Learning predictive checklist from arbitrary data modalities with probabilistic logic programming

Anonymous Authors[1]

1. Introduction

Recent years have seen a growing development of machine learning models in the healthcare domain thanks to their impressive performance on a wide range of medical tasks such as ... (Davenport & Kalakota, 2019; Esteva et al., 2019). However, despite the proliferation of architectures, the adoption of machine learning models in clinical practice remains a significant challenge (Futoma et al., 2020; Ahmad et al., 2018; Ghassemi et al., 2020; De Brouwer et al., 2022). Indeed, ensuring the level robustness required for healthcare applications is difficult for complex models due to their inherent black box nature. Non-interpretable models make stress testing arduous and thus hamper the confidence required to deploy them in critical applications such as clinical practice. Recent works have thus aimed at developing novel architectures that are both interpretable and retain most of the performance from their black box counterparts (Ahmad et al., 2018).

One such approach is learning medical checklists from available medical records. Due to their simplicity and their ability to assist clinicians in complex situations, checklists have become increasingly popular in medical practice (Haynes et al., 2009). However, the design of such checklists is usually performed by expert clinicians, who manually collect evidence about the particular clinical problem of interest (Hales et al., 2008). As the number of available medical records grows, the manual collection of evidence becomes more tedious, bringing the need for partially automated design of medical checklists. Recent works have taken a step in that direction by learning predictive checklists from boolean, categorical medical or continuous tabular data (Zhang et al., 2021; Makhija et al., 2022).

Nevertheless, many available clinical data, such as images or time series, are nor categorical nor tabular by nature. They therefore fall outside the limits of applicability of previous approaches for learning checklists from data.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

In this work, we relax the data modalities assumptions and propose a novel approach to learning checklists from arbitrary type of input data. We resolve the previous limitations by formulating checklists withing a probabilistic logic programming framework that allows to extract predictive boolean concepts from the data that eventually serve as a support for the checklist. On top of accepting a larger set of data modalities, we show that our approach is also faster than previous checklist learning models.

Contributions.

- We propose the first framework to learn predictive checklists from arbitrary input data modalities. Our approach can learn checklist and extract concepts from time series and images, among others.
- In contrast with previous works, our approach avoids using a (mixed-)integer programming formulation. This makes ProbChecklistfaster and more amenable to modern deep learning optimization schemes.
- We validate our learning framework on different data modalities such as images and clinical time series, displaying significantly improved performance compared to state-of-the-art checklist learning schemes.

2. Related works

Our works shares its motivation with the interpretable machine learning literature. Conceptually, it builds upon the recent body of work on learning predictive checklists from data. Finally, our model formulation is directly inspired by the literature on probabilistic logic programming.

Interpretable machine learning. Motivated by the lack of robustness and trust of black box models, a significant effort has been dedicated to developing more human-interpretable machine learning models in the last years (Ahmad et al., 2018; Murdoch et al., 2019). Among them, one distinguishes between *intrinsic* (*i.e.* when the model is itself intepretable such as decision trees)) and *posthoc* (*i.e.* when trained models are interpreted a posteriori) methods (Du et al., 2019). Checklists belong to the former category as

they an intuitive and easy to use decision support tool. Compared to decision trees, checklist are more concise (there is no branching structure) and can thus be potentially more effective in high stress environments (). Our approach also relies on building concepts from the input data. Because the concepts are learnt from data, they may themselves lack a clear interpretation. Both intrinsic and posthoc interpretability techniques can then be applied for the concept extraction pipeline.

Friede & Niepert (2021) proposes new "tricks" to learn discrete-continuous computation graphs.

Checklist learning. Due to their simplicity and their ability to assist clinicians in complex situations, checklists have become increasingly popular in medical practice (Haynes et al., 2009). However, the design of such checklists is usually performed by expert clinicians, who manually collect evidence about the particular clinical problem of interest (Hales et al., 2008). As the number of available medical records grows, the manual collection of evidence becomes more tedious, bringing the need for partially automated design of medical checklists. Recent works have taken a step in that direction by learning predictive checklists from boolean or categorical medical data (Zhang et al., 2021). Makhija et al. (2022) have extended this approach by allowing for continuous tabular data using mixed integer programming. Our work builds upon these recent advances but allows for arbitrary input data modalities such as time series or images. What is more, in contrast to previous works, our method does not rely on integer programming and thus exhibits much faster times and is more amenable to the most recent deep learning stochastic optimization schemes.

