Received March 11, 2021, accepted March 21, 2021, date of publication March 24, 2021, date of current version April 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068854

Making Deep Learning-Based Predictions for Credit Scoring Explainable

XOLANI DASTILE[1] AND TURGAY CELIK[2,3,4], (Member, IEEE)
[1]School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg 2000, South Africa
[2]School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg 2000, South Africa
[3]Wits Institute of Data Science, University of the Witwatersrand, Johannesburg 2000, South Africa
[4]School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China

Corresponding author: Turgay Celik (celikturgay@gmail.com)

ABSTRACT Credit scoring has become an important risk management tool for money lending institutions. Over the years, statistical and classical machine learning models have been the most researched risk management tools in credit scoring literature, and recently the focus has turned to deep learning models. This transition is due to better performances that are shown by deep learning models in different domains. Despite deep learning models' superior performances, there is still a need for explaining how these models make their predictions. The non-transparency nature of deep learning models has created a bottleneck for their use in credit scoring. Explanations of decisions are important for lending institutions since it is a requirement for automated decisions that are generated by non-transparent models to be explained. The other issue in using deep learning models, specifically 2D Convolutional Neural Networks (CNNs), in credit scoring is the need to have the data in image format. We propose an explainable deep learning model for credit scoring which can harness the performance benefits offered by deep learning and yet comply with the legislation requirements for the automated decision-making processes. The proposed method converts tabular datasets into images and thus allowing the application of 2D CNNs in credit scoring. Each pixel of the image corresponds to a feature bin of the tabular dataset. The predictions from the 2D CNNs were explained using state-of-the-art explanation methods. Furthermore, explanations were evaluated using a sanity check methodology and also performances of the explanation methods were compared quantitatively. The proposed explainable deep learning model outperforms the other credit scoring methods on publicly available credit scoring datasets.

INDEX TERMS Credit scoring, convolutional neural network, deep learning, explainable artificial intelligence.

I. INTRODUCTION

Credit scoring, proposed by Durand [1], has become an important risk management tool for money lending institutions. Over the years, statistical and classical machine learning models have been the most researched risk management tools in credit scoring literature, and recently the focus has turned to deep learning models [2]. This transition is due to better performances that are shown by deep learning models in different domains. Despite better performances, there is still a need for explaining how deep learning models make their predictions. According to Liu *et al.* [3], an explanation is defined as "the ability to provide a visual or textual presentation of connections between input features and output predictions". Similarly, Doshi-Velez and Kim [4] defined an explanation as "the ability to explain or to provide the meaning in understandable terms to a human". Note that in this study we treat explanation and interpretability as interchangeable terms although in literature some argue that these terms differ [5], Rothman [6] annotates that an explanation creates comprehensibility and clarity, on the other hand interpretability tells the meaning. This study intends not to pursue the philosophical debate on the differences between explanation and interpretability. The aim of this current study is to explain predictions made by a deep learning model in a credit scoring setting.

Explaining a prediction is particularly important in credit scoring since it is now a requirement for lending institutions under the Basel Accord [7] to explain to an applicant why her/his loan application was denied. Regulations such

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Xiang.

as the European Union General Data Protection Regulation (GDPR) [8] have made it compulsory for machine learning models to explain their predictions under the notion ''right to explanation''. Explanations foster trust in model predictions and assure that no discriminations are incurred during the application process when assessing credit worthiness of loan applicants. Explanations are playing an important role in areas where decisions are critical, for example in healthcare and in banking, to mention a few. This requirement of the ''right to explanation'' is advancing the research field that is coined as Explainable Artificial Intelligence (XAI) [6]. XAI is not a new concept and it has a rich history of more than 50 years [9]. Breiman was amongst the early adopters of interpretable systems and Breiman is known for developing Classification and Regression Trees and is quoted in one of the papers stating that ''On interpretability, trees are an A+'' [10].

Samek and Müller [11] highlighted at high level, the different techniques (old and emerging) that are used for XAI. The need, the purpose of different users, the evaluation methods and the future directions of XAI were also discussed in [11]. However, Rudin [12] provides a contrasting view on XAI and highlights the reasons why interpretable models should be used over explainable black box models for high-stakes decisions such as criminal justice, healthcare and credit scoring. Their argument was based on the premise that if a second model is used to explain a black box model (i.e. post-hoc explanation), there is a high chance that the explanation is not ''faithful to what the original model computes'' [12]. The recommendation was that instead of using post-hoc explanations, inherently interpretable models (i.e. ante-hoc explanations) that perform similarly to some of the black box models should be used for high-stake decisions. It was argued further that there is no truth in the statement ''there is a trade-off between accuracy and interpretability'', in other words it is a myth that non-interpretable models are more accurate than inherently interpretable models. To decide whether to use inherently interpretable models or not should be based on the circumstantial needs. In their review article Chari *et al.* [13] synthesised a taxonomy of explanations from the literature. This taxonomy consists of nine different types of explanations such as case-based, contextual, contrastive, scientific, statistical, trace-based, everyday and simulation-based explanations. The aim was to help produce explanations that are aligned to the circumstantial needs. Despite these different views from the literature, our study was motivated by [14]. However, our study differs from [14] in such a way that we have systematically discretized data into optimal categories using weights of evidence and we used both categorical and continuous features as opposed to [14].

