Neural Bayesian Network Understudy

Paloma Rabaey, Cedric De Boom, Thomas Demeester
IDLab, Dept. of Information Technology
Ghent University - imec
Ghent, Belgium
first.last@ugent.be

Abstract

Bayesian Networks may be appealing for clinical decision-making due to their inclusion of causal knowledge, but their practical adoption remains limited as a result of their inability to deal with unstructured data. While neural networks do not have this limitation, they are not interpretable and are inherently unable to deal with causal structure in the input space. Our goal is to build neural networks that combine the advantages of both approaches. Motivated by the perspective to inject causal knowledge while training such neural networks, this work presents initial steps in that direction. We demonstrate how a neural network can be trained to output conditional probabilities, providing approximately the same functionality as a Bayesian Network. Additionally, we propose two training strategies that allow encoding the independence relations inferred from a given causal structure into the neural network. We present initial results in a proof-of-concept setting, showing that the neural model acts as an understudy to its Bayesian Network counterpart, approximating its probabilistic and causal properties.

1 Introduction

Bayesian Networks (BNs) have been the subject of a large body of research, and are considered an important technology for high-stakes decision making, in particular in the healthcare domain. Kyrimi et al. [2021b] provide a clear overview of the key properties that make BNs highly appealing in that context: (i) their ability to model complex problems involving causal dependencies as well as uncertainty, (ii) their ability to combine data and expert knowledge, (iii) their interpretable graphical structure, and (iv) their ability to model interventions and reason diagnostically as well as prognostically. Despite efforts from the research community to support healthcare practitioners in using BNs [Arora et al., 2019, Kyrimi et al., 2020], their practical adoption remains very limited. The analysis of Kyrimi et al. [2021a] indicates data inadequacies as one of several key limiting factors: BNs have a limited capacity to address continuous data, and cannot directly deal with unstructured data such as images or free text. According to Tayefi et al. [2021], the value of information entailed by unstructured text fields in electronic medical records for clinical decision support cannot be underestimated.

In contrast to Bayesian networks, neural networks are well-suited for dealing with unstructured data. However, they are inherently not interpretable, which stands in the way of clinical adoption [Montani and Striani, 2019, Peiffer-Smadja et al., 2020]. Whereas Hinton [2018] argues strongly in favour of the adoption of neural networks in healthcare, he also recognises interpretability as a property that is desired by practitioners.

Our envisioned research goal is to build models that combine the advantages of both approaches: a fully neural model that can be trained on observations including unstructured data, but that also adopts the desirable properties of BNs as listed above. Realistically speaking, we can at best expect only approximate probabilistic reasoning capabilities from a generic neural network, and it will lack

NeurIPS 2022Workshop on Causal Machine Learning for Real-World Impact (CML4Impact 2022).

the direct interpretability of a BN. However, if the model displays the causal properties of a BN, the ideas of counterfactual explanation [Wachter et al., 2017] can be applied to explain its predictions.

This paper forms the first step along that outlined research track. We focus on the problem of training a neural network on a given set of discrete data samples as well as knowledge of a Directed Acyclic Graph (DAG) [Russell and Norvig, 2009] reflecting causal relationships between the variables. We do not yet aim to augment such models with unstructured data, but for now require that it behaves as a BN with the given DAG, if we were to train it on the same samples. In this sense, our model can be considered a *neural understudy* to the BN. We want to emphasise that our current preliminary work does not aim to build a neural network which outperforms its Bayesian network counterpart, but one that approximates its capabilities. After this necessary first step, we expect to reap the benefits of the neural network understudy by incorporating unstructured data. While the current work presents results only in a proof-of-concept setting, the possibility to reason under uncertainty, while combining causal structure and unstructured data, has great potential in real-world (clinical) applications.

In short, this paper makes the following contributions: (1) we propose a neural network model able to infer the probability of a set of target variables conditioned on observed evidence, (2) we introduce two approaches for injecting DAG knowledge while training the neural network, and (3) we present empirical results (including robustness analysis) in a small proof-of-concept setting. The necessary code to reproduce these results can be found in our Github repository.[1]

2 Related work

This section focuses on recent works most related to the proposed ideas; we do not aim to provide a broad overview in terms of related research fields. A basic premise for the presented work is the existence of causal knowledge in addition to observational data, to be injected into a neural network model. Our goal is therefore not *causal discovery*, intended to recover the causal mechanisms underlying the distribution from which the observed data was generated [Ke et al., 2019, Brouillard et al., 2020].

Geiger et al. [2022] present a method to align different parts of a neural network with nodes in a causal graph, resulting in a model with improved interpretability. Harradon et al. [2018] pursue a similar goal, summarising a neural model into a Bayesian causal model, to provide counterfactual explanations for the neural model. Instead of starting from a neural network and using causal models to understand or guide its inner workings, we work the other way round.

In Rohekar et al. [2018], conditional independencies in the input distribution are encoded hierarchically in the network structure, as a way of performing unsupervised neural network structure learning. Pawlowski et al. [2020] and Parafita and Vitrià [2020] present each parent-child relation in a given structural causal model as a deep neural network unit. While we don't explicitly align parts of the neural network with a causal model, we do include structural information in the training process as a regulariser. This is related to the work from Kyono et al. [2020], who propose a prediction task with a regularisation objective to detect causal relationships between features. It is not based on prior knowledge of causal relationships, as in our work.