Probabilistic logical programming. Probabilistic logic reasoning combines logic and probability theory. It represents a refreshing framework from deep learning in the path towards artificial intelligence, focusing on high-level reasoning. Examples of areas relying on these premises include statistical artificial intelligence (Raedt et al., 2016; Koller et al., 2007) and probabilistic logic programming (De Raedt & Kimmig, 2015). More recently, researchers have proposed hybrid architectures, embedding both deep learning and logical reasoning components (Santoro et al., 2017; Rocktäschel & Riedel, 2017; Manhaeve et al., 2018).

Probabilistic logic reasoning has been identified as important component for explainable or interpretable machine learning, due to its ability to incorporate knowledge graphs (Arrieta et al., 2020). Combination of deep learning and logic reasoning programming have been implemented in interpretable computer vision tasks, among others (Bennetot et al., 2019; Oldenhof et al., 2023).

Prob log is a good way to combine discrete and continuous components in an architecture.

3. Methods

3.1. Problem Statement

We consider a supervised learning problem where we have access to N input data points $\mathbf{x}_i \in \mathcal{X}$ and corresponding binary labels $y_i \in \{0, 1\}$. Each input data point consists of a collection of K data modalities: $\mathbf{x}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^K\}$. Each data modality can either be continuous ($\mathbf{x}_i^k \in \mathbb{R}^{d_k}$) or binary ($\mathbf{x}_i^k \in \{0, 1\}^{d_k}$). Examples of continuous data modalities include images, time series, or tabular data. Categorical data are assumed to be represented in expanded binary format. We set d as the overall dimension of $\mathbf{x}_i$. That is, $d = \sum_{k=1}^{K} d_k$. The N input data points and labels are aggregated in a data structure $\mathbf{X}$ and a vector $\mathbf{y}$ respectively.

Our objective is to learn a decision function $f : \mathcal{X} \to \{0, 1\}$ from some domain $\mathbb{F}$ that minimizes some error criterion d between the predicted and the true label. The optimal function f^* then writes:

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathbf{x},\mathbf{y} \sim \mathcal{D}}[d(f(\mathbf{x}), \mathbf{y})], \tag{1}$$

where $\mathcal{D}$ stands for the observational data distribution.

3.2. Background

3.2.1. Predictive checklists

Generally, we define a predictive checklist as a linear classifier applying on a list of M binary concepts $\mathbf{c}_i \in \{0, 1\}^M$. A checklist will predict a data point, consisting of M concepts $\mathbf{c}_i = \{c_i^1, \ldots, c_i^M\}$, as positive if the number of concepts such that $c_i^m = 1$ is larger or equal to a threshold T. That is, given a data point with concepts $\mathbf{c}_i$, the predicted label of a checklist with threshold T writes:

$$\hat{y}_i = \begin{cases} 1 & \text{if} \quad \sum_{m=1}^{M} c_i^m \geq T \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The only parameter of a checklist is the threshold T. Nevertheless, the complexity lies in the definition of the list of concepts that will be given as input to the checklist. This step can be defined as mapping ϕ that produces the binary concepts from the input data:

$$\mathbf{c}_i = \phi(\mathbf{x}_i). \tag{3}$$

Existing approaches for learning checklists from data differ by their mapping ϕ. Zhang et al. (2021) assume that the

input data is already binary. The mapping ϕ_M is then a binary matrix $\Phi \in \{0,1\}^{M \times k}$ such that $\Phi \mathbf{1}_k = \mathbf{1}_k$, where $\mathbf{1}_k$ is a column vector of ones[1]. One then computes $\mathbf{c}_i$ as $\mathbf{c}_i = \Phi_M \mathbf{x}_i$. The element of Φ_M as well as the number of concepts M (hence the dimension of the matrix) are learnable parameters.

The approach of Makhija et al. (2022) relaxes the binary input data assumption by allowing for the creation of binary concepts from continuous data through thresholding. Writing $\mathbf{x}_i^b$ and $\mathbf{x}_i^c$ for the binary and real parts of the input data respectively, the concept creation mechanism transforms the real data to binary with thresholding and then uses the same matrix Φ_M. We have $\mathbf{c}_i = \Phi_M[\mathbf{x}_i^b, \text{sign}(\mathbf{x}_i^b - \mathbf{t}_i)]$, where $[\cdot,\cdot]$ is the concatenation operator, $\mathbf{t}_i$ is a vector of thresholds, $\text{sign}(\cdot)$ is an element-wise function that returns 1 is the element is positive and 0 otherwise. In this formulation one learns the number of concepts M, the binary matrix Φ_M as well as the thresholds values $\mathbf{t}_i$.