Our contributions in this paper are as follows. 1) This is the first study to convert tabular data into images using a novel method and also it is the first method after such a conversion takes place that employs different explanation methods to explain the decision on each predicted credit sample. 2) We empirically showed that the explanations from different explanation methods can be used as optimal features and we observed that there is a huge boost in the classifiers' performance. In this work, we showed for the first time that the explanations are good features that can be directly applied on the classifiers and perform better when compared to the same classifiers operating on the raw data.

The remainder of this paper is organised as follows, Section II reviews and examines previous research of deep learning models in credit scoring, explanations of convolutional neural networks in other domains as well as the conversion of tabular data into images. Section III discusses the proposed framework of this study. Section IV discusses conversion of tabular datasets into images, training of 2D convolutional neural networks and the techniques used for explaining individual predictions. The experiments are described in Section V. The results are presented in Section VI and the conclusion is in Section VII.

II. RELATED WORK

In recent years, researchers have shown an interest in the application of deep learning models in credit scoring. For instance, Zhu *et al.* [14] used a hybrid method by combining a relief algorithm (which performs feature selection) with Convolutional Neural Network (CNN) to perform credit scoring and the hybrid relief-CNN model showed better performances when compared to Logistic Regression and Random Forest (RF) models. Tomczak and Zieba [15] used Restricted Boltzmann Machine (RBM) in credit scoring. A comprehensible scoring table from RBM showed better performances compared to classical machine learning models. Li *et al.* [16] conducted a study of deep learning with clustering and merging and the results showed high prediction accuracies. Deep Multi-Layer Perceptron (DMLP) and Deep CNN (DCNN) were used by Neagoe *et al.* [17] to assess the credit worthiness of applicants. The DCNN significantly outperformed DMLP. Hamori *et al.* [18] compared Deep Neural Network (DNN) with RF, bagging and boosting methods on credit datasets and the results showed that the ensemble methods performed better than DNN.

Deep learning with RF feature importance approach was proposed by Ha and Nguyen [19] for credit scoring. The empirical results showed that the proposed approach performed better than baseline methods such as the Decision Trees, k-Nearest Neighbour, Naíve-Bayes, Multi-Layer Perceptron and Random Forest with the exception of Support Vector Machine on German credit dataset. The proposed approach outperformed all baseline methods on Australian credit dataset. In their empirical study Sirignano *et al.* [20] developed a cohort of neural network ensemble models to assist in predicting the probability of default in credit risk using German and Australian credit datasets. The results showed that the proposed neural network ensemble models provide similar or better performance results compared to the literature results as well as to baseline single classifiers. Tripathi *et al.* [21] proposed a novel algebraic activation function to improve the performance of Extreme Learning

Machine (ELM) model in credit scoring. The ELM consists of an input layer, a single hidden layer and an output layer. The study also proposed Bat algorithm which is an evolutionary approach to initialise the weights and biases of the ELM. The results showed a significant improvement on German and Australian credit datasets. Edla *et al.* [22] combined a binary particle swarm optimization and gravitational search algorithm (BPSOGSA) with a multi-layer ensemble classifier using five heterogeneous classifiers. The purpose of BPSOGSA was to select predictive features. The results showed that the proposed hybrid model outperforms the Random Forest model and other majority voting ensemble techniques on German, Australian and Japanese credit datasets. Similarly, Tripathi *et al.* [23] proposed a hybrid credit scoring model using a dimension reduction approach (i.e. neighbourhood rough set) and a multi-layer ensemble classifier. The experimental results showed that the proposed model can achieve satisfactory performance in credit scoring.

Regulated institutions are not willing to adapt models that cannot explain predictions and this is hampering the use of machine learning, specifically deep learning models. However, Ariza-Garzón *et al.* [24] conducted a study in credit scoring where the focus was to make non-transparent machine learning models explainable. In the study, classical machine learning models were compared to a logistic regression model for Peer-to-Peer (P2P) lending and predictions were explained using SHAP values. The results showed that the classical machine learning models not only perform better in terms of classification but also perform better in terms of explanations. Moreover, classical machine learning models showed that they can reflect dispersion, non-linearity and structural breaks in the relationships between each independent variable and the response variable [24]. The credit scoring literature has not researched extensively in explanations of deep learning predictions.