Attempts have been made to build neural networks that implement belief propagation [Pearl, 1988] to improve on accuracy and efficiency [Garcia Satorras and Welling, 2021, Kuck et al., 2020]. Our proposed neural networks also aim at answering probabilistic queries, but can freely learn the inference mechanics from the data.

Another key related contribution is DeepProbLog [Manhaeve et al., 2018]. This neuro-symbolic model is able to implement our envisioned goal: its probabilistic logic programming core can represent any BN exactly, and its so-called neural predicates are able to encode unstructured data. However, whereas a fully neural model can at best reason only approximately with probabilities, it may have some benefits over the exact probabilistic reasoning component in DeepProbLog. Indeed, in future research we aim to model complex realistic observational data. Some of its properties a suitable model should be able to capture, may be hard to express in a probabilistic program.

[1] `https://github.com/prabaey/NBN-understudy`

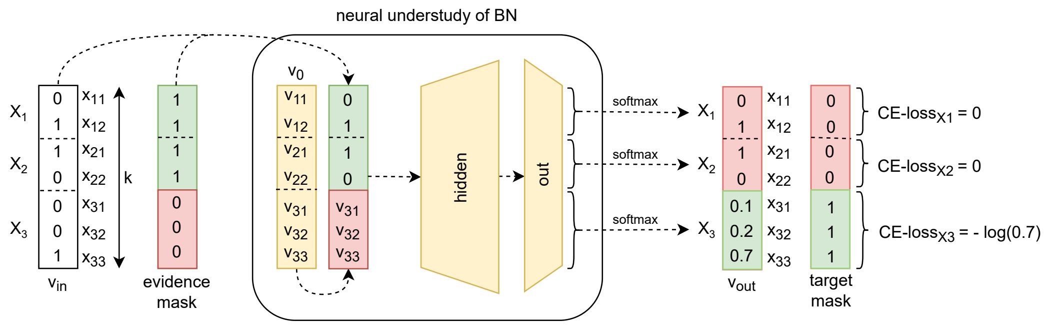


Figure 1: Illustration: training with evidence $\mathcal{E} = \{X_1, X_2\}$ and targets $\mathcal{T} = \{X_3\}$, for $P(X_3|X_1 = x_{12}, X_2 = x_{21})$. The mask at the input (output) indicates selection of evidence (target) entries.

Finally, our work is not to be confused with the so-called *Bayesian neural networks* MacKay [1992], aiming to obtain a probability distribution over neural networks. Instead, our goal is training a neural understudy for a Bayesian Network, i.e., a neural model that behaves similarly.

3 Training a neural understudy of a Bayesian network

The following paragraphs describe our proposed neural architecture with full flexibility in the choice of input and output variables (Section 3.1), followed by our proposed training strategies to inject causal structure into the model (Section 3.2).

3.1 Neural architecture and training

A fully specified BN can be queried by providing inputs to any selection of variables (henceforth called the *evidence*), after which the probabilities for each of the remaining variables (the *targets*) can be inferred, conditioned on the evidence. We will require the same ability from our neural counterpart, and show that this can be achieved by using complementary masks at its inputs and outputs. This only induces restrictions on the input and output layers but not on the internal neural architecture.

We focus on problems involving a set of N discrete variables $\mathcal{V} = \{X_1, X_2, ..., X_N\}$, in which X_i can take n_i possible values in $\mathcal{X}_i = \{x_{i1}, ..., x_{in_i}\}$. Given a set $\mathcal{E} \subset \mathcal{V}$ of m evidence variables ($m < N$), each with an assigned value $\{E_1 = e_1, ..., E_m = e_m\}$ (for convenience written as $\mathcal{E} = e$), our goal is to train a neural network to predict the probability distribution $P(X|\mathcal{E} = e)$ for each of the remaining variables X. The latter are called the target variables, aggregated in the set $\mathcal{T} = \mathcal{V} \setminus \mathcal{E}$.

After training on an observed set of individual samples of the form $\{X_1 = x_1, \ldots, X_N = x_N\}$, our model should be able to deal with any selection of evidence variables. The key idea to achieve that is through dynamic masking (i.e., a different mask for any variable split $\mathcal{E}$ vs. $\mathcal{T}$): all variables are present at the model input and output, but target variables are filtered out of the inputs through an evidence mask, whereas predictions for the evidence variables are filtered from the outputs through a target mask. This is illustrated in Fig. 1.

Model architecture: Each input to the model is a vector v_{in} of dimension $k = \sum_{i=1}^{N} n_i$, and is formed by concatenating the n_i-sized one-hot representation of the value held by each of the N variables X_i ($i = 1, \ldots, N$). The masked positions of the target variables in v_{in} are substituted by the corresponding components of a static vector v_0. The input v_{in} is subsequently passed through one or more hidden layers, resulting at the output in a k-dimensional vector of logits. These are then locally normalised through softmax activations, for each variable X_i considering its corresponding n_i entries. The output vector v_{out} is finally constructed by replacing the entries at the positions of evidence variables, by the corresponding one-hot representations from v_{in}. The values corresponding to the target variables can be interpreted as the predicted target distributions given the evidence. This is illustrated in Fig. 1. Note that the k-dimensional vector v_0 is obtained by applying the linear output layer with per-variable softmax normalisation on a randomly initialised trainable vector.