Different formulations for ϕ lead to different implementations strategies to efficiently learn the resulting parameters from the available data. The approach of Zhang et al. (2021) requires learning a binary matrix and thus uses integer programming. Conversely, the approach of Makhija et al. (2022) relies on mixed integer programming to jointly optimize the threholds $\mathbf{t}_i$.

3.3. ProbChecklist

Overall description of the method.

3.3.1. Concept Extractor

As described in Section 3.2, learning predictive checklist hinges on extracting binary concepts from the input data. Instead of directly learning binary concepts, we extract soft concepts that we subsequently discretize. For each of the K data modalities, we have a soft concept extractor $\psi_k : \mathbb{R}^{d_k} \to [0,1]^{d'_k}$ that maps the input data to a vector of probabilities $\mathbf{p}_i^k$, where d'_k is the number of soft concepts to be extracted from data modality k. Concatenating the outputs of the K concept extractors results in a vector of probabilities $\mathbf{p}_i \in [0,1]^{d'}$, with the d' the total number of soft concepts.

3.3.2. Checklist inference

The checklist prediction formula of Equation 2 can be understood as logical rules in a probabilistic logical program. Together with the probabilities of each concepts, encoded in vector $\mathbf{p}_i$ that represent d' probabilistic facts, this represents a probabilistic logical program $\mathcal{P}_\theta$. We refer to θ as the set of learnable parameters in the probabilistic logical program.

[1]This corresponds effectively to every row of Φ summing to 1.

We want to maximize the probability of a the prediction being correct. That is, we want to maximize the probability of the query $q := \hat{y}_i = y_i$,

$$\hat{\theta} = \arg\min_\theta -P_{\mathcal{P}_\theta}(\hat{y}_i = y_i) \tag{4}$$

$$= \arg\min_\theta -\sum_w P(w) \cdot \mathbb{I}[F(w) \equiv (\hat{y}_i = y_i)] \tag{5}$$

By interpreting the probabilities $\mathbf{p}_i$ as the probability that the corresponding binary concepts are equal to 1 (*i.e.* $\mathbf{p}_i[j] = P(\mathbf{c}_i[j] = 1)$, where $[j]$ indexes the j-th component of the vector), we have the following loss.

Proposition 3.1. *The probability of the query $\hat{y}_i = y_i$ in the predictive checklist is given by*

$$P_{\mathcal{P}_\theta}(\hat{y}_i = 1) = 1 - P_{\mathcal{P}_\theta}(\hat{y}_i = 0)$$
$$= \sum_{t=T}^{d'} \sum_{\sigma \in \Sigma^d} \prod_{j=1}^{d'} (\mathbf{p}_i[j])^{\sigma(j)} (1 - \mathbf{p}_i[j])^{1-\sigma^d(j)}$$

where Σ_d is the set of selection functions $\sigma : [d'] \to \{0,1\}$ such that $\sum_{j=1}^{d'} \sigma(j) = d$.

The detailed derivations are presented in Appendix A. We use the log-likelihood as the loss function, which leads to the following binary cross entropy:

$$\mathcal{L} = -(y_i(P_{\mathcal{P}_\theta}(\hat{y}_i = 1)) + (1 - y_i)(P_{\mathcal{P}_\theta}(\hat{y}_i = 0))) \tag{6}$$

$$\mathcal{L} = -(y_i(P_{\mathcal{P}_\theta}(\sum_{m=1}^{M} c_m \geq T) + (1 - y_i)(P_{\mathcal{P}_\theta}(\sum_{m=1}^{M} c_m < T))) \tag{7}$$

3.3.3. Learning

The set of parameters of the checklist, θ, comprise multiple elements: the parameters of the different soft concept extractors (θ_ψ), the number of concepts to be extracted for each data modality d_k, the checklist threshold T. As the soft concept extractors are typically parametrized by neural networks, optimizing $\mathcal{L}$ with respect to θ_ψ can be achieved via gradient based methods. d_k and T are constrained to be integers and are thus treated as hyper-parameters in our experiments.