There has been interests in credit scoring and other fields for converting tabular datasets into images in order to apply 2D CNN models. For example, Zhu *et al.* [14] combined relief algorithm with CNN model. The relief algorithm was used to select predictive features and the CNN was used for classification. The study converted credit scoring tabular data into images by bucketing features and mapping them into image pixels. However, the study considered only numeric features. The results showed that the relief-CNN hybrid model performed better than the benchmark models in credit scoring such as the random forest and the logistic regression. Sharma *et al.* [25] proposed a novel approach called DeepInsight to convert non-image data into images and apply CNN models. The types of data that were used in the study were Ribonucleic Acid (RNA) sequence data, vowels data, text data and artificial data. The results showed a better performance from the DeepInsight approach compared to the state-of-the-art machine learning models such as the Ada-boost and the Random Forest models. Yang *et al.* [26] converted a multivariate time-series data firstly into two-dimensional colored images and secondly the two-dimensional colored images were combined to form a single image and lastly a CNN was applied on a single image for classification. The study compared three transformation methods for converting time series data into images. The transformation methods were Gramian Angular Summation Field (GASF), Gramian Angular Difference Field (GADF), and Markov Transition Field (MTF). The study was focusing on assessing the effect of using different transformation methods, the sequences of combining images, and the complexity of CNN architectures on classification accuracy. The results showed that the selection of transformation methods and the sequence of combination do not significantly impact the prediction outcome. Further, the results also showed that the proposed framework performed better than the other classification models in terms of the accuracy metric. Buturovi'c and Miljkovi'c [27] proposed a novel approach called TAbular Convolution (TAC) for converting tabular dataset into images for the application of 2D CNN models. The study used gene expression data obtained from blood samples of patients with bacterial or viral infections. The results showed a similar performance between the TAC approach and the state-of-the-art non-CNN machine learning classifiers in terms of accuracy when classifying gene expression data. Singh *et al.* [28] converted sensor dataset obtained from the floor surface pressure mapping into image data. Thereafter, a pre-trained CNN was applied on the image data and the results showed that the proposed method performed significantly better (by 10%) than the traditional machine learning methods. Note that none of these methods used the proposed conversion method that was undertaken in this current study. Furthermore, to the best of our knowledge, there is no other method in the literature which transforms tabular data into images in the way we proposed and applies explanation methods on predictions from 2D CNN.

Several studies have also focused on making CNNs interpretable in other domains. For instance, Tamajka *et al.* [29] used activations of the CNN as discreptors. Firstly the data was split into training and test sets, then the CNN was trained using a shallow network, the activations from the first pooling layer were used as a database of activations from known observations. Secondly, the test dataset was fed through a similar CNN and activations were extracted. Thirdly, to identify the class of each test set observation, its activation was compared to the activations of the training set samples and the class was assigned using a majority vote. This is similar to k-Nearest Neighbour approach and the authors used both cosine and euclidean as distance measures. The study used MNIST dataset and the results showed that the proposed interpretable model achieved almost similar results to the trained original CNN. Simonyan *et al.* [30] trained a DCNN on ILSVRC-2013 dataset which consists of 1.2 million images and two visualisation techniques were considered based on computing the gradient of the class score with respect to the input image. The finding was an establishment of the connection between the gradient-based CNN visualisation methods and deconvoltional networks. Liu *et al.* [31] applied CNN on MNIST dataset with the aim of interpreting the inner workings of the Fully-Connected (FC) layer. Firstly, the activations of the hidden layer of the FC layer were clustered to form factors. A factor is simply a label/class of a cluster. Secondly, identifications (IDs) were assigned to each cluster. Thirdly, the IDs were combined with the original data to form a meta-level data. Lastly, a decision tree was trained on meta-level data. For interpreting the hidden layer of the FC layer, the activations of the hypothesis class vs. true class were plotted on the x-y plane using all rows/depths of the decision tree. If there was an overlap between the activations, then this implied that there was no separation between the hypothesis class and the true class, this process was repeated until there was only one class or there was less overlap between the activations. The finding was that the proposed process was faithful to the original CNN model, i.e. accuracy was not compromised by the interpretable CNN.

III. PROPOSED FRAMEWORK

The proposed framework serves as a guideline for using 2D CNNs on tabular datasets. Firstly, this framework proposes that the feature values from tabular datasets should be mapped into different bins that are used to calculate weights of evidence and thereafter images to be created from the bins. After creating the images, the image dataset is split into training and test sets. Secondly, a 2D CNN needs to be trained on the training set and its performance to be assessed on the test set. Thirdly, the predictions from the 2D CNN need to be explained using state-of-the-art methods such as the Grad-CAM, SHAP values, Saliency Map and LIME. Fourthly, explanations must be validated using sanity check methodology [32]. Lastly, all explanation techniques must be compared quantitatively for the purpose of identifying the best performing explanation technique.

