Annals of Operations Research
https://doi.org/10.1007/s10479-022-04692-6

ORIGINAL RESEARCH

Multicriteria interpretability driven deep learning

Marco Repetto[1,2]

Accepted: 28 March 2022

Abstract
Deep Learning methods are well-known for their abilities, but their interpretability keeps them out of high-stakes situations. This difficulty is addressed by recent model-agnostic methods that provide explanations after the training process. As a result, the current guidelines' requirement for "interpretability from the start" is not met. As a result, such methods are only useful as a sanity check after the model has been trained. In an abstract scenario, "interpretability from the start" implies imposing a set of soft constraints on the model's behavior by infusing knowledge and eliminating any biases. By inserting knowledge into the objective function, we present a Multicriteria technique that allows us to control the feature effects on the model's output. To accommodate for more complex effects and local lack of information, we enhance the method by integrating particular knowledge functions. As a result, a Deep Learning training process that is both interpretable and compliant with modern legislation has been developed. Our technique develops performant yet robust models capable of overcoming biases resulting from data scarcity, according to a practical empirical example based on credit risk.

Keyword Artificial intelligence · Machine learning · Deep learning · Multiple objective programming · Multicriteria optimization

1 Introduction

Deep Learning (DL) models are widely employed currently in a variety of industries, including self-driving cars (Rao & Frtunikj, 2018), brain-computer interfaces (Zhang et al., 2019), and gaming (Vinyals et al., 2019). DL approaches have become more accessible thanks to recent software and technology, allowing scholars and practitioners to use them in various disciplines. On the software side, current frameworks such as Tensorflow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019) have made it possible to create DL models without the need for ad-hoc compilers, as Lecun et al. (1990) did. On the hardware side, the cost of the necessary gear to train such models has decreased, allowing many people to create and deploy complex Neural Networks for very little money (Zhang et al., 2018). Apart from computer science,

✉ Marco Repetto
marco.repetto@unimib.it

1 Siemens Italy, Digital Indutries, Milan, Italy

2 Department of Economics, Management and Statistics, University of Milano-Bicocca, Milan, Italy

the democratization of such strong technology benefited many other disciplines. Economics (Nosratabadi et al., 2020) and Finance (Ozbayoglu et al., 2020) are two of those that benefited the most. Governments are interested in DL applications because they are concerned about potential social consequences. However, when it comes to training, these models demand more attentiveness, especially in high-stakes judgements (Rudin, 2019). To counteract these negative consequences, governments created a number of regulatory requirements, and the law began to expand on the right to explanation concept (Dexe et al., 2020). Scholars have been constructing post-hoc interpretation methods in the aim to build interpretable but DL grounded models. These techniques, on the other hand, are at conflict with newer standards, which demand "interpretability from the beginning" (Commission, 2019). Another concern is that such methods rely solely on interpretation after a model has been trained, preventing the input of prior data or the removal of biases. This research focuses on assuring the interpretability of DL models from the start by injecting knowledge and examining their potential in empirical scenarios such as credit risk prediction. Knowledge is directly infused into the learning algorithm level by our methodology. Knowledge injection, as defined by von Rueden et al. (2021), entails restricting feature relationships and can take place in four ways: (i) on the training data; (ii) on the hypothesis set; (iii) on the learning algorithm; and (iv) on the final hypothesis.

In this regard, we make three relevant contributions to the literature. First, our technique allows the Decision Maker (DM) to inject previous knowledge directly into the model learning processes. Therefore this technique may alleviate the model's failure to generalize due to scarce data or biased one. Our approach is similar to the Physics-guided Neural Networks (PGNN) proposed by Daw et al. (2021). However, in PGNN, the effects constraints are conditional on the context as in Muralidhar et al. (2018) or applied to non-DL techniques (Kotłowski & Słowiński, 2009; Lauer & Bloch, 2008; von Kurnatowski et al., 2021). A key advantage of our approach is that it can be applied to any DL architecture and is not conditional on features' context. As a proof of concept, we propose a credit risk empirical assessment as it is a high-stakes context. Moreover, in this field, recent frameworks as proposed by Bücker et al. (2021) do not allow for interpretability from the start. In other words, these techniques can spot model biases but cannot counter them, as their scope sole post hoc explainability. Our methodology can handle both these aspects by leaving a model compliant with the new guidelines on Sustainable AI. Second, we account for nonlinear effects and local knowledge gaps. Defining ad-hoc knowledge functions on model parameters allows for this constraint. This extra stipulation is required for two reasons. To begin, the credit risk empirical literature argues that the performance of DL models is mostly related to their flexibility (Ciampi & Gordini, 2013). The second reason for introducing nonlinearity is that knowledge in some areas of the feature space may be missing. Third, we investigate the interaction between model-agnostic post hoc interpretability approaches, such as Accumulated Local Effects (Apley & Zhu, 2020). In our plan, these methods serve two important functions. They initially provide graphical visuals to the DM, allowing him to communicate with non-experts. Second, they serve as sanity checks for our technique and explainability-based hyperparameter optimization.

This is how the rest of the paper is organized. Section two covers knowledge injection into the model as well as multicriteria problem formulation. The data sample used to test our technique, software packages, and hardware is shown in Section three. The findings are summarized in Section four and the most important ones are examined. The fifth section draws to an end.

2 Methodology

2.1 Deep Learning

DL is an AI subfield and type of Machine Learning technique aimed at developing systems that can operate in complex environments (Goodfellow et al., 2016). Deep architectures underpin DL systems which can be defined as:

DL is a subfield of AI and a form of Machine Learning technique focused at producing systems that can operate in complex contexts (Goodfellow et al., 2016). Deep architectures are the foundations of DL systems, which are defined as:

$$\mathcal{F} = \{f(\cdot, w), w \in \mathcal{W}\} \tag{1}$$

where $f(\cdot, w)$ denotes a shallow architecture, such as the Perceptron presented by Rosenblatt (1958). McCulloch and Pitts (1943) presented a method in which binary neurons grouped in a ring could do simple logic operations before Rosenblatt. In modern Artificial Neural Networks (ANNs) designs, neither the Perceptron nor the system presented by McCulloch and Pitts (1943) are employed. Gradient-based optimization techniques, notably Stochastic Gradient Descent, are used in modern architectures (SGD).

This research puts our method to the test on two different DL architectures: the Multilayer Perceptron (MLP) and the ResNet (RN). The MLP was chosen because it is an off-the-shelf solution in many use-cases, particularly in credit risk (Ciampi et al., 2021) and various publications on retail credit risk (Lessmann et al., 2015). MLP is made up of a densely connected network of nodes organized in a direct acyclic network. Inputs are supplied into the node's activation function after being weighted and shifted by a bias term and impact each successive layer until the final output layer.

In a binary classification task, the output of an MLP can be described as in Arifovic and Gencay (2001) by:

$$f(x) = \phi\left(\beta_0 + \sum_{j=1}^{d} \beta_j G\left(\gamma_{j0} + \sum_{i=1}^{p} \gamma_{ji} x_i\right)\right) \tag{2}$$

Because it features "shortcut connections," RN differs from the canonical MLP architecture in that it mitigates the problem of degradation in the event of numerous layers (He et al., 2015). Although the use of shortcut connections is not new in the literature (Venables & Ripley, 1999), He et al. (2015) proposed that identity mapping be used instead of any other nonlinear transformation. The simplest building unit of the ResNet architecture is depicted in Fig. 1. The shortcut has an impact on both layers in this architecture. And the final output gets both the x inputs and the layers' transformation.

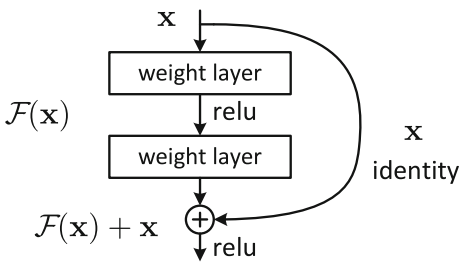


Fig. 1 Residual Network skip connection block

2.2 Multicriteria optimization

Multiple Criteria Decision-Making (MCDM) is a branch of Operations Research and Management Science. MCDM methods allow the DM to include numerous, possibly conflicting, criteria into the analytical processes. MCDM problems are more common than single criteria ones and have been studied in several fields, including economics, engineering, finance, and management Colapinto et al. (2015). MCDM techniques are used extensively in DL, especially in the training process and on final model evaluation Yang et al. (2020). An MCDM problem takes the following form:

$$\min_{\boldsymbol{x}} \quad \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\} \tag{3}$$

$$\text{subject to} \quad \boldsymbol{x} \in S \tag{4}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ is the ith objective and the vector $\boldsymbol{x}$ contains the decision variables that belong to the feasible set S.

When dealing with MCDMmulticriteria optimization problems, scalarization is a frequent strategy. A vector optimization problem is scalarized into a single objective optimization problem. To solve our problem, we start with a weighted sum scalarization:

$$\min_{\boldsymbol{x}} \quad \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}) \tag{5}$$

$$\text{subject to} \quad \boldsymbol{x} \in S \tag{6}$$

where the weights express the relative preference of the DM toward a specific goal. Preferences incorporation can happen in two ways, either a priori or a posteriori. In our approach, we use an a posteriori method as this best suits the DM's lack of knowledge, which may be uncertain about the relative importance of each objective.

2.3 Knowledge injection

As posed by von Rueden et al. (2021), knowledge in this paper is validated information about relations between entities in specific contexts. This type of formulation allows for formalization, suggesting that knowledge can be represented mathematically. Let's assume we have a deep architecture such that $\hat{y} = \mathcal{F}(\boldsymbol{x}, \mathcal{W})$. We observe the true label y in a supervised setting, and we can train a model using a differentiable loss function, similar to how we train a model for regression using the mean squared error (MSE):

$$Loss_f(\boldsymbol{x}, \mathcal{W}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{7}$$

or in our case, of a binary classification with the binary cross-entropy:

$$Loss_f(\boldsymbol{x}, \mathcal{W}) = \frac{1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right] \tag{8}$$

The first goal of our loss function, namely data fitting, will be achieved. Instead, the knowledge injection will operate on the effects of the characteristics on the model outcome hence our knowledge-based goal will be:

$$Loss_k(\boldsymbol{x}, \mathcal{W}) = k(\boldsymbol{x}) \odot \frac{\delta \mathcal{F}(\boldsymbol{x}, \mathcal{W})}{\delta \boldsymbol{x}} \tag{9}$$

where $k(\boldsymbol{x})$ is a function that penalizes/favorites certain effects of the gradient with range $[-1, 1]$, and the right-hand side of the Hadamard product is the gradient of our DL model at the feature level $\boldsymbol{x}$. Because knowledge does not survive throughout the entire feature space, one option is to define $k(\boldsymbol{x})$, which translates the feature space to the knowledge we expect on that specific feature neighbor.

In its most straightforward formulation, $k(\boldsymbol{x})$ can be a scalar. Three influences on the model may be identified in this situation. If $k(\boldsymbol{x}) = \mathbf{1}$ then all partial derivatives shuold be negative. therefore the result is decreasing monotonicity. The opposite holds for the case when $k(\boldsymbol{x}) = \mathbf{-1}$. When $k(\boldsymbol{x}) = \mathbf{0}$ there is no constraint on the gradient behavior, meaning that knowledge is non-existent and therefore not injected. Following Daw et al. (2021), we can add a new constraint to our Multicriteria function that gauges network complexity, such as a L_2 regularization on the weights. The following unconstrained minimization emerges as a result:

$$\min_{\mathcal{W}} \boldsymbol{\lambda}^{\top} \begin{bmatrix} Loss_f(\boldsymbol{x}, \mathcal{W}) \\ ||\mathcal{W}||_2 \\ Loss_k(\boldsymbol{x}, \mathcal{W}) \end{bmatrix} \tag{10}$$

The key feature of our technique is that it injects knowledge at the learning phase. As a result, our approach is subject to the same restrictions affecting any differentiable programming problem. A crucial implication is that our approach works with categorical data, provided the embedding.

2.4 Interpretability methods

A DL model is essentially a blackbox. The number of parameters and transformations that the inputs have before reaching the output prevents any meaningful understanding by the DM. Model interpretability methods try to address this issue. These techniques are numerous Molnar et al. (2020) and sortable under different axes. An example is whether a technique provides a global or local explainability measure. A second example is whether a technique has access to the model's parameters or not. The latter are called model-agnostic techniques, the former model-aware. In general model-agnostic methods, as the name suggests, apply to all models. Instead, model-aware techniques are specific to each class of model. Such characteristic means that model-aware models are more efficient and converge faster to the genuine interpretability metric. At least faster than their model-agnostic counterpart. The first interpretability method was a model-agnostic one, the Partial Dependence (PD). Proposed by Friedman (1991), PD evaluates the change in the average predicted value for a given feature. This metric is computed by varying the features over their marginal distribution Goldstein et al. (2015). As they rely solely on the marginal distribution, PD can be misleading in the case of feature dependence. Intuitively, given two financial ratios that share the same indicator, it is nonsense to let just one vary over its entire marginal distribution. The resulting synthetic data point used for PD evaluation will be outside the data envelope. Because of this reason, Apley and Zhu (2020) developed a new metric, the Accumulated Local Effect (ALE). The differences between PD and ALE are two. The first one is that they rely on features' conditional distribution. In practice, this is achieved by binning the feature space. The bin size is an arbitrary parameter. A narrow binning will result in shaky ALE, whereas wide bins can still have the problem of extrapolation outside the data envelope. The second one is that the effects are accumulated rather than averaged, as in the case of PD. In their application,

ALE is a model-agnostic technique. However, if the prediction function is differentiable, ALE can be rewritten in model-aware form:

$$ALE_{\hat{f},S}(x) = \int_{z_{0,S}}^{x} \left[\int \frac{\delta \hat{f}(z_S, X_{\setminus S})}{\delta z_S} d\mathcal{P}(X_{\setminus S}|z_S) \right] dz_S - C \tag{11}$$

where $\hat{f}$ is the black-box model and $\frac{\delta \hat{f}(z_S, X_{\setminus S})}{\delta z_S}$ its gradient. S identifies the subset of variables' index. X is the matrix containing all the variables, and x is the vector containing the feature values per observation. z identifies the boundaries of the K partitions, such that $z_{0,S} = min(x_S)$, Last, C is a constant term to center the plot.

Having the gradient inside an interpretability measure is not new in the literature. Baehrens et al. (2010) proposed an interpretability technique based on the product of the model's gradient with feature values. Simonyan et al. (2014) proposed Saliency Maps based on the gradient of model output to the input features. Therefore a knowledge injection strategy as proposed in 9 will have an impact on these techniques. The empirical experiment provides an analysis of such impact.

3 Data, software, hardware

3.1 Data

To test the goodness of our approach, we provide an empirical application in the context of credit risk. In particular on the problem of bankruptcy prediction. We used a publicly available dataset of Polish enterprises donated to the UCI Machine Learning Repository by Zikeba et al. (2016). The data contains information about the financial conditions of Polish companies belonging to the manufacturing sector. The dataset contains 64 financial ratios ranging from liquidity to leverage measures [1]. Moreover, the dataset distinguishes five classification cases that depend on the forecasting period. In our empirical setting, we focus on bankruptcy status after one year. In this subset of data, the total number of observations is 5910, out of which only 410 represents bankrupted firms. It is worth noting that we do not counter the class imbalance in the empirical setting, although this is something done commonly in the literature. We retained class imbalance to test the robustness of our approach even in conditions of scarcity of a particular class and used robust metrics such as the Area Under the Receiving Operating Curve (AUROC). Moreover, as our empirical experiment focuses on testing our approach on model interpretability, we restricted the number of features we considered to six. This is due to the fact that ALEs are inspected as plots, and having a plot for each feature increases complexity without providing any additional benefit to the reader or our approach. The choice was to focus on Attr 13, Attr 16, Attr 23, Attr 24, Attr 26, and Attr 27. The attributes were selected by using a ROC-based feature selection (Kuhn & Johnson, 2019).

We give an empirical application in the area of credit risk to test the validity of our technique. In specifically, the challenge of predicting insolvency. Zikeba et al. (2016) provided a publicly available dataset of Polish businesses to the UCI Machine Learning Repository. The report includes information on the financial health of Polish manufacturing enterprises. There are 64 financial ratios in the dataset, spanning from liquidity to leverage measures [2].

[1] For a complete description of the indicators, please consider Table 4 in the Appendix.

[2] Please see Table 4 in the Appendix for a detailed description of the indicators.

Furthermore, the dataset separates five classification scenarios based on the predicting period. In our empirical setting, we focus on bankruptcy status after one year. The total number of observations in this subset of data is 5910, with only 410 being defunct companies. It's worth emphasizing that we don't correct for class imbalance in the empirical context, despite the fact that this is a frequent practice in the literature. We kept the class imbalance to evaluate the durability of our strategy even when a single class was scarce, and we used robust measures like the Area Under the Receiving Operating Curve to do so (AUROC). Furthermore, we limited the number of attributes we analyzed to six because our empirical experiment focused on assessing our strategy on model interpretability. This is because ALEs are inspected as plots, and having a plot for each feature adds to the complexity without adding any value to the reader or our method. Attr 13, Attr 16, Attr 23, Attr 24, Attr 26, and Attr 27 were chosen as the focus points. The attributes were chosen using the (Kuhn & Johnson, 2019) ROC-based feature selection method.

3.2 Software and hardware

R is used to manage the entire workflow (R Core Team, 2021). The preprocessing relied on the tidymodels ecosystem (Kuhn & Wickham, 2020). The DL models are developed in Julia (Bezanson et al., 2017) using the Flux framework (Innes et al., 2018; Innes, 2018). The interoperability between the two languages is possible via the JuliaConnectoR library (Lenz et al., 2021). As for the hardware, the pipeline is carried out on a local machine with 12 logical cores (Intel i7-9850H), 16 GB RAM, and a Cuda enabled graphic card (NVIDIA Quadro T2000). Both Julia and R codes are freely available for research reproducibility on GitLab, and an ad-hoc Docker container has been created on DockerHub.

4 Results

We apply a typical practice in the field of DL to test the performance of our strategy on the dataset of Polish enterprises. We divided the dataset into two parts: training and testing. The dataset is divided into two parts: one for training the model and the other for testing its performance. A configuration like this will suffice in the event of a model with no hyperparameters. In DL, however, this is seldom the case because these models require extensive hyperparameter calibration. The hyperparameters in our setup are the elements contained in $\boldsymbol{\lambda}$. As a result, using the training set to perform what is known as hyperparameter optimization (Goodfellow et al., 2016) is a frequent method. As a result, the training set is split into training and validation sets, and the model is fitted and validated using various parameters. In our example, we used the bootstrap technique to extract ten samples from the training set, and we trained our model with several hyperparameter combinations. We used grid search, which is also known as full factorial design (Montgomery, 2017), to implement such hyperparameter search. The model was then trained on the whole training set, and the appropriate hyperparameters were used to classify bankruptcy state on the test set.

We divided the study into three stages to ensure a solid understanding of the findings. The MLP and the RN were used to undertake hyperparameter optimization and subsequent hold-out testing, with the former being the best performing model. Second, we used ALE plots to examine the effect of the MLP with knowledge injection on interpretation. Finally, we put our method to the test by reducing the amount of data we used.

4.1 Performance review

Because our model validation included ten bootstrap samples, table 1 shows the mean AUROC as well as its standard errors. The first remarkable finding is that without regularization and knowledge injection, both the MLP and the RN perform badly. This finding is consistent with that of Zikeba et al. (2016), which indicated that ANN designs suffer from poor generalization. Instead, the gain in performance when both are present is of tremendous interest. The table is set with regularization and knowledge injection. With low amounts of knowledge injection and a moderate level of regularization, the RN appears to perform better. What matters, though, is how the MLP behaves. The model is more responsive to knowledge injection and outperforms the competition in model validation. When $\lambda_3 = 0.3$, this result is obvious. With a little level of standard error, all of the MLP models have an AUROC of 0.8 or above. When regularization starts to ramp up $\lambda_2 = 0.3$ even with inserted information, another key outcome of the MLP is performance degradation. It's worth mentioning that the dataset has a major class imbalance, and nothing has been done to address it in order to test the efficacy of our approach in this setting. Indeed, knowledge injection reduces misclassification and results in a model that is comparable to other reliable classifiers.

The results in Table 1 are encouraging. The performances from the test set, on the other hand, must be incorporated for measuring model generalization error. These results are presented in Table 2 by just considering the ideally parametrized models and their baseline, that is, the model with $\lambda_1 = 1$. This table clearly shows that the MLP generalizes far better than its competitor, with a minor drop in performance that is in accordance with predictions. This finding suggests that knowledge injection combined with modest regularization can improve a DL classifier's generalization performance and make it more resistant to class imbalances.

Table 1 Performances of Multilayer Perceptron and Residual Network on the training set with various hyperparameter settings. The performance is measured as the mean and standard errors of the Area Under the Receiving Operating Curve in each bootstrap sample. Bold values indicate the best performing hyperparametrization for each model

λ_1	λ_2	λ_3	Multilayer perceptron		Residual network	
			Mean	Standard error	Mean	Standard error
1.0	0.0	0.0	0.6585	0.0178	0.5061	0.0843
0.9	0.1	0.0	0.6924	0.0111	0.5308	0.0805
0.8	0.2	0.0	0.6302	0.0757	0.6326	0.0180
0.7	0.3	0.0	0.7175	0.0059	0.5584	0.0900
0.9	0.0	0.1	0.7905	0.0087	0.6418	0.0740
0.8	0.1	0.1	0.8286	0.0149	0.5744	0.0769
0.7	0.2	0.1	0.7586	0.0336	0.5059	0.0746
0.6	0.3	0.1	0.6163	0.1219	**0.6604**	0.0879
0.8	0.0	0.2	0.8263	0.0129	0.5124	0.0664
0.7	0.1	0.2	0.8242	0.0170	0.6102	0.0186
0.6	0.2	0.2	0.8206	0.0123	0.5037	0.0443
0.5	0.3	0.2	0.7617	0.0601	0.6249	0.0593
0.7	0.0	0.3	0.8202	0.0119	0.6074	0.0172
0.6	0.1	0.3	0.8289	0.0139	0.5410	0.0417
0.5	0.2	0.3	**0.8306**	0.0135	0.5318	0.0150
0.4	0.3	0.3	0.8178	0.0198	0.5628	0.0378

Table 2 Performances of Multilayer Perceptron and Residual Network on the test set with validated and baseline hyperparameter settings. The performance is measured as the Area Under the Receiving Operating Curve in the test sample

λ_1	λ_2	λ_3	Multilayer perceptron	Residual network
1.0	0.0	0.0	0.577	**0.582**
0.5	0.2	0.3	**0.821**	–
0.4	0.3	0.1	–	0.518

Bold means better performance

In the sections that follow, we'll look at how the MLP performs in terms of interpretability and robustness to data scarcity with and without knowledge injection. We'll just concentrate on the MLP because it's the most performant architecture.

4.2 Interpretability review

Current interpretability methods, as indicated in the preceding sections, are vital tools for model debugging and inspecting any model bias. As a result, Fig. 2a shows MLP ALEs with and without knowledge injection. The model's ALEs without knowledge injection exhibit a number of misbehaviors that could be related to class imbalance or hidden biases in the training sample. In depth:

- Attr 13: which is also known as the EBITDA-To-Sales ratio, is a profitability indicator. Therefore we should expect to decrease the probability of bankruptcy, especially in cases where the ratio is positive. The opposite occurs instead. An increase of the ratio above zero increases the probability of bankruptcy. This effect is at odds with the literature on the subject as, for example, in Platt and Platt (2002).
- Attr 16: is the inverse and a proxy of the Debt-To-EBITDA ratio which is leverage ratio. For the inverse of a leverage ratio, we would assume a negative impact on bankruptcy as in Beaver (1968).
- Attr 23: is the Net profit ratio and is a productivity ratio Lee and Choi (2013) which tends to have a negative impact on bankruptcy.

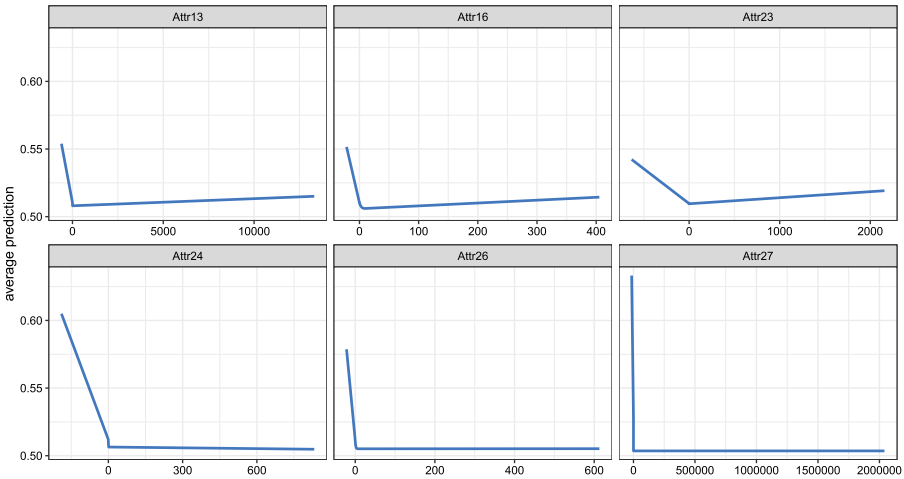
To account for these typical biased effects, we assumed the following logistic form for the knowledge function of all features:

$$k(x) = \frac{1}{1 + e^{-100x}} \tag{12}$$

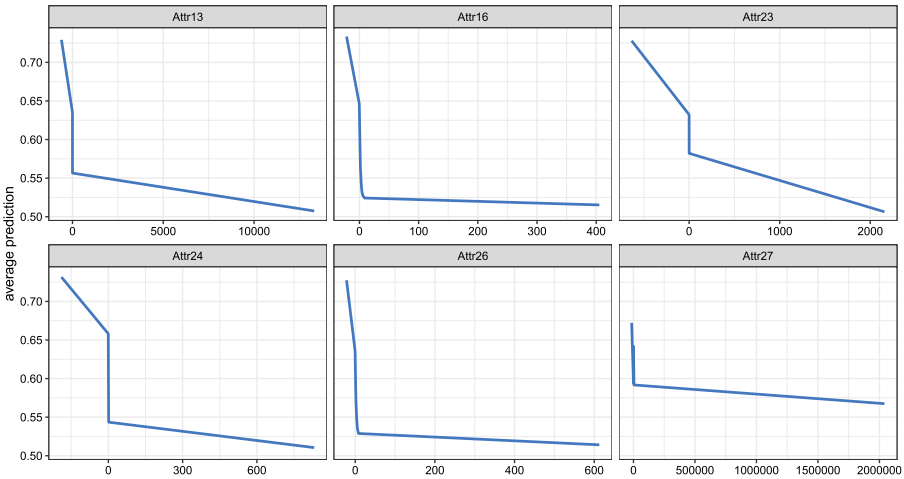
Such a knowledge function penalizes only positive effects above zero and retains the correctly captured effects below it. With this setting, in the case of moderate knowledge injection, the effects align with the literature findings.

4.3 Robustness checks

A key topic is how model performance degrades as the amount of training data decreases. Knowledge injection has been used in the past to solve difficulties like those described in von Kurnatowski et al. (2021). We steadily reduced the training set and measured the matching performance on the test set to see how our approach dealt with scarce data. These results are shown in Table 3, which shows how the test set performs as the proportion of training data



(a) Accumulated Local Effects plot of the Multilayer Perceptron architecture, without regularization and knowledge injection (i.e. $\lambda_1 = 1, \lambda_2 = 0.0, \lambda_3 = 0.0$).



(b) Accumulated Local Effects plot of the Multilayer Perceptron architecture, with regularisation and knowledge injection optimally selected from the hyperparameter optimization procedure (i.e. $\lambda_1 = 0.5, \lambda_2 = 0.2, \lambda_3 = 0.3$).

Fig. 2 Accumulated Local Effects plot of the Multilayer Perceptron architecture, with and without regularization and knowledge injection

drops. Our strategy, which is in line with the literature on knowledge injection, eliminates performance degradation even when only half of the dataset is used for training.

Table 3 Performances of Multilayer Perceptron on the test set under different proportions of train/test split. The performance is measured as the Area Under the Receiving Operating Curve in the test sample

Train/Test	With knowledge ($\lambda_1 = 0.5, \lambda_2 = 0.2, \lambda_3 = 0.3$)	Without knowledge ($\lambda_1 = 0.1, \lambda_2 = 0.0, \lambda_3 = 0.0$)
0.85	0.829	0.615
0.80	0.828	0.543
0.75	0.821	0.577
0.7	0.822	0.605
0.65	0.790	0.613
0.6	0.817	0.646
0.55	0.803	0.505
0.5	0.823	0.641

5 Conclusion

We developed a novel method for knowledge injection at the level of feature effects in a DL model in this study. Model interpretability is a very important problem, and new legislation requires it from the start. Recent post-hoc interpretability solutions fall short of this requirement. Our solution solves the problem by allowing model interpretation to be controlled from the beginning. The method entails addressing a multicriteria minimization problem in which greedy data fitting and regularization conflict with knowledge adherence. By constructing ad-hoc knowledge functions on the model's parameters, we were able to account for partial knowledge and nonlinearity. To demonstrate our approach, we presented a use case of bankruptcy prediction using a dataset from a Polish firms. The findings imply that knowledge injection enhances model performance and keeps model interpretation consistent with literature findings, avoiding idiosyncratic effects caused by class imbalance or potential dataset biases. The DM can test the effects of our approach using post-hoc interpretability approaches, which are critical for fine-tuning the model before it goes into production. Another important subject we addressed was model performance degradation in the event of data scarcity. Our findings show that knowledge injection provides the modeler with more flexibility in terms of obtaining the data required for proper model training.

In terms of research, this new paradigm opens up a slew of new possibilities. The examination of increasingly complicated knowledge functions is one avenue for future research. A second line of inquiry would be to ensure knowledge consistency across many contexts, such as time series. These are only a few examples of potential new research initiatives that will pave the way for knowledge-informed DL.

Funding Open access funding provided by Universitá degli Studi di Milano - Bicocca within the CRUI-CARE Agreement.

Appendix A. Dataset indicators

See Table 4.

Table 4 Dataset financial indicators with corresponding description

ID	Description	Effect	ID	Description	Effect
Attr 1	Net profit/total assets	↘	Attr 33	Operating expenses/short-term liabilities	↗
Attr 2	Total liabilities/total assets	↗	Attr 34	Operating expenses/total liabilities	↗
Attr 3	Working capital/total assets	↘	Attr 35	Profit on sales/total assets	↘
Attr 4	Current assets/short-term liabilities	↘	Attr 36	Total sales/total assets	↘
Attr 5	{[}(cash + short-term securities + receivables - short-term liabilities)/(operating expenses - depreciation){]} * 365	↗	Attr 37	(Current assets - inventories)/ long-term liabilities	↘
Attr 6	Retained earnings/total assets	↘	Attr 38	Constant capital/total assets	↗
Attr 7	EBIT/total assets	↘	Attr 39	Profit on sales/sales	↘
Attr 8	Book value of equity/total liabilities	↘	Attr 40	(Current assets - inventory - receivables)/short-term liabilities	↗
Attr 9	Sales/total assets	↘	Attr 41	Total liabilities/((profit on operating activities + depreciation) * (12/365))	↗
Attr 10	Equity/total assets	↗	Attr 42	Profit on operating activities/sales	↘
Attr 11	(Gross profit + extraordinary items + financial expenses)/total assets	↗	Attr 43	Rotation receivables + inventory turnover in days	↗
Attr 12	Gross profit/short-term liabilities	↘	Attr 44	(Receivables * 365)/sales	↗
Attr 13	(Gross profit + depreciation)/sales	↘	Attr 45	Net profit/inventory	↘
Attr 14	(Gross profit + interest)/total assets	↘	Attr 46	(Current assets - inventory)/short-term liabilities	↗
Attr 15	(Total liabilities * 365)/(gross profit + depreciation)	↗	Attr 47	(Inventory * 365)/cost of products sold	↗
Attr 16	(Gross profit + depreciation)/total liabilities	↘	Attr 48	EBITDA (profit on operating activities - depreciation)/total assets	↘
Attr 17	Total assets/total liabilities	↘	Attr 49	EBITDA (profit on operating activities - depreciation)/sales	↘
Attr 18	Gross profit/total assets	↘	Attr 50	Current assets/total liabilities	↘
Attr 19	Gross profit/sales	↘	Attr 51	Short-term liabilities/total assets	↗
Attr 20	(Inventory * 365)/sales	↗	Attr 52	(Short-term liabilities * 365)/cost of products sold)	↗
Attr 21	Sales (n)/sales (n-1)	↘	Attr 53	Equity/fixed assets	↗
Attr 22	Profit on operating activities/total assets	↘	Attr 54	Constant capital/fixed assets	↗
Attr 23	Net profit/sales	↘	Attr 55	Working capital	↘
Attr 24	Gross profit (in 3 years)/total assets	↘	Attr 56	(Sales - cost of products sold)/sales	↘

Table 4 continued

ID	Description	Effect	ID	Description	Effect
Attr 25	(Equity - share capital)/total assets	↗	Attr 57	(Current assets - inventory - short-term liabilities)/(sales - gross profit - depreciation)	↗
Attr 26	(Net profit + depreciation)/total liabilities	↘	Attr 58	Total costs/total sales	↗
Attr 27	Profit on operating activities/financial expenses	↘	Attr 59	Long-term liabilities/equity	↗
Attr 28	Working capital/fixed assets	↘	Attr 60	Sales/inventory	↘
Attr 29	Logarithm of total assets	↘	Attr 61	Sales/receivables	↘
Attr 30	(Total liabilities - cash)/sales	↗	Attr 62	(Short-term liabilities *365)/sales	↗
Attr 31	(Gross profit + interest)/sales	↘	Attr 63	Sales/short-term liabilities	↘
Attr 32	(Current liabilities * 365)/cost of products sold	↗	Attr 64	Sales/fixed assets	

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., & Mané, D., et al. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/.

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82*(4), 1059–1086. https://doi.org/10.1111/rssb.12377.

Arifovic, J., & Gencay, R. (2001). Using genetic algorithms to select architecture of a feedforward artiÿcial neural network. *Physica A*, p. 21.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research, 11,* 1803–1831.

Beaver, W. H. (1968). Alternative accounting measures as predictors of failure. *The Accounting Review, 43*(1), 113–122.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review, 59*(1), 65–98. https://doi.org/10.1137/141000671.

Bücker, M., Szepannek, G., Gosiewska, A., & Biecek, P. (2021). Transparency, auditability, and explainability of machine learning models in credit scoring. *Journal of the Operational Research Society*. https://doi.org/10.1080/01605682.2021.1922098.

Ciampi, F., Giannozzi, A., Marzi, G., & Altman, E. I. (2021). Rethinking SME default prediction: A systematic literature review and future perspectives. *Scientometrics*, pp. 1–48.

Ciampi, F., & Gordini, N. (2013). Small enterprise default prediction modeling through artificial neural networks: An empirical analysis of Italian small enterprises. *Journal of Small Business Management, 51*(1), 23–45.

Colapinto, C., Jayaraman, R., & Marsiglio, S. (2015). Multi-criteria decision analysis with goal programming in engineering, management and social sciences: a state-of-the art review. *Annals of Operations Research, 251*(1–2), 7–40. https://doi.org/10.1007/s10479-015-1829-1.

Daw, A., Karpatne, A., Watkins, W., Read, J., & Kumar, V. (2021). Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. http://arxiv.org/abs/1710.11431.

Dexe, J., Ledendal, J., & Franke, U. (2020). An empirical investigation of the right to explanation under gdpr in insurance. *Lecture Notes in Computer Science*, pp. 125-139. ISSN 1611-3349. https://doi.org/10.1007/978-3-030-58986-8_9.

European Commission. (2019). *Ethics guidelines for trustworthy ai*. European Commission: Technical report.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics, 19*(1), 1–67.

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics, 24*(1), 44–65.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. http://arxiv.org/abs/1512.03385.
Innes, M. (2018). Flux: Elegant machine learning with julia. Journal of Open Source Software. https://doi.org/10.21105/joss.00602.
Innes, M., Saba, E., Fischer, K., Gandhi, D., Rudilosso, M. C., Joy, N. M., Karmali, T., Pal, A., & Shah, V. (2018). Fashionable modelling with flux. *CoRR*, abs/1811.01457. https://arxiv.org/abs/1811.01457.
Kotłowski, W., & Słowiński, R. (2009). Rule learning with monotonicity constraints. In *Proceedings of the 26th annual international conference on machine learning*, ICML '09, pp. 537–544. Association for Computing Machinery. ISBN 978-1-60558-516-1. https://doi.org/10.1145/1553374.1553444.
Kuhn, M., & Johnson, K. (2019). Feature Engineering and Selection: A Practical Approach for Predictive Models. CRC Press.
Kuhn, M., & Wickham, H. (2020). *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*. https://www.tidymodels.org.
Lauer, F., & Bloch, G. (2008). Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomputing, 71*(7–9), 1578–1594. https://doi.org/10.1016/j.neucom.2007.04.010.
Lecun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1990). Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann. https://proceedings.neurips.cc/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html.
Lee, S., & Choi, W. S. (2013). A multi-industry bankruptcy prediction model using back-propagation neural network and multivariate discriminant analysis. *Expert Systems with Applications, 40*(8), 2941–2946. https://doi.org/10.1016/j.eswa.2012.12.009.
Lenz, S., Hackenberg, M., & Binder, H. (2021). The JuliaConnectoR: A functionally oriented interface for integrating Julia in R. http://arxiv.org/abs/2005.06334.
Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An upyear of research. *European Journal of Operational Research, 247*(1), 124–136.
McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115–133. https://doi.org/10.1007/BF02478259.
Molnar, C., Casalicchio, G., & Bischl, B. (2020). Interpretable machine learning—A brief history, state-of-the-art and challenges. In Irena K., Michael K., Annalisa A., Corrado L., Luiza A., Albrecht Z., Riccardo G., Özlem Ö., Ribeiro, R. P., Ricard G., João, G., Linara A., Yamuna K., Pedro M. F., Donato M., Ibéria M., Michelangelo C., Giuseppe M., Elio M., Zbigniew W. R., Peter C., Eirini N., Erich S., Arthur Z., Anna M., Przemyslaw B., Salvatore R., Benjamin K., Andreas L., and Jon Atle G., (eds). *ECML PKDD 2020 Workshops*, Communications in Computer and Information Science, pages 417–431. Springer International Publishing. ISBN 978-3-030-65965-3. https://doi.org/10.1007/978-3-030-65965-3_28.
Montgomery, D. C. (2017). *Design and Analysis of Experiments*. John wiley & sons.
Muralidhar, N., Islam, M. R., Marwah, M., Karpatne, A., & Ramakrishnan, N. (2018). Incorporating Prior Domain Knowledge into Deep Neural Networks. In *2018 IEEE International conference on big data (Big Data)*, pp. 36–45. https://doi.org/10.1109/BigData.2018.8621955.
Nosratabadi, S., Mosavi, A., Duan, P., Ghamisi, P., Filip, F., Band, S. S., et al. (2020). Data science in economics: Comprehensive review of advanced machine learning and deep learning methods. *Mathematics, 8*(10), 1799. https://doi.org/10.3390/math8101799.
Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2020.106384.
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché Buc, F., Fox E., and Garnett, R. (Eds). *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
Platt, H. D., & Platt, M. B. (2002). Predicting corporate financial distress: Reflections on choice-based sample bias. *Journal of economics and finance, 26*(2), 184–199.
R Core Team. (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.
Rao, Q., & Frtunikj, J. (2018). Deep learning for self-driving cars: Chances and challenges. In *Proceedings of the 1st International workshop on software engineering for AI in autonomous systems*, SEFAIS '18,

pp. 35–38. Association for Computing Machinery. ISBN 978-1-4503-5739-5. https://doi.org/10.1145/3194085.3194087.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386–408. https://doi.org/10.1037/h0042519.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. http://arxiv.org/abs/1312.6034.

Venables, W. N., & Ripley, B. D. (1999). *Modern Applied Statistics with S-PLUS*. Statistics and Computing. Springer-Verlag, 3 edition. ISBN 978-1-4757-3121-7. https://doi.org/10.1007/978-1-4757-3121-7.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z.

von Kurnatowski, M., Schmid, J., Link, P., Zache, R., Morand, L., Kraft, T., Schmidt, I., & Stoll, A. (2021). Compensating data shortages in manufacturing with monotonicity knowledge. http://arxiv.org/abs/2010.15955.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., Walczak, M., Garcke, J., Bauckhage, C., & Schuecker, J. (2021). Informed Machine Learning—A Taxonomy and Survey of Integrating Knowledge into Learning Systems. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1. ISSN 1041-4347, 1558-2191, 2326-3865. https://doi.org/10.1109/TKDE.2021.3079836.

Yang, M., Nazir, S., Xu, Q., & Ali, S. (2020). Deep learning algorithms and multicriteria decision-making used in big data: A systematic literature review. *Complexity, 2020,* 1–18. https://doi.org/10.1155/2020/2836064.

Zhang, D., Cao, D., & Chen, H. (2019). Deep learning decoding of mental state in non-invasive brain computer interface. In *Proceedings of the International conference on artificial intelligence, information processing and cloud computing*, AIIPCC '19, pp. 1–5. Association for Computing Machinery. ISBN 978-1-4503-7633-4. https://doi.org/10.1145/3371425.3371441.

Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion, 42,* 146–157. https://doi.org/10.1016/j.inffus.2017.10.006.

Zikeba, M., Tomczak, S. K., & Tomczak, J. M. (2016). Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Systems with Applications*.