3.3.4. Checklist construction

The construction above relies on soft concepts extraction for each data modality. Yet, at test time, a checklist operates on binary input data. We thus binarize the predicted soft concepts by setting

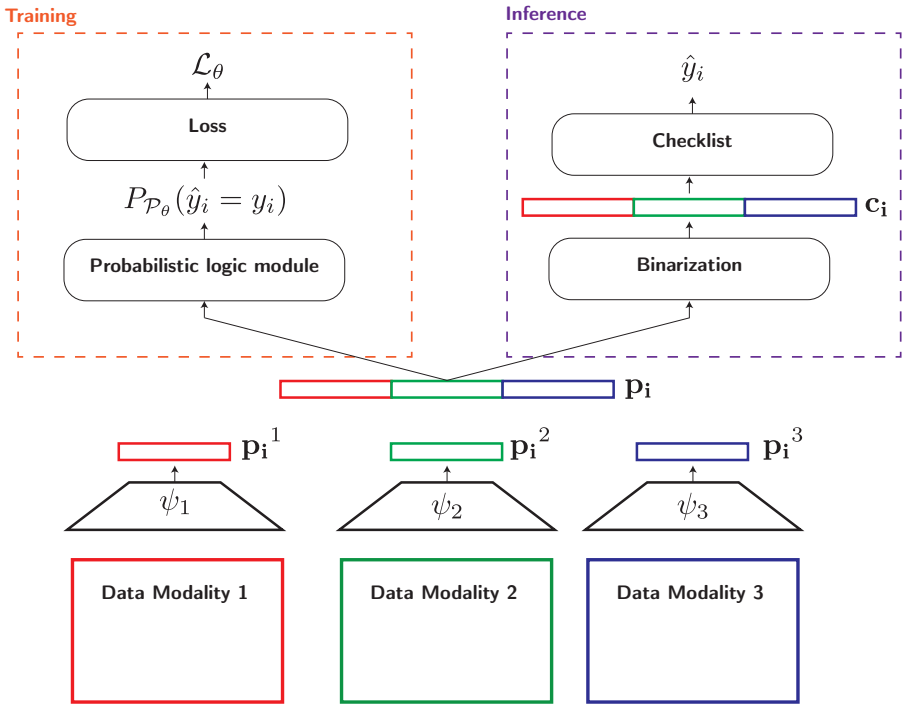

Figure 1. Description of the overall architecture

$$\mathbf{c}_i[j] = \mathbb{I}[\mathbf{p}_i[j] > 0.5]. \tag{8}$$

What is more, after training, we construct the final checklist by pruning the concepts that are never used in the training data (*i.e.* concepts j such that $\mathbf{c}_i[j] \forall i$ are pruned).

3.3.5. Interpretability of the concept extractors

Discussion about how interpretable the concept extractors are.

4. Experiments

In this section, we demonstrate our method, ProbChecklist, on three prediction tasks.

4.1. MNIST Checklist

Data. We generate a synthetic dataset from MNIST and define a rule set based on which the instances are classified as 0 or 1. We divide all the images in the MNIST dataset into groups of $\mathbf{K}$ images, and each group forms one sample. In the experiments, we set $\mathbf{K} = \mathbf{4}$, which creates a dataset consisting of 10500 samples for training, 4500 for validation, and 2500 for testing. We start by learning $\mathbf{M}$ concepts from images directly using $\mathbf{K}$ concepts learners, which will later be used for recovering the set checklist. Concept learners would ensure that additional information beyond the labels is also captured.

We construct a ground truth checklist using the image labels to simulate a binary classification setup. A sample is assigned $\mathbf{y} = +\mathbf{1}$ if $\mathbf{T}$ out of $\mathbf{K}$ items are true. The final checklist used for experimentation is:

- **Ground Truth Checklist ($\mathbf{T} = 3$, $\mathbf{K} = 4$)**
  - ☐ **Image 1** $\in \{0, 2, 4, 6, 8\}$
  - ☐ **Image 2** $\in \{1, 3, 5, 7, 9\}$
  - ☐ **Image 3** $\in \{4, 5, 6\}$
  - ☐ **Image 4** $\in \{6, 7, 8, 9\}$

The final dataset contains 20.44% positive samples.

Architecture. We define a simple CNN architecture with three convolutional layers as the concept learner for this task. All K images of a given sample are presented to different CNNs, which output a $\mathbf{d_k}$-dimensional concept probability vector. We then concatenate them to obtain all the $\mathbf{M}$ concepts for given sample ($\mathbf{M} = \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{d_k}$). This forms the input to the probabilistic logic module where the query $q := \mathbb{P}(\mathbf{d} > \mathbf{T})$ is computed, d is the number of positive concepts ($\mathbf{d} = \sum_{\mathbf{m=1}}^{\mathbf{M}} \mathbf{c_m}$). The loss used for computing the gradient update is the cross-entropy between the ground truth label of the sample and the $\mathbb{P}(\mathbf{d} > \mathbf{T})$.