IV. METHODOLOGY

This section discusses the methodology undertaken in this study and its components in detail.

A. WEIGHT OF EVIDENCE

The *Weight of Evidence* (WOE) is used as a transformation technique for features in credit scoring [33]. The first step in calculating WOE is to create bins for each feature. Thereafter, the WOE is calculated for each bin. We denote the weight of evidence for bin b in feature f as $WOE_b^{(f)}$ which is calculated as follows

$$WOE_b^{(f)} = \ln\left(\frac{\%Dst.Gds_b^{(f)}}{\%Dst.Bds_b^{(f)}}\right), \tag{1}$$

where $Dst.Gds_b^{(f)}$ and $Dst.Bds_b^{(f)}$ are the distribution of good and bads for bin b in feature f computed according to Eq. (2) and Eq. (3), respectively.

$$Dst.Gds_b^{(f)} = \frac{\#ofgoods_b^{(f)}}{(\#ofgoods_b^{(f)} + \#ofbads_b^{(f)})} \tag{2}$$

$$Dst.Bds_b^{(f)} = \frac{\#ofbads_b^{(f)}}{(\#ofgoods_b^{(f)} + \#ofbads_b^{(f)})} \tag{3}$$

WOE computations are performed as follows [33]. Each bin is created in such a way that the bin contains at least 5% of data. This is to ensure that no bins are empty. There are no bins with 0 counts for goods and bads. This means that even if the data is not balanced, each bin will have both goods and bads and no bin will contain one type of class. The bins should have a monotonically decreasing or increasing relationship with the response/target variable. This is to assure that there are no reversals in the relationship between the features and the response variable. To crystallise this, suppose a feature age has a relationship with a default risk target variable. In general, younger people tend to be at a higher risk of defaulting on their loans compared to older people, and this implies that risk decreases monotonically with increasing age. Note that in this study we are not directly using WOE for converting tabular datasets into images but bins that are used to calculate WOE. The WOE are only used as inputs to information value calculation which is discussed in the following section.

Table 1 shows an example calculation of WOE. The `Age` feature is bucketed into different bins. For each bin, the distribution of goods and bads are calculated. Each bin has at least 5% of data and no bin contains 0 counts of either bads or goods. The goods are non-defaults and the bads are defaults. The WOEs are showing a decreasing trend. Later, we will show how we use bins for calculating WOE in order to convert our dataset into images. The bins for WOE calculations are optimal bins in terms of class distribution and also no bins contain single class information.

TABLE 1. Calculating Weights Of Evidence (WOE) for `Age` feature. The `Age` feature is bucketed into different bins. For each bin, the distribution of goods (Dst.Gds) and bads (Dst.Bds) are calculated. Each bin has at least 5% of data and no bin contains 0 counts of either bads or goods. The goods are non-defaults and the bads are defaults. The WOEs are showing a decreasing trend.

Age	Count	Dst.count	#of Goods	Dst.Gds	# of Bads	Dst.Bds	WOE
Missing	35	18.13%	30	18.75%	5	15.15%	21.31%
18 - 22	50	25.91%	42	26.25%	8	24.24%	7.96%
23 - 29	30	15.54%	25	15.63%	5	15.15%	3.08%
30 - 34	47	24.35%	38	23.75%	9	27.27%	-13.83%
34+	31	16.06%	25	15.63%	6	18.18%	-15.15%
Total	193		160		33		

B. INFORMATION VALUE

The Information Value (IV) is used to select predictive features and is calculated as follows for each feature f,

$$IV^{(f)} = \sum_{b=1}^{B^{(f)}} (\%Dst.Gds_b^{(f)} - \%Dst.Bds_b^{(f)}) \times WOE_b^{(f)}, \tag{4}$$

where $B^{(f)}$ represents the number of bins for feature f. The following thresholds apply as a general rule of thumb when using IV [33]:

TABLE 2. Bins for WOE calculations in (a) Australian dataset, (b) German dataset, and (c) HMEQ dataset. Legend for German dataset: `DM` (Deutsche Mark), `Acc Bal` (Account Balance), `Age` (years), `DCredit` (Duration of credit), `LEmploy` (Length of current employment), `Asset` (Most valuable available asset), `VStocks` (Value of savings/stocks) and `PStatus` (Payment status of previous credit). Legend for HMEQ dataset: `Loan` (Requested loan amount), `Mortdue` (Amount due on existing mortgage), `Value` (Value of current property), `Job` (Occupancy), `Derog` (Number of major derogatory reports), `Delinq` (Number of delinquent credit lines), `Clage` (Age of oldest credit line in months), `Ninq` (Number of recent credit enquiries) and `Debtinc` (Debt-to-income ratio).