Model training: Whereas a BN is naturally able to answer any query once its conditional probability tables are specified, we need to explicitly train our neural network for that ability. To that end, we iterate over the observed samples, randomly dividing the variables into evidence set $\mathcal{E}$ and targets $\mathcal{T}$, and minimise the average cross-entropy loss for the target variables. The training loss can be expressed as $\mathcal{L}^T = \frac{1}{|\mathcal{T}|} \sum_{X_i \in \mathcal{T}} -\log \hat{p}_{ij}$, in which $\hat{p}_{ij}$ denotes the entry in v_{out} corresponding to the actual observed input value x_{ij} of target variable X_i, i.e., the predicted output probability for the target class of X_i.

Even though the model only receives discrete data samples and therefore never observes the class probabilities it is meant to predict, this training strategy steers the predicted probabilities towards the empirical (conditional) class probabilities. It only requires that each sample's frequency of occurrence during training corresponds to its relative frequency in the set of observations. Appendix A.1 proves this for the canonical case of two binary variables, but the presented derivation can be extended to the multi-variable case.

3.2 Training with causal structure

We now propose the following two training strategies to augment the neural network with knowledge of the causal structure between the variables. They are based on the assumption that the causal knowledge can be represented as a DAG, reflecting the (conditional) independence relations that exist between the variables.

1. Injecting independence relations through regularisation (REG): In general, we can say a given DAG implies M pairwise conditional independence relations between two variables, conditioned on a set of observed variables, shortly written as $X \perp Y \mid \mathcal{A}$, with $X, Y \in \mathcal{V}$ and the conditioning set $\mathcal{A} \subset \mathcal{V} \setminus \{X, Y\}$. The first proposed training strategy is to inject these independence relations by constructing regularisation loss terms $\mathcal{L}^{\text{REG}}_{X \perp Y \mid \mathcal{A}}$ that quantify how strongly the model violates them:

$$\mathcal{L}^{\text{REG}}_{X \perp Y \mid \mathcal{A}} = \frac{1}{n} \sum_{j=1}^{n} \big(\hat{p}(X = x_j \mid \mathcal{A} = a, Y = y) - \hat{p}(X = x_j \mid \mathcal{A} = a, Y = y')\big)^2 \tag{1}$$

in which the summation runs over all n possible values of X. The model's predicted probability for $X = x_j$ with as evidence the assignment $\mathcal{A} = a$ and $Y = y$, is denoted as $\hat{p}(X = x_j \mid \mathcal{A} = a, Y = y)$. The loss term expresses that if the independence relation were satisfied by the model, its predicted probability for any value of X, given any evidence $\mathcal{A} = a$, should be independent of the value assumed by Y. The conditioning set $\mathcal{A}$ is randomly instantiated every time the corresponding loss term applies during training, and the values y and y' of Y are chosen randomly (with $y \neq y'$).

During training, the regular loss $\mathcal{L}^T$ per data sample is augmented with the regularisation loss $\mathcal{L}^{\text{REG}}$ of one sampled independence relation, weighted with a hyperparameter α.

2. Injecting independence relations through evidence corruption (COR): The second training strategy is based on the intuition that for particular observed data samples, the value of some of the evidence variables may no longer matter besides the other observed variables, when accounting for the relevant independence relations. During training, we detect these cases, and randomly corrupt (i.e., re-sample) those values in the evidence presented to the network.

Consider training on a particular observed sample, the variables divided into evidence $\mathcal{E}$ and targets $\mathcal{T}$. We then go through the known independence relations $X \perp Y \mid \mathcal{A}$, to see which ones are relevant to the training instance. This is the case if either of the following two conditions hold: (1) $(\{Y\} \cup \mathcal{A}) = \mathcal{E}$ and $X \subset \mathcal{T}$, or (2) $(\{X\} \cup \mathcal{A}) = \mathcal{E}$ and $Y \subset \mathcal{T}$. When condition (1) holds, the predicted outcome of target X should not depend on the observed value of Y, as prescribed by the independence relation, since the conditioning set that is needed for this relation to hold is indeed part of the evidence. The observed input value for Y can hence be disregarded, and we randomly assign a new value from its possible classes. Such a corruption of the input will condition the model to ignore Y when predicting X, given that all variables in $\mathcal{A}$ are also provided as evidence. Note that predictions for other variables in $\mathcal{T}$ need to be done based on the original (i.e., non-corrupted) value for Y, as there is no guarantee that the same independence relation holds for those targets as well. We apply a similar reasoning when condition (2) holds, now corrupting the value of X instead of Y. If no relevant independence relation is found for a given selection of evidence and targets, the input sample is not corrupted and passed as-is to the model. In Appendix A.2, we consider the basic two-variable case and show that the corruption strategy leads to the desired predicted probabilities.

4 Neural understudy of Bayesian network: proof-of-concept

The experiments presented in this section aim at answering the following research questions:

- **RQ1**: How does the basic neural network perform in comparison with a Bayesian Network trained on the same set of samples, in terms of accuracy in predicted probabilities and in terms of sample efficiency? (Section 4.2)
- **RQ2**: What is the effect of injecting causal structure knowledge into our neural network on its performance? (Section 4.2)
- **RQ3**: How does each model's performance change when faced with a partially incorrect specification of the causal structure? (Section 4.3)

To this end, we compare the following models, trained on an artificial dataset of samples generated from a given joint distribution, as specified in the next section:

- **Bayesian network baseline (BN)**: Bayesian Network with the correct DAG, its joint probability distribution estimated from the training samples using Maximum Likelihood Estimation with a K2 prior (see Appendix A.3).
- **Neural network baseline (NN)**: Basic neural network approach as presented in Section 3.1, implemented in a single-layer feed-forward model (see Appendix A.4 for details).
- **NN with DAG-based regularisation (NN+REG)**: This model extends the basic NN model with independence relation information, through the regularisation technique REG (Section 3.2).
- **NN with DAG-based corruption (NN+COR)**: This model, also based on NN, injects independence relations during training with the corrupted inputs strategy introduced in Section 3.2.