Baselines. We compare the performance of our method against several classifiers including a logistic regression (LR), a classical multilayer perceptron (MLP), and the MIP from Makhija et al. (2022). The process to obtain the M-dimensional concept probabilities is kept the same for all methods.

Results. In Table 1, we report the results of the baselines and ProbChecklist for $\mathbf{d_k} = \mathbf{4}$ $(\mathbf{M} = \mathbf{16})$ on the test samples, and in Table 2, we show the performance effects of varying $\mathbf{d_k}$.

Our method outperforms all the baselines, not only in terms of accuracy but also achieves a higher recall indicating that it identifies the minority class better than these standard approaches. The MIP failed to find solutions for some folds of the dataset and didn't generalise well on the test samples.

Model	Accuracy	ROC-AUC	Precision	Recall	Specificity	T	M
Dummy Classifier	20.44	-	0.2024	1	0.7956	-	-
CNN + MLP	94.72 ± 4.32	0.974 ± 0.03	0.895 ± 0.1	0.835 ± 0.13	0.976 ± 0.02	-	-
CNN + LR	95.04 ± 0.31	0.985 ± 0.002	0.914 ± 0.01	0.836 ± 0.016	0.98 ± 0.003	-	-
pretrained CNN + MIP	79.56	-	0	0	1	8	13.5 ± 0.5
Ours	97.04 ± 0.177	0.994 ± 0.001	0.937 ± 0.005	0.917 ± 0.006	0.984 ± 0.001	8.4 ± 1.2	16

Table 1. Performance results on synthetic MNIST Checklist dataset. We report accuracy, precision, recall as well as conciseness of the learnt checklist.

A significant improvement in the performance was observed when $\mathbf{d_k}$ increased from $\mathbf{1}$ to $\mathbf{2}$, which suggests that having learning one concept per image is inadequate to capture all the signal in the sample. It is also interesting to note that the performance reaches a saturation point after $\mathbf{d_k} = 3$. The value of $\mathbf{d_k}$ can be tuned to find the optimal number of concepts for a given data modality.

$\mathbf{d_k}$	Accuracy	Precision	Recall	Specificity	T	M
1	72.63 ± 1.42	0.427 ± 0.013	0.979 ± 0.005	0.661 ± 0.018	3	4
2	92.768 ± 1.6	0.752 ± 0.045	0.97 ± 0.006	0.917 ± 0.02	5	8
3	96.52 ± 0.33	0.9 ± 0.009	0.933 ± 0.008	0.973 ± 0.002	7	12
4	**96.808 ± 0.24**	0.917 ± 0.015	0.929 ± 0.01	0.978 ± 0.004	8.4 ± 1.2	16
5	96.68 ± 0.269	0.918 ± 0.013	0.92 ± 0.009	0.979 ± 0.004	9.4 ± 1.36	20

Table 2. MNIST Checklist: Performance of ProbChecklist with varying $\mathbf{d_k}$

4.2. Sepsis Prediction using Static Tabular Data

Data. We use the PhysioNet 2019 Early Sepsis Prediction time series dataset (Reyna et al., 2019), which was collected from the ICUs of three hospitals. Hourly data of vital signs and laboratory tests are available for the thirty-four non-static features in the dataset. Additionally, four static parameters, gender, duration, age, and anomaly start point, are included. We treat the occurrence of sepsis in patients as the binary outcome variable.

The original dataset contains 32,268 patients, with only 8% sepsis patients. We create five subsets with 2200 patients, of which nearly 37% are positive. For this task, we use basic summary extraction functions such as the mean, standard deviation, and last entry of the clinical time series of each patient to tabulate the dataset. We subsequently perform feature selection and only keep the top ten informative features $(\mathbf{K} = \mathbf{10})$ based on logistic regression weights.

Architecture. Since each feature is simply a single-valued function $(\mathbf{x_i} \in \mathbb{R})$, the concept learners are single linear layers followed by a sigmoid activation function. The remaining steps are the same as the previous task, i.e. these concept probabilities are then passed to the probabilistic logic module for $\mathbb{P}(\mathbf{d} > \mathbf{T})$ computation and loss backpropagation.

Baselines. We use standard baselines like MLP and LR, along with checklist-specific architectures, namely Unit weighting, SETS checklist, Integer Linear Program (Zhang et al., 2021) with mean thresholds, MIP from Makhija et al. (2022). Unit weighting distils a pre-trained logistic regression into a checklist, as also used in Zhang et al. (2021). SETS checklist (Makhija et al., 2022) consists of a modified logistic regression incorporating a temperature parameter to learn feature thresholds for binarization, this is followed by unit weighting.

Results. We present the results of ProbChecklist and baselines in Table 3. Our performance is slightly lower than the MIP baseline. The highest accuracy is achieved by an MLP, but that comes at the cost of lower interpretability.

Model	Accuracy	Precision	Recall	Specificity	M	T
Dummy Classifier	37.226	0.372	1	0.628	-	-
MLP Classifier	64.962 ± 2.586	0.5726 ± 0.046	0.483 ± 0.074	0.76043 ± 0.0562	-	-
Logistic Regression	62.555 ± 1.648	0.624 ± 0.0461	0.144 ± 0.0393	0.9395 ± 0.0283	-	-
Unit Weighting	58.278 ± 3.580	0.521 ± 0.093	0.4386 ± 0.297	0.6861 ± 0.251	9.6 ± 0.8	3.2 ± 1.16
SETS Checklist	56.475 ± 7.876	0.517 ± 0.106	0.6639 ± 0.304	0.494 ± 0.3195	10 ± 0	6 ± 0.632
ILP mean thresholds	62.992 ± 0.82	0.544 ± 0.087	0.1196 ± 0.096	0.9326 ± 0.0623	4.4 ± 1.01	2.8 ± 0.748
MIP	63.688 ± 2.437	0.563 ± 0.050	0.403 ± 0.082	0.7918 ± 0.06	8 ± 1.095	3.6 ± 0.8
Concepts + LR	61.168 ± 1.45	0.565 ± 0.059	0.324± 0.15	0.805 ± 0.1	-	-
Ours	62.579 ± 2.58	0.61 ± 0.076	0.345 ± 0.316	0.815 ± 0.185S	10	3.6 ± 1.2

Table 3. Performance results for Sepsis Prediction task using Tabular Data.

4.3. Sepsis Prediction from Time Series Data

Data. We use the PhysioNet 2019 Early Sepsis Prediction (Reyna et al., 2019) for this task also. The preprocessing steps and formation of subsets are the same as described in Section 4.2. Instead of computing summary statistics from the clinical time series, we train different concept learners to capture the dynamics of the time series. This allows us to encapsulate additional information, such as sudden rise or fall in feature values as concepts.

The processed dataset contains 2272 training, 256 validation, 256 testing samples, and 56 time steps for each patient. Like the experimental setup in Section 4.2, we fix $\mathbf{K} = \mathbf{10}$, i.e. use the top ten out of thirty-four features based on logistic regression weights.

Architecture. We define $\mathbf{K}$ CNNs with two convolutional

layers which accept one-dimensional signals and convolve them into $\mathbf{d_k}$ concepts. Then we gather all the learnt concepts for a patient ($\mathbf{M} = \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{d_k}$) and dispatch them to the probabilistic logic module for training.

Baselines. The set of baselines are same as those in the previous task (Section 4.2). The checklist-specific architectures like Unit Weighting, SETS Checklist, ILP mean thresholds, and MIP Checklist lack the ability to process complex data modalities such as time series data, so we use feature mean values for training them. Lastly, for CNN + MLP, we employ the concept learner described above, which is followed by an MLP, and cross-entropy loss is calculated on the predicted class, i.e. $\mathbb{P}(\hat{\mathbf{y}} = \mathbf{1})$.

Results. We summarise the results for this task in Table 4. ProbChecklist performs remarkably as compared to the baselines. An increase in the recall values is observed as $\mathbf{d_k}$ increases.

d_k	Evaluation	Accuracy	Precision	Recall	Specificity	T	M
1	Logistic Regression	60.627 ± 1.379	0.4887 ± 0.106	0.1843 ± 0.073	0.8792 ± 0.048	-	-
	Unit Weighting	60.532 ± 1.567	0.4884 ± 0.087	0.1882 ± 0.102	0.8745 ± 0.066	5 ± 0.63	9.2 ± 0.748
	ILP mean thresholds	62.481 ± 0.426	0.529 ± 0.242	0.0529 ± 0.051	0.964 ± 0.031	3.4 ± 1.496	3.6 ± 1.85
	SETS Checklist	39.547 ± 1.53	0.3942 ± 0.014	0.989 ± 0.017	0.0085 ± 0.012	1.8 ± 2.23	9.8 ± 0.4
	MIP Checklist	60.767 ± 1.022	0.5117 ± 0.055	0.142 ± 0.05	0.912 ± 0.036	3.5 ± 0.866	6.5 ± 1.5
	CNN + MLP	63.465 ± 2.048	0.585 ± 0.05	0.234 ± 0.071	0.895 ± 0.021	-	-
1	**Ours**	63.716 ± 3.02	0.613 ± 0.12	0.233 ± 0.035	0.9 ± 0.036	3 ± 0.89	10
2	**Ours**	62.969 ± 3.355	0.582 ± 0.061	0.343 ± 0.176	0.817 ± 0.145	3 ± 1.095	20
3	**Ours**	63.671 ± 1.832	0.609 ± 0.115	0.354 ± 0.157	0.823 ± 0.108	4.4 ± 1.356	30