A2	A3	A4	A5	A6	A7	A8	A9	A10	A12	A13	A14
[13.75,24.08]	[0,1.5]	g	aa	bb	[0,0.17]	f	f	[0,1]	g	[0,160]	[1,6]
[24.17,34.08]	[1.54,5.04]	gg	c	dd	[0.21,1]	t	t	[2,67]	p	[163,2000]	[7,100001]
[34.17,80.25]	[5.09,28.00]	p	cc	ff	[1.54,2.63]				s		
			d	h	[2.71,28.5]						
			e	j							
			ff	o							
			i	v							
			j	z							
			k								
			m								
			q								
			r								
			w								
			x								

(a)

Acc Bal	Age	DCredit	LEmploy	Asset	VStocks	PStatus
$\geq$ 200 DM	[19,33]	[13,24]	$\geq$ 7 year	Life insurance	100 $\leq \cdots <$ 500 DM	All credits duly paid
$<$ 200 DM	[34,75]	[26,72]	1 $\leq \cdots \leq$ 3 years	No assets	500 $\leq \cdots <$ 1000 DM	Account in arrears
Negative DM		[4,12]	4 $\leq \cdots \leq$ 6 years	Owns a car	No savings	Delay in payment
No account			Unemployed	Real estate	$<$ 100 DM	Existing credits duly paid
			$<$ 1 year		$\geq$ 1000 DM	No credit lines

(b)

Loan	Mortdue	Value	Job	Derog	Delinq	Clage	Ninq	Debtinc
[1100,12900]	[2063,42000]	[106450,855909]	Migrant worker	[0,1]	[0,1]	[0,90]	[0,1]	[0.52,31.92]
[13000,20800]	[42003,61712]	[72045,106431]	Office worker	[2,10]	[2,15]	[108,129]	[2,17]	[31.93,203.31]
[20900,89900]	[61767,88197]	[8000,72000]	Other			[130,166]		
	[88210,399550]		Professional Executive			[167,194]		
			Sales person			[195,226]		
			Self employed			[227,278]		
						[279,1168]		
						[91,107]		

(c)

R1) < 0.02: unpredictive;
R2) 0.02 to 0.1: weak predictor;
R3) 0.1 to 0.3: medium predictor;
R4) 0.3 to 0.5: strong predictor; and
R5) ≥ 0.5: suspicious or too good to be true.

Before converting our datasets into images, we first use IV to select predictive features, i.e., a feature f is selected if $IV^{(f)} \geq 0.1$.

C. CONVERSION OF TABULAR DATA INTO IMAGES

Once the data is transformed using bins for WOE calculation, we then create a *one-hot-encoding* transformation for each binned feature. Below we show a matrix example for each applicant, i.e. $\forall i \in \{1, 2, \cdots, N\}$ where N is the number of applicants:

$$\mathbf{x}^{(i)} = \begin{bmatrix} 0 & 0 & 1 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{B \times D}, \tag{5}$$

where $B = \max\{B^{(f)}\}$ which is a maximum number of bins for feature f.

Let $\mathbf{x}^{(i)}$ be a feature image representation of record $\mathbf{f}^{(i)} = [f_1^{(i)}, f_2^{(i)}, \ldots, f_D^{(i)}]$ and $y^{(i)}$ be a corresponding label for a dataset $\{\mathbf{f}^{(i)}, y^{(i)}\}_{i=1}^{N}$ of N records. The feature image is nothing but a sparse binary matrix (see Eq. (5)) of size $B \times D$ representing one-hot-encoding of each feature according to

$$\mathbf{x}^{(i)} = \left[E(f_1^{(i)}), E(f_2^{(i)}), \ldots, E(f_D^{(i)})\right], \tag{6}$$

where $E : f_j \mapsto \{0, 1\}^B$ is a function to perform one-hot-encoding for a feature f_j and B is defined as a maximum number of distinct bins of features, i.e. $B = \max\{B^{(f_j)}\}_{j=1}^{D}$. It is worth to note that for some features $B^{(f_j)} < B$ and in such cases, function $E(f_j)$ appends $(B - B^{(f_j)})$ zeros to one-hot-encoding representation of f_j. It is also worth to note that features become nominal with one-hot-encoding, hence some features in Table 2 do not follow a logical order. For example, the feature `Duration of Credit` in Table 2(b) has bin [13, 24] as the first bin and [4, 12] as the last bin. For this feature, one would expect the bins to be arranged in an increasing order, however it is not always the case with one-hot-encoding.

