by a considerable margin. When we increase δ, the classification performance improves and peaks at $\delta = 0.01$ and $\delta = 0.05$. Since the standard deviation of performance is smaller at $\delta = 0.01$ than at $\delta = 0.05$, we set $\delta = 0.01$ for all our experiments.

5. Conclusion

The performance of neural networks is dependent on the quantity of training data available. Incorporating human expert knowledge into model training can reduce the burden on data requirements. In this paper, we introduced a general, model-agnostic rule loss for incorporating conditional sentences into neural network training for classification tasks. We also proposed to use confidence learning for automatically weighing the contribution of multiple candidate rules. We empirically demonstrated performance increases by incorporating rules in three real-world datasets, showing that this effect can be especially significant when training data is limited.

Acknowledgement

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Grant A20H6b0151). We would also like to thank the IAF-ICP grant I2001E0076 for funding part of this study.

References

[1] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, M. Walczak, J. Pfrommer, A. Pick, R. Ramamurthy,

J. Garcke, C. Bauckhage, J. Schuecker, Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems, IEEE Transactions on Knowledge and Data Engineering (2021).

[2] S. Cuomo, V. S. D. Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: Where we are and what's next, Journal of Scientific Computing 92 (88) (2022).

[3] S. N. Tran, A. S. d'Avila Garcez, Deep logic networks: Inserting and extracting knowledge from deep belief networks, IEEE Transactions on Neural Networks and Learning Systems 29 (2) (2018) 246–258.

[4] L. de Penning, A. Garcez, L. Lamb, J.-j. Meyer, A neural-symbolic cognitive agent for online learning and reasoning, International Joint Conference on Artificial Intelligence (IJCAI) (2011).

[5] Y. Gorishniy, I. Rubachev, V. Khrulkov, A. Babenko, Revisiting deep learning models for tabular data, Advances in Neural Information Processing Systems (NeurIPS) (2021).

[6] S. Saha, J. Roy, T. K. Hembram, B. Pradhan, A. Dikshit, K. Maulud, A. Alamri, Comparison between deep learning and tree-based machine learning approaches for landslide susceptibility mapping, Water 13 (09 2021).

[7] S. Seo, S. Arik, J. Yoon, X. Zhang, K. Sohn, T. Pfister, Controlling neural networks with rule representations, Advances in Neural Information Processing Systems (NeurIPS) (2021).

[8] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G. Van den Broeck, A semantic loss function for deep learning with symbolic knowledge, International Conference on Machine Learning (ICML) (2018).

[9] K. Ahmed, T. Li, T. Ton, Q. Guo, K.-W. Chang, P. Kordjamshidi, V. Srikumar, G. V. den Broeck, S. Singh, PYLON: A PyTorch framework for learning with constraints, AAAI Conference on Artificial Intelligence (2022).

[10] Z. Hu, X. Ma, Z. Liu, E. Hovy, E. Xing, Harnessing deep neural networks with logic rules, Annual Meeting of the Association for Computational Linguistics (ACL) (2016).

[11] S. Yang, X. He, B. Zhu, Learning physical constraints with neural projections, Advances in Neural Information Processing Systems (NeurIPS) (2020).

[12] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, K. Kavukcuoglu, Interaction networks for learning about objects, relations and physics, Advances in Neural Information Processing Systems (NIPS) (2016).

[13] Y. S. Abu-Mostafa, Learning from hints in neural networks, Journal of Complexity 6 (2) (1990) 192–198.

[14] L. Gan, K. Kuang, Y. Yang, F. Wu, Judgment prediction via injecting legal knowledge into neural networks, AAAI Conference on Artificial Intelligence (2021).

[15] J. Yu, J. Jiang, R. Xia, Global inference for aspect and opinion terms

co-extraction based on multi-task neural networks, IEEE/ACM Transactions on Audio, Speech, and Language Processing 27 (1) (2019) 168–177.

[16] W. Wang, S. Pan, Integrating deep learning with logic fusion for information extraction, AAAI Conference on Artificial Intelligence (2020).

[17] N. Muralidhar, M. R. Islam, M. Marwah, A. Karpatne, N. Ramakrishnan, Incorporating prior domain knowledge into deep neural networks, IEEE International Conference on Big Data (2018).

[18] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, Deepproblog: Neural probabilistic logic programming, Advances in Neural Information Processing Systems (NeurIPS) (2018).

[19] Z. Yang, A. Ishay, J. Lee, NeurASP: Embracing neural networks into answer set programming, International Joint Conference on Artificial Intelligence (IJCAI) (2020).

[20] Z. Yang, J. Lee, C. Park, Injecting logical constraints into neural networks via straight-through estimators, International Conference on Machine Learning (ICML) (2022).

[21] G. G. Towell, J. W. Shavlik, Knowledge-based artificial neural networks, Artificial Intelligence 70 (1) (1994) 119–165.

[22] A. Garcez, G. Zaverucha, The connectionist inductive learning and logic programming system, Applied Intelligence 11 (1999) 59–77.

[23] R. Riegel, A. G. Gray, F. P. S. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, S. Ikbal, H. P.

Karanam, S. Neelam, A. Likhyani, S. K. Srivastava, Logical neural networks, arXiv (2020).

[24] A. A. Mali, A. G. Ororbia II, C. L. Giles, A neural state pushdown automata, IEEE Transactions on Artificial Intelligence 1 (3) (2020) 193–205.

[25] P. Sen, B. W. S. R. de Carvalho, R. Riegel, A. G. Gray, Neuro-symbolic inductive logic programming with logical neural networks, AAAI Conference on Artificial Intelligence (2022).

[26] P. Sen, B. W. Carvalho, I. Abdelaziz, P. Kapanipathi, S. Roukos, A. Gray, Logical neural networks for knowledge base completion with embeddings & rules, Conference on Empirical Methods in Natural Language Processing (EMNLP) (2022).

[27] J. N. Reimann, A. Schwung, S. X. Ding, Neural logic rule layers, Information Sciences 596 (2022) 185–201.

[28] H. Narasimhan, Learning with complex loss functions and constraints, International Conference on Artificial Intelligence and Statistics (AISTATS) (2018).

[29] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[30] S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis, Physics-informed

neural networks (pinns) for fluid mechanics: A review, Acta Mechanica Sinica 37 (2021) 1727 – 1738.

[31] Y. Chen, C. Schmid, C. Sminchisescu, Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera, IEEE/CVF International Conference on Computer Vision (ICCV) (2019).

[32] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, M. Vechev, DL2: Training and querying neural networks with logic, International Conference on Machine Learning (ICML) (2019).

[33] M. Hashemi, A. Fathi, Permuteattack: Counterfactual explanation of machine learning credit scorecards, CoRR (2020).

[34] W. B. Kannel, Elevated systolic blood pressure as a cardiovascular risk factor, The American Journal of Cardiology (2000) 251–255.

[35] R. B. D'Agostino Sr, R. S. Vasan, M. J. Pencina, P. A. Wolf, M. Cobain, J. M. Massaro, W. B. Kannel, General cardiovascular risk profile for use in primary care, Circulation (2008) 743–753.