Table 4. Performance results for sepsis prediction task using time series.

4.4. Impact of the binarization scheme

In Tables 5 and 6 we difference in performance between two types of evaluation schemes, one is the checklist, calculated by binarizing the concept probabilities and thresholding the total number of positive concepts at $\mathbf{T}$, and the second is the model evaluation, which involves thresholding $\mathbb{P}(\mathbf{d} > \mathbf{T})$ to obtain predictions.

d_k	Evaluation	Accuracy	Precision	Recall	Specificity	T	M
1	Model	86.04 ± 0.463	0.814 ± 0.027	0.412 ± 0.021	0.976 ± 0.004	3	4
	Checklist	72.63 ± 1.42	0.427 ± 0.013	0.979 ± 0.005	0.661 ± 0.018		
2	Model	96.888 ± 0.064	0.925 ± 0.008	0.922 ± 0.012	0.981 ± 0.003	5	8
	Checklist	92.768 ± 1.6	0.752 ± 0.045	0.97 ± 0.006	0.917 ± 0.02		
3	Model	97.064 ± 0.187	0.94 ± 0.008	0.915 ± 0.013	0.985 ± 0.002	7	12
	Checklist	96.52 ± 0.33	0.9 ± 0.009	0.933 ± 0.008	0.973 ± 0.002		
4	Model	97.04 ± 0.177	0.937 ± 0.005	0.917 ± 0.006	0.984 ± 0.001	8.4 ± 1.2	16
	Checklist	**96.808 ± 0.24**	0.917 ± 0.015	0.929 ± 0.01	0.978 ± 0.004		
5	Model	97.032 ± 0.135	0.943 ± 0.005	0.91 ± 0.01	0.986 ± 0.001	9.4 ± 1.36	20
	Checklist	96.68 ± 0.269	0.918 ± 0.013	0.92 ± 0.009	0.979 ± 0.004		

Table 5. MNIST Checklist: Performance of ProbChecklist with varying $\mathbf{d_k}$

d_k	Evaluation	Accuracy	Precision	Recall	Specificity	T	M
1 **(Ours)**	Model	64.105 ± 1.69	0.586 ± 0.047	0.309 ± 0.025	0.857 ± 0.018	3 ± 0.89	10
	Checklist	63.716 ± 3.02	0.613 ± 0.12	0.233 ± 0.035	0.9 ± 0.036		
2 **(Ours)**	Model	62.656 ± 2.377	0.525 ± 0.025	0.565 ± 0.032	0.667 ± 0.027	3 ± 1.095	20
	Checklist	62.969 ± 3.355	0.582 ± 0.061	0.343 ± 0.176	0.817 ± 0.145		
3 **(Ours)**	Model	60.781 ± 2.09	0.503 ± 0.019	0.545 ± 0.026	0.648 ± 0.019	4.4 ± 1.356	30
	Checklist	63.671 ± 1.832	0.609 ± 0.115	0.354 ± 0.157	0.823 ± 0.108		

Table 6. Performance results for sepsis prediction task using time series.

As expected, the performance of the checklist is slightly lower than the model because concept weights are binarized leading to a more coarse-grained evaluation.

4.5. Interpreting the learnt concepts

We evaluate the concepts learnt for the MNIST synthetic dataset for $\mathrm{d}_k = 2$

5. Conclusion

Predictive checklists are widely used in the medical domain to assist complex decision-making and patient triaging. All the currently available approaches cannot process complex data modalities such as histopathological images, surgical videos, and multimodal datasets. In this work, we have proposed a mathematically-sound novel method, ProbChecklist, to learn checklists from abstract data modalities using the Probabilistic Logic Programming framework. We have benchmarked the performance of ProbChecklist on three prediction tasks, and it outperforms the baseline methods. This framework allows us to represent checklist learning as a set of predicates and probabilistic facts. Since concept learners can be designed for different modalities, our approach can be extended to a multimodal setting with ease. ProbChecklist can be quickly adopted by clinicians to learn checklists suited to their needs. Additional constraints and queries can be easily implemented to promote reliability and safety in a clinical setup.

The main limitation of the current approach is the high computational complexity of inference in probabilistic programming. The original formulation of ProbChecklist was exponential in the number of concepts learnt, which made experimentation infeasible. But we solved that by saving the combinations and performing vectorised operations, however, this increases memory utilization. There is a tradeoff between time and space complexity. In the ongoing work, we are investigating approximations to explore a lower number of combinations instead of all the possible worlds. This would make the method more adaptable. We are simultaneously working on interpreting and evaluating arbitrary learnt concepts. This involves experimenting with different neural architectures to enhance the capability of the proposed method.

References

Ahmad, M. A., Eckert, C., and Teredesai, A. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, pp. 559–560, 2018.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot,

Image 1 = {0,2,4,6,8}

Image 2 = {1,3,5,7,9}

Image 3 = {4,5,6}

Image 4 = {6,7,8,9}

A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

Bennetot, A., Laurent, J.-L., Chatila, R., and Díaz-Rodríguez, N. Towards explainable neural-symbolic visual reasoning. *arXiv preprint arXiv:1909.09065*, 2019.

Davenport, T. and Kalakota, R. The potential for artificial intelligence in healthcare. *Future healthcare journal*, 6 (2):94, 2019.

De Brouwer, E., Gonzalez, J., and Hyland, S. Predicting the impact of treatments over time with uncertainty aware neural differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 4705–4722. PMLR, 2022.

De Raedt, L. and Kimmig, A. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.

Du, M., Liu, N., and Hu, X. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1): 68–77, 2019.

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., and Dean, J. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

Friede, D. and Niepert, M. Efficient learning of discrete-continuous computation graphs. *Advances in Neural Information Processing Systems*, 34:6720–6732, 2021.

Futoma, J., Simons, M., Panch, T., Doshi-Velez, F., and Celi, L. A. The myth of generalisability in clinical research and machine learning in health care. *The Lancet Digital Health*, 2(9):e489–e492, 2020.

Ghassemi, M., Naumann, T., Schulam, P., Beam, A. L., Chen, I. Y., and Ranganath, R. A review of challenges and opportunities in machine learning for health. *AMIA Summits on Translational Science Proceedings*, 2020:191, 2020.

Hales, B., Terblanche, M., Fowler, R., and Sibbald, W. Development of medical checklists for improved quality of patient care. *International Journal for Quality in Health Care*, 20(1):22–30, 2008.

Haynes, A. B., Weiser, T. G., Berry, W. R., Lipsitz, S. R., Breizat, A.-H. S., Dellinger, E. P., Herbosa, T., Joseph, S., Kibatala, P. L., Lapitan, M. C. M., et al. A surgical safety checklist to reduce morbidity and mortality in a global population. *New England journal of medicine*, 360(5): 491–499, 2009.

Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.-F., Meek, C., Neville, J., et al. *Introduction to statistical relational learning*. MIT press, 2007.

Makhija, Y., De Brouwer, E., and Krishnan, R. G. Learning predictive checklists from continuous medical data. *arXiv preprint arXiv:2211.07076*, 2022.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31, 2018.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

Oldenhof, M., Arany, Á., Moreau, Y., and De Brouwer, E. Weakly supervised knowledge transfer with probabilistic logical reasoning for o-object detection. *International Conference on Learning Representations (ICLR)*, 2023.

Raedt, L. D., Kersting, K., Natarajan, S., and Poole, D. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning*, 10(2):1–189, 2016.

Reyna, M. A., Josef, C., Seyedi, S., Jeter, R., Shashikumar, S. P., Westover, M. B., Sharma, A., Nemati, S., and Clifford, G. D. Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019. In *2019 Computing in Cardiology (CinC)*, pp. Page–1. IEEE, 2019.

Rocktäschel, T. and Riedel, S. End-to-end differentiable proving. *Advances in neural information processing systems*, 30, 2017.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Zhang, H., Morris, Q., Ustun, B., and Ghassemi, M. Learning optimal predictive checklists. *Advances in Neural Information Processing Systems*, 34:1215–1229, 2021.

A. Derivations for the loss function

We should add the derivations here.