Each entry of $\mathbf{x}^{(i)}$ is then mapped into a pixel, where ones correspond to white pixels (and white pixels symbolise the presence of a bin) and zeros to black pixels (and black pixels symbolise the absence of a bin). Thus, each white pixel of an image represents a defined bin for WOEs calculations.

D. CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) [34] are widely known for solving tasks in image recognition, speech recognition and time series problems. The CNN consists of *convolutional layers*, *pooling layers* and *dense layers* (please see Figure 1). In what follows we briefly discuss layers of a CNN. For more details on CNNs please refer to our previous paper [2].

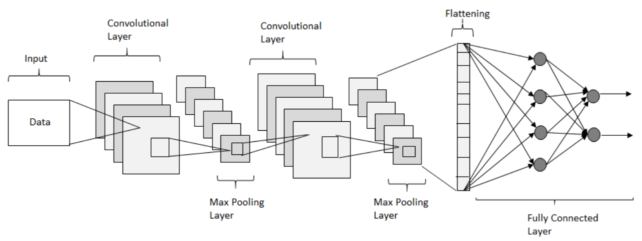

FIGURE 1. Schematic view of a Convolutional Neural Network [2].

1) INPUT

In image classification tasks, an input is a *tensor* of shape (*height*, *width*, *channel*). The channel represents a color of an image where 1 denotes a gray-scale image and 3 denotes an image with color/s.

2) CONVOLUTIONAL LAYER

Local patterns in image classification such as edges and textures are learned at convolutional layers [35]. Each convolutional layer consists of feature/activation maps that are responsible for learning different local patterns in an image. The feature maps are formed by sliding a learned *feature detector* also known as a learned filter on different parts of an image. Thereafter, a non-linear function such as the Rectified Linear Unit (ReLU) function is applied on feature maps.

3) POOLING LAYER

A pooling layer is responsible for downsizing feature maps by performing either max pooling or average pooling. The *spatial invariance* occurs at a pooling layer to assure that features are preserved irrespective of where they are located in an image.

4) FLATTENING

Flattening transforms all pooled feature maps into a single vector which acts as an input in a dense layer/fully connected layer.

5) CALCULATION IN EACH LAYER OF CNN

Normally, a deep learning neural network learns a set of parameters such as the weights during training. The parameters are learned in the convolutional layers and dense layers. To calculate the number of learnable parameters in the convolutional layers, we multiply the shapes of the filters such as the width w, height h, previous layer's filters p and filters of the current layer c resulting in

$$\#\text{ of parameters} = ((w \times h \times p) + 1) \times c), \tag{7}$$

where 1 is for the bias term, w is the shape of width of the filter, h is the shape of height of the filter, p is the number of filters in the previous layer and c is the number of filters in the current layer. In the convolutional layers, a feature map (i.e. a matrix of numbers) is produced by striding a filter along an input image (in the input layer) or along another feature map (in the convolutional layer). Each entry of the feature map is calculated as follows

$$K[r, c] = (f * t)[r, c] = \sum_{i=1}^{r} \sum_{j=1}^{c} t[i, j]f[r-i, c-j], \tag{8}$$

where r and c represent rows and columns, respectively, of a resulting feature map, f denotes an input image, t represents a kernel or a filter, $*$ denotes a convolutional operator. The feature map $\mathbf{A}^{(l-1)}$ is then multiplied by a weight matrix $\mathbf{W}^{(l)}$ and added with a bias term $\mathbf{b}^{(l)}$ to form

$$\mathbf{Z}^{(l)} = \mathbf{A}^{(l-1)} \cdot \mathbf{W}^{(l)} + \mathbf{b}^{(l)}, \tag{9}$$

and a ReLU function σ is applied on $\mathbf{Z}^{(l)}$ to propagate the feature maps into subsequent convolutional layers

$$\mathbf{A}^{(l-1)} = \sigma^{(l)}(\mathbf{Z}^{(l)}), \tag{10}$$

where l denotes the l-th convolutional layer. In the pooling layer, either a max pooling or mean pooling is applied to produce down-sized feature maps. Each entry of the pooled feature map is

$$P[i, j] = max[\mathbf{A}^{*}_{i,j} \subset \mathbf{A}^{(l)}]. \tag{11}$$

In the dense layer, the number of learn-able parameters is

$$\#\text{ of parameters} = ((c \times p) + 1 \times c), \tag{12}$$

where c denotes current layer neurons, p denotes previous layer neurons and 1 is for the bias term. For dense layers or fully connected layers, each neuron in the l-th layer receives signals from neurons of the previous layers $l-1$. The signals are multiplied by the corresponding weights and are summed together with a bias term and thereafter a transfer function is applied on the summed product to form an activation. Suppose $w_{ji}^{(l)}$ is a weight connection from neuron i to neuron j in the l-th layer. The activation $a_j^{(1)}$ of each neuron in the first hidden layer of the fully connected layers is