4.1 Evaluation

For the experiments in this section, we make use of the "Asia" network based on [Lauritzen and Spiegelhalter, 1988], as depicted in Fig. 2. Despite its simplicity, with only 7 binary variables, it has sufficient connectivity between the variables to make the problem non-trivial. Following the rules of D-separation [Verma and Pearl, 1990], we can in total extract 191 unique independence relations from its DAG, for the training strategies NN+REG and NN+COR.

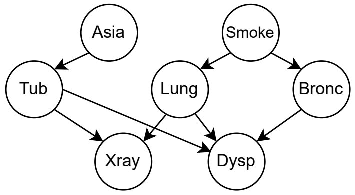


Figure 2: The "Asia" BN model, describing a basic lung cancer detection system (for details, see Appendix A.3).

From the ground-truth "Asia" model (detailed in Appendix A.3), we can draw data samples to use as training instances. Through variable elimination [Russell and Norvig, 2009], we can build test queries with the ground-truth conditional probabilities of target variables, for any particular assignment of evidence variables. Just like during training, this means that one query may contain multiple target variables. We can then evaluate models by iterating through the test queries, and calculating the mean absolute error (MAE) between the predicted target distribution and the desired one. For each query, we compute the MAE per target variable, sum all their contributions and divide by the number of targets in the query. We use two different test sets. The first set contains all possible evidence assignments, in total 2,059 queries, and leads to the **total MAE**. The second set contains 1,000 queries, each constructed by drawing a sample from the joint probability distribution, then randomly selecting the set of evidence variables, and obtaining the conditional probabilities for the target variables. It measures the **sample MAE**, assigning more weight to the model's performance for queries with commonly observed evidence values, while the total MAE puts equal focus on common as well as rare assignments of the evidence variables.

4.2 Performance of Bayesian network vs. neural understudy

In Fig. 3 we compare the sample efficiency of our 4 models in terms of total and sample MAE. For the NN+REG model, we set the regularisation parameter α to 10. The training hyperparameters (hidden layer dimensions, batch size, learning rate, number of training epochs) are kept fixed for all neural models to allow for a direct comparison. Details on the training process and hyperparameters are given in Appendix A.4. The results allow us to answer the first two research questions.

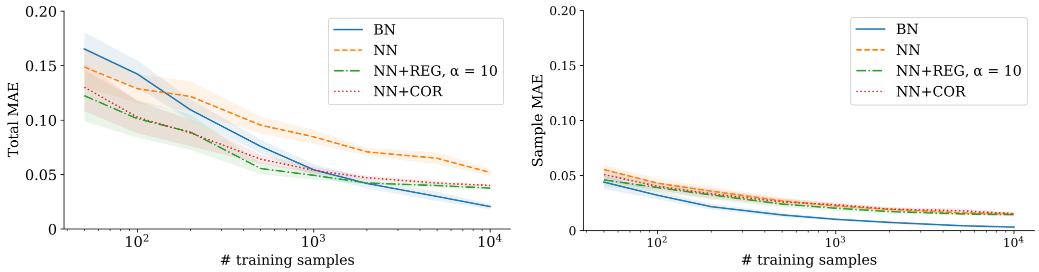


Figure 3: Comparison of 4 models in terms of total MAE and sample MAE for different sizes of the training set. The lines reflect the mean across 10 random seeds, used to sample the training set and initialise the neural networks. The shading represents the 95% confidence interval.

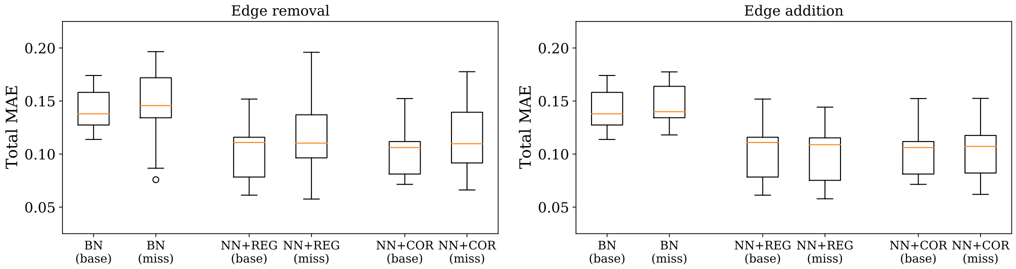


Figure 4: Boxplots visualising the total MAE for models receiving the correct DAG ("base") vs. a partially miss-specified DAG ("miss"), i.e., either one edge removed (left) or added (right), for different models (BN; NN+REG; NN+COR) trained on 100 observed samples. All train runs are done for 5 random seeds, and for "miss" runs, 5 random DAG corruptions are done (resulting in 25 runs per model for the "miss" setting).

RQ1 (Sample efficiency) The proposed training strategy leads to neural understudy models that can approximately infer conditional probabilities. For smaller training sets, the basic NN model shows similar performance as the BN baseline in terms of total MAE. As the training set grows larger, the BN model's knowledge of the DAG structure allows it to significantly overtake the basic NN model in terms of performance. The BN baseline outperforms the neural models across the board when looking at sample MAE, though the gap is not large.

RQ2 (Effect of including DAG info) Including information on the DAG structure in the neural model improves its performance, with a more visible effect on the total MAE. In terms of that metric, injecting the independence relations during training allows the neural model to outperform the BN baseline for smaller sample sets. The NN+REG model performs slightly better than the NN+COR model, albeit not significantly. We hypothesise that this stems from the REG technique systematically iterating over all conditional independence relations, whereas the COR technique only applies a relation when the randomly sampled evidence forms a match. The strength of the neural models appears to lie in their improved performance for rare evidence combinations, which are more heavily disadvantaged in smaller datasets.

4.3 Robustness against miss-specification of causal structure

We now explore the impact of injecting partially incorrect information on the causal structure. To this end, we create 10 new DAGs, by randomly removing or adding [2] one edge at a time to the ground-truth DAG from Fig. 2. While still being trained on samples from the correct "Asia" network, the BN, NN+REG and NN+COR models now get their conditional independence relations from a partially incorrect DAG. The results displayed in Fig. 4 allow answering the third research question.

RQ3 (Partially incorrect DAG) While the inclusion of DAG information in the neural understudy brings clear additional benefits in terms of total MAE compared to its most basic form, we must be careful to correctly specify this causal structure. When we assume two variables to be independent when they are not (i.e. by removing an edge in the DAG), both the Bayesian baseline and the neural

[2] We were careful not to add edges which might introduce cycles in the network.

models become less stable, showing higher variation across sample sets. Adding an edge in the DAG seems to do no harm, since this makes for a more conservative estimate of the DAG structure (we assume two variables to have some dependency when they do not).

5 Conclusions and future work

We presented ideas for building neural networks that behave like Bayesian Networks. In future research we aim to combine the benefits of neural networks (i.e., encoding unstructured data) with some advantages of Bayesian networks, such as their ability to combine uncertainty with causal structure. As a first contribution in that direction, we presented a method to learn a neural network model on observational sample data, and two strategies to encode known causal relationships between the variables, by injecting independence relations.

We tested our method on a single small-scale example, and saw that our proposed training strategy generally works: our neural understudy models (w.r.t. to their Bayesian Network counterpart trained on the same samples) were able to make approximate predictions of conditional probabilities. The inclusion of causal structure resulted in similar performance of the neural understudy compared to the BN baseline, with the neural understudy slightly outperforming the BN under low-sample regimes when testing on all possible queries. We saw the performance of the NN models become less stable as soon as incorrect independence relations were injected, though this behaviour was also observed for the BN.

We see multiple avenues for future work. We will first extend our experiments to larger, more realistic settings, to see whether the conclusions based on our small example still hold. We then aim to investigate how our models can be extended towards continuous variables as well as unstructured data, to be able to answer probabilistic queries concerning a combination of any of these inputs. For example, we envision the use of pre-trained language encoders, although that will require considerable adaptation of the straightforward model with corresponding discrete nodes at input and output.

References

Ankur Ankan and Abinash Panda. pgmpy: Probabilistic Graphical Models using Python. In *Proceedings of the 14th Python in Science Conference*, pages 6 – 11, 2015.

Paul Arora, Devon Boyne, Justin J. Slater, Alind Gupta, Darren R. Brenner, and Marek J. Druzdzel. Bayesian networks for risk prediction using real-world data: A tool for precision medicine. *Value in Health*, 22(4), 2019.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, volume 33, pages 21865–21877. Curran Associates, Inc., 2020.

Víctor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 685–693. PMLR, 2021.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7324–7338. PMLR, 2022.

Michael Harradon, Jeff Druce, and Brian E. Ruttenberg. Causal learning and explanation of deep neural networks via autoencoded activations, 2018. URL `http://arxiv.org/abs/1802.00541`.

Geoffrey Hinton. Deep Learning—A Technology With the Potential to Transform Health Care. *Journal of the American Medical Association (JAMA)*, 320(11):1101–1102, 2018.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Bernhard Schölkopf, Michael C. Mozer, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions, 2019. URL `https://arxiv.org/abs/1910.01075`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Jonathan Kuck, Shuvam Chakraborty, Hao Tang, Rachel Luo, Jiaming Song, Ashish Sabharwal, and Stefano Ermon. Belief propagation neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 667–678. Curran Associates, Inc., 2020.

Trent Kyono, Yao Zhang, and Mihaela van der Schaar. Castle: Regularization via auxiliary causal graph discovery. In *Advances in Neural Information Processing Systems*, volume 33, pages 1501–1512. Curran Associates, Inc., 2020.

Evangelia Kyrimi, Mariana Raniere Neves, Scott McLachlan, Martin Neil, William Marsh, and Norman Fenton. Medical idioms for clinical bayesian network development. *Journal of Biomedical Informatics*, 108, 2020.

Evangelia Kyrimi, Kudakwashe Dube, Norman Fenton, Ali Fahmi, Mariana Raniere Neves, William Marsh, and Scott McLachlan. Bayesian networks in healthcare: What is preventing their adoption? *Artificial Intelligence in Medicine*, 116, 2021a.

Evangelia Kyrimi, Scott McLachlan, Kudakwashe Dube, Mariana R. Neves, Ali Fahmi, and Norman Fenton. A comprehensive scoping review of bayesian networks in healthcare: Past, present and future. *Artificial Intelligence in Medicine*, 117, 2021b.

S. L. Lauritzen and D. J. Spiegelhalter. Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 50(2):157–224, 1988.

David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Stefania Montani and Manuel Striani. Artificial intelligence in clinical decision support: a focused literature survey. *Yearbook of Medical Informatics*, 28(1):120–127, 2019.

Álvaro Parafita and Jordi Vitrià. Causal inference with deep causal graphs, 2020. URL `https://arxiv.org/abs/2006.08380`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc., 2019.

Nick Pawlowski, Daniel Coelho de Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. In *Advances in Neural Information Processing Systems*, volume 33, pages 857–869. Curran Associates, Inc., 2020.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

N. Peiffer-Smadja, T.M. Rawson, R. Ahmad, A. Buchard, P. Georgiou, F.-X. Lescure, G. Birgand, and A.H. Holmes. Machine learning for clinical decision support in infectious diseases: a narrative review of current applications. *Clinical Microbiology and Infection*, 26(5):584–595, 2020.

Raanan Y. Rohekar, Shami Nisimov, Yaniv Gurwicz, Guy Koren, and Gal Novik. Constructing deep neural networks by bayesian network structure learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3051–3062, Red Hook, NY, USA, 2018. Curran Associates Inc.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach.* Pearson, 3 edition, 2009.

Maryam Tayefi, Phuong Ngo, Taridzo Chomutare, Hercules Dalianis, Elisa Salvi, Andrius Budrionis, and Fred Godtliebsen. Challenges and opportunities beyond structured data in analysis of electronic health records. *Wiley Interdisciplinary Reviews: Computational Statistics*, 13, 02 2021.

Thomas Verma and Judea Pearl. Causal networks: Semantics and expressiveness. In *Uncertainty in Artificial Intelligence*, volume 9 of *Machine Intelligence and Pattern Recognition*, pages 69–76. North-Holland, 1990.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

A Appendix

A.1 Sample-based training of neural network

We provide a small-scale proof of how optimising the cross-entropy loss for the observed samples leads to probabilistic outcomes. While the illustrative setting for which we provide the proof only concerns two binary variables, the proof can be extended to a more general setting.

Table 1: Absolute frequency table showing the occurrence of $X = x$ and $Y = y$ in the hypothetical training set ($\nu = n_{00} + n_{01} + n_{10} + n_{11}$).

x	y	freq.
0	0	n_{00}
0	1	n_{01}
1	0	n_{10}
1	1	n_{11}

Assume we have two binary variables, and observe ν samples $\{X = x, Y = y\}$ during every training epoch. The distribution of the training data is shown in Table 1. Our model simultaneously optimises three partial training losses, each corresponding to a possible evidence mask: $\mathcal{L}_{X\to Y}$ (target Y, evidence X), $\mathcal{L}_{Y\to X}$ (target X, evidence Y), $\mathcal{L}_{XY}$ (target X and Y, no evidence). The random evidence/target mask decides which loss is optimised during which iteration, and the sum of these three losses forms the overall training objective. We will calculate the optimum for each partial training loss and show that this leads to the desired predictions for the targets at hand.

Evidence X, target Y: Optimising the partial loss $\mathcal{L}_{X\to Y}$ should lead us to an estimate for query $P(Y = y|X = x)$. The model returns a prediction $\hat{y}_{x\to y}$ for target $Y = y$ taking evidence $X = x$ as an input. Taking into account the frequencies of each sample as listed in Table 1, we obtain the partial loss $\mathcal{L}_{X\to Y}$ as shown in eq. (3). Here, we use the definition of binary cross-entropy loss as listed in eq. (2) and simplify by filling in the possible values (0 or 1) for target y.

$$\mathcal{L}_{\mathcal{CE}} = -ylog(\hat{y}) - (1-y)log(1-\hat{y}) \tag{2}$$

$$\begin{aligned}\mathcal{L}_{X\to Y} &= -n_{01}log(\hat{y}_{0\to 1}) - n_{00}log(\hat{y}_{0\to 0}) - n_{11}log(\hat{y}_{1\to 1}) - n_{10}log(\hat{y}_{1\to 0}) \\ &= -n_{01}log(\hat{y}_{0\to 1}) - n_{00}log(1-\hat{y}_{0\to 1}) - n_{11}log(\hat{y}_{1\to 1}) - n_{10}log(1-\hat{y}_{1\to 1})\end{aligned} \tag{3}$$

When given $X = 0$ as evidence, we will optimise this loss for $\hat{y}_{0\to 1}$. Equation (4) illustrates how calculating the partial derivative and setting it to zero, leads to the optimum for $\hat{y}_{0\to 1}$ (whereby $\hat{y}_{0\to 0}$ = 1 - $\hat{y}_{0\to 1}$). A similar derivation for $X = 1$ leads to the optimum for $\hat{y}_{1\to 1}$ (and $\hat{y}_{1\to 0}$ = 1 - $\hat{y}_{1\to 1}$). In other words, the predicted value for target Y moves towards its relative frequency in the training set, conditioned on the observed evidence values.

$$\frac{\partial\mathcal{L}_{X\to Y}}{\partial\hat{y}_{0\to 1}} = 0 \Rightarrow \hat{y}_{0\to 1} = \frac{n_{01}}{n_{00}+n_{01}}; \quad \frac{\partial\mathcal{L}_{X\to Y}}{\partial\hat{y}_{1\to 1}} = 0 \Rightarrow \hat{y}_{1\to 1} = \frac{n_{11}}{n_{10}+n_{11}} \tag{4}$$

Evidence Y, target X: Optimising the partial loss $\mathcal{L}_{Y\to X}$ should lead us to an estimate for query $P(X = x|Y = y)$. The derivation is symmetrical to the previous case, with the roles of X and Y switched.

No evidence, targets X and Y: By optimising $\mathcal{L}_{XY}$, the model jointly optimises its predicted outcome for queries $P(X = x)$ and $P(Y = y)$. A training case only contributes to this partial loss when the evidence mask chooses both variables as targets. Now, we can write the loss as in eq. (5), where $\hat{x}$ is the prediction for $X = 1$ given no evidence, and analogous for $\hat{y}$. We use the same notation for the relative frequencies as before, and again use the definition of binary CE-loss from eq. (2).

$$\begin{aligned}\mathcal{L}_{XY} = &-(n_{10} + n_{11})log(\hat{x}) - (n_{00} + n_{01})log(1 - \hat{x}) \\ &-(n_{01} + n_{11})log(\hat{y}) - (n_{00} + n_{10})log(1 - \hat{y})\end{aligned} \tag{5}$$

Again, we can optimise the loss above for $\hat{x}$ and $\hat{y}$. This leads to the optima shown in eq. (6). The optimum for $\hat{x}$ simply corresponds to the relative frequency of seeing $X = 1$ in the training set, which is indeed what we want as a prediction for $P(X = 1)$. The same goes for $\hat{y}$.

$$\frac{\partial \mathcal{L}_{XY}}{\partial \hat{x}} = 0 \Rightarrow \hat{x} = \frac{n_{10} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}}; \frac{\partial \mathcal{L}_{XY}}{\partial \hat{y}} = 0 \Rightarrow \hat{y} = \frac{n_{01} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}} \tag{6}$$

We emphasise that the results above only hold if training happens uniformly over the available training instances, so that the frequency of occurrence of a target given a particular evidence actually corresponds with the empirical probability in the training data. We ensure that this is the case by properly shuffling our batches within each epoch.

A.2 Injecting independence relations through evidence corruption

We again consider the simple setting of two binary variables X and Y, to show that the NN+COR method works as expected. We now add the knowledge that X and Y are independent. The total loss is made up of $\mathcal{L}_{X\to Y}$, $\mathcal{L}_{Y\to X}$ and $\mathcal{L}_{XY}$, as defined in Section A.1. Since the DAG-based corruption is only executed when the evidence set is not empty, only the partial losses $\mathcal{L}_{X\to Y}$ and $\mathcal{L}_{Y\to X}$ are affected. We zoom in on how to adapt the first one according to this new setting and how this affects the predicted outputs. The derivation for the other partial loss is symmetrical.

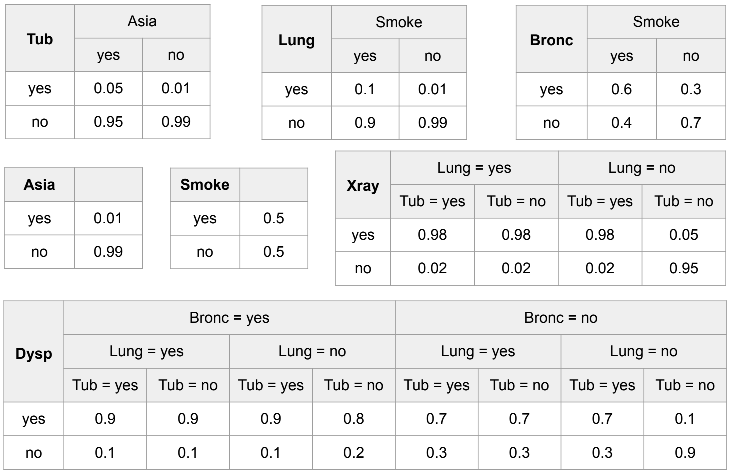
Say we receive a sample $\{X = x, Y = y\}$ during training and the mask indicates that X is evidence, while Y is the target. Since $X \perp Y$, the value of the target y should be independent of the observed evidence value x. Therefore, we corrupt the value of x, setting it to 1 with probability γ and to 0 with probability $1 - \gamma$. We can use the frequencies from Table 1 to write out the contribution of each training sample to the partial loss $\mathcal{L}_{X\to Y}$, taking into account that the evidence is corrupted for a fraction of the training samples. This is depicted in eq. (7). The predicted targets $\hat{y}_{0\to1}$ and $\hat{y}_{1\to1}$ are defined in the same way as described in Section A.1.

$$\begin{aligned}\mathcal{L}_{X\to Y} = &- (1-\gamma)(n_{01} + n_{11})log(\hat{y}_{0\to1}) - (1-\gamma)(n_{00} + n_{10})log(1 - \hat{y}_{0\to1}) \\ &- \gamma(n_{11} + n_{01})log(\hat{y}_{1\to1}) - \gamma(n_{10} + n_{00})log(1 - \hat{y}_{1\to1})\end{aligned} \tag{7}$$

When taking the partial derivative of the loss as shown in eq. (8), we get the optima for $Y = 1$ with either value of X as evidence. Due to the corruptions we implemented in the training process, we now get $\hat{y}_{0\to1}$ = $\hat{y}_{1\to1}$. The prediction for Y is indeed independent of the value of X (in accordance to $X \perp Y$) and simply equal to the relative frequency of observing $Y = 1$ in the training set.

$$\begin{aligned}\frac{\partial \mathcal{L}_{X\to Y}}{\partial \hat{y}_{0\to1}} = 0 \Rightarrow \hat{y}_{0\to1} = \frac{n_{01} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}} \\ \frac{\partial \mathcal{L}_{X\to Y}}{\partial \hat{y}_{1\to1}} = 0 \Rightarrow \hat{y}_{1\to1} = \frac{n_{01} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}}\end{aligned} \tag{8}$$

Note that the parameter γ plays no role in the optimum for $\hat{y}$. It does not matter according to which distribution we corrupt the evidence. In our implementation, we sample uniformly over all possible classes for the variable in question to corrupt its value. We could also opt to pull a random sample from the training set and switch out the value of the evidence variable to its value in this sample.



Tub	Asia	
	yes	no
yes	0.05	0.01
no	0.95	0.99

Lung	Smoke	
	yes	no
yes	0.1	0.01
no	0.9	0.99

Bronc	Smoke	
	yes	no
yes	0.6	0.3
no	0.4	0.7

Asia	
yes	0.01
no	0.99

Smoke	
yes	0.5
no	0.5

Xray	Lung = yes		Lung = no	
	Tub = yes	Tub = no	Tub = yes	Tub = no
yes	0.98	0.98	0.98	0.05
no	0.02	0.02	0.02	0.95

Dysp	Bronc = yes				Bronc = no			
	Lung = yes		Lung = no		Lung = yes		Lung = no	
	Tub = yes	Tub = no	Tub = yes	Tub = no	Tub = yes	Tub = no	Tub = yes	Tub = no
yes	0.9	0.9	0.9	0.8	0.7	0.7	0.7	0.1
no	0.1	0.1	0.1	0.2	0.3	0.3	0.3	0.9

Figure 5: Conditional probability tables defining the ground-truth "Asia" Bayesian network, based on [Lauritzen and Spiegelhalter, 1988].

A.3 Bayesian network implementation

We use the *pgmpy* Python library [Ankur Ankan and Abinash Panda, 2015] (version 0.1.17) for sampling, training and inference in our Bayesian networks.

To train our Bayesian networks from observed samples, we use the Maximum Likelihood Estimator. This estimator studies the co-occurrence of particular values of each variable and its parents in the training set, filling up the CPTs as such. We use a K2 prior as a smoothing strategy, to counteract the extremely skewed probability distributions that might appear in the CPTs when particular combinations of variables are never observed in the training set.

Artificial samples are generated from the ground-truth Bayesian network using the method of forward sampling with a particular seed. As a ground-truth Bayesian model, from which our artificial training and test sets are created, we use the "Asia" model (see Fig. 2 for its DAG structure). The CPTs that define its joint distribution with 20 parameters are shown in Fig. 5.

We use the technique of variable elimination to perform exact inference on our Bayesian networks. There are some cases where this method fails because the query contains some evidence combination it has never seen before. When coming across such a case during test time, we simply throw out the evidence for this particular query and take $P(X)$ as an estimate. We believe this makes for a fairer comparison than simply ignoring those queries, since our NN is in fact able to provide an estimate for all queries, even if it has never seen a particular evidence set before.

A.4 Neural network training details

Our neural network architecture and its training procedure are implemented in *Pytorch* [Paszke et al., 2019].

As shown in Fig. 1, our neural network is made up of 3 layers. The network receives an input vector of dimension k. In our "Asia" example, k is equal to 14 (all 7 variables have 2 classes). First, the values of the target variables (as selected by the evidence mask) are substituted by their respective value from v_0. The full input is then transformed to dimension h, using an input-to-hidden linear layer with tanh activation. Then, a hidden-to-hidden linear layer of size h, again with tanh activation is applied. Finally, the hidden-to-output layer transforms the activations back to dimension k. This layer applies N softmax functions to transform the activations belonging to each variable separately into normalised probability values, as depicted in Fig. 1.

For initialisation of the input vector on the positions of the target variables, we use the vector v_0 of size k, obtained by applying the hidden-to-output layer to a trainable vector of size h, followed by the per-variable softmax normalisation. When no evidence is present, v_0 serves as the full input vector, and leads to the model predicting the empirical mean probability for all variables.

We chose a hidden size h of 50, since we noticed this allowed enough flexibility while still constraining the parameter space sufficiently to avoid overfitting. With these dimensions, our network has 4014 trainable parameters in total. We use the Adam optimiser [Kingma and Ba, 2015] with a learning rate of 0.001 and otherwise default parameters in Pytorch. The training samples are fed to the neural model in batches of size 16, and the model is trained for 500 epochs. In the NN+REG model, we additionally use a regularisation batch size of 16.

The hyperparameter α controls the trade-off between training and regularisation loss. The MSE naturally has a smaller order of magnitude than the cross-entropy loss, therefore rather large choices for α (10, 100, 1000) work best. We settled for α equal to 10 since this seemed to lead to the best performance of the NN+REG method, though we did not observe a big difference between any choice of α within the range of 1 to 1000.