$$a_j^{(1)} = \sigma(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)}) \tag{13}$$

and the activation in the subsequent hidden layers is

$$a_k^{(l)} = \sigma(\sum_{j=1}^{n} w_{kj}^{(l)} a_j^{(l-1)} + b_k^{(l)}) \tag{14}$$

where d denotes the number of input features, n denotes the number of hidden neurons in hidden layer $l-1$ and σ is a transfer function (e.g sigmoid or ReLU function). In the output layer of the fully connected layers, a binary cross-entropy function is

$$H(p,q) = -\frac{1}{2}\sum_{i=1}^{2} y_i log(p(y_i)) + (1-y_i)log(q)), \quad (15)$$

where $p(y_i)$ is the predicted output, y_i is the actual target value and $q = 1 - p(y_i)$. The binary cross-entropy function is used to propagate errors backwards to update the weights of the CNN during training.

E. EXPLANATIONS

1) SALIENCY MAP

A Saliency Map [9] provides a visual explanation by highlighting important regions of an image that result in a prediction of a specific class. Specifically in this study, the image pixels correspond to bins that were created for WOE calculations. There are several variants of Saliency Map explanations, e.g., gradient explanation, gradient class activation map (Grad-CAM) explanation and layer-wise relevance propagation (LRP) explanation to mention a few. However, in this study we focused on gradient explanation and Grad-CAM. For an image $\mathbf{x}$ belonging to a class c, the Saliency Map $\mathbf{M}$ is computed as a derivative

$$\omega = \frac{\partial \hat{g}(\mathbf{x})}{\partial \mathbf{x}} \quad (16)$$

of a prediction $\hat{g}(\mathbf{x})$ with respect to an input image $\mathbf{x}$ where $\hat{g}$ is a cost function. Each pixel of $\mathbf{M}$ corresponds to the importance of the relative pixel of an input image [32], which is normally a class-specific score. The gradient measures how much a change in each input dimension would change the prediction $\hat{g}(\mathbf{x})$ in a small neighborhood around the input [32].

2) GRAD-CAM

Gradient weighted Class Activation Map (Grad-CAM) [36] is determined using a last convolutional layer of a CNN model. Firstly, a gradient of the score with respect to an activation map of the last convolutional layer is determined via a back-propagation approach

$$\frac{\partial y^c}{A^k},$$

where y^c is a score for class c and A^k is a feature map activation. Secondly, the neuron importance weights are calculated as follows

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{A_{ij}^k},$$

where $\frac{1}{Z}\sum_i\sum_j$ is a global average pooling and $Z = i \times j$. Lastly, each importance weight is multiplied by the corresponding feature map activation and the products are summed up to form a saliency map given as follows

$$\mathcal{L}^c_{Grad-CAM} = \sum_k \alpha_k^c A^k,$$

where $\mathcal{L}^c_{Grad-CAM} \in \mathbb{R}^{u\times v}$ and $u \times v$ represent the size of the feature map. The saliency map that is produced by the Grad-CAM highlights the important regions in an image that are corresponding to a predicted class.

3) LIME

Local Interpretable Model-Agnostic Explanations (LIME) [37] is used to explain predictions of black-box models. LIME uses for example a decision tree or a linear model around an instance of interest to explain the instance's class prediction [37]. The idea behind explaining predictions is to foster trust in predictions if any actions need to be taken. For example, if a credit analyst depends on a model for granting/rejecting a loan application, he/she will need to have a model's prediction explanation in order to trust the prediction and subsequently make a decision. We use LIME to create explanations as follows.

a: CREATE PERTURBATIONS OF AN IMAGE OF INTEREST:

Let $\mathbf{x}$ be an *original representation* of an image. Image $\mathbf{x}$ is then segmented into M super-pixels. A super-pixel is a group of pixels in an image. Let K be a number of perturbed images. A perturbation of image $\mathbf{x}$ is created by randomly turning on and off some of the super-pixels. This results into $\hat{\mathbf{X}} = \{0,1\}^{K\times M}$ where $M = B \times D$ (i.e. B is the maximum number of bins and D is the number of features) and each row of $\hat{\mathbf{X}}$ is an *interpretable representation* of each perturbed image. Here, a 1 represents that a super-pixel is on and a 0 represents that a super-pixel is off.

b: PREDICT CLASSES OF THE PERTURBED IMAGES:

A non-linear machine learning model g (which is in our case the model we have trained on the images) is used to predict a class of each perturbed image $\mathbf{x}^{(i)\prime}$. The prediction of $\mathbf{x}^{(i)\prime}$ is $\hat{g}(\mathbf{x}^{(i)\prime})$.

c: COMPUTE DISTANCES BETWEEN THE ORIGINAL IMAGE AND EACH OF THE PERTURBED IMAGES:

The distance between each perturbed image and the image being explained is computed using the cosine distance. The cosine distance is given as

$$D_i(\mathbf{x}, \mathbf{x}^{(i)\prime}) = \frac{\mathbf{x}\cdot\mathbf{x}^{(i)\prime}}{\|\mathbf{x}\|\|\mathbf{x}^{(i)\prime}\|}, \quad (17)$$

$\forall i = \{1,2,\cdots,K\}$ where $\mathbf{x} = \{1\}_{i=1}^{M}$ is the original image with all super-pixels enabled and $\mathbf{x}^{(i)\prime} = \{0,1\}_{i=1}^{M}$ are perturbed images.

d: COMPUTE THE WEIGHT FOR EACH PERTURBED IMAGE:

This is achieved by mapping each of the distances calculated above into a kernel function that has values between zero and one. This is shown by the formula below

$$\pi_{\mathbf{x}^{(i)\prime}} = \exp\big(-D_i(\mathbf{x}, \mathbf{x}^{(i)\prime})^2/\sigma^2\big), \tag{18}$$

where σ is the width of the kernel.

e: FIT A SPARSE LINEAR MODEL:

The interpretable representations $\hat{\mathbf{X}}$ of the perturbed images, the predictions $\hat{g}(\mathbf{x}^{(i)\prime})$ and the weights $\pi_{\mathbf{x}^{(i)\prime}}$ are then used to fit a sparse linear model. The sparse linear model is given as

$$\psi(\mathbf{x}^{(i)\prime}, \hat{g}(\mathbf{x}^{(i)\prime}), \pi_{\mathbf{x}^{(i)\prime}}) = \beta_0^{(i)} + \sum_{j=1}^{M} \beta_j^{(i)}(\pi_{\mathbf{x}^{(i)\prime}} x_j^{(i)\prime}), \tag{19}$$

$\forall i = \{1, 2, \cdots, K\}$ where M represents the number of super-pixels. Each coefficient $\beta_j^{(i)}$ in the sparse linear model corresponds to one super-pixel in the segmented image. The coefficients represent how important each super-pixel is for the prediction of the class of interest. Thus, the top super-pixels (in terms of the coefficient magnitudes) are used to explain the prediction made by the model g.

4) SHAP VALUES

The SHAP values [38] use an idea of coalition game theory, where there are players, a game and a payout. The aim of game theory is to distribute the payout between the players based on their contributions. In machine learning, the players are the feature values of a tabular dataset, the game is the prediction task, the payout is the prediction minus the average prediction for all dataset records. Hence, the SHAP value for any record is the average marginal contribution of a feature across all possible feature coalitions. Suppose we have a linear model and we want to explain a prediction for a record $\mathbf{f}^{(i)}$. The prediction for $\mathbf{f}^{(i)}$ is

$$\hat{g}(\mathbf{f}^{(i)}) = \beta_0 + \beta_1 f_1^{(i)} + \cdots + \beta_p f_d^{(i)}, \tag{20}$$

where $f_1^{(i)}, f_2^{(i)}, \cdots, f_d^{(i)}$ are feature values. Let ϕ_j be a contribution of feature j on prediction $\hat{g}(\mathbf{f}^{(i)})$. The contribution ϕ_j is given as

$$\begin{aligned} \phi_j(\hat{g}) &= \beta_j f_j^{(i)} - E(\beta_j \mathbf{F}_j) \\ &= \beta_j f_j^{(i)} - \beta_j E(\mathbf{F}_j), \end{aligned}$$

where $E(\mathbf{F}_j)$ is the expected value for feature j. If all feature contributions are summed up for one record, the result is

$$\begin{aligned} \sum_{j=1}^{d} \phi_j(\hat{g}) &= \sum_{j=1}^{d} (\beta_j f_j^{(i)} - E(\beta_j \mathbf{F}_j)) \\ &= (\beta_0 + \sum_{j=1}^{d} \beta_j f_j^{(i)}) - (\beta_0 + \sum_{j=1}^{d} E(\beta_j \mathbf{F}_j)) \\ &= \hat{g}(\mathbf{f}^{(i)}) - E(\hat{g}(\mathbf{F})), \end{aligned}$$

which is the difference between the predicted value for record $\mathbf{f}^{(i)}$ and the average predicted value for dataset $\mathbf{F} = \{\mathbf{F}_j\}_{j=1}^{d}$ where $\mathbf{F}_j = \{f_{ij}\}_{i=1}^{N}$.

For machine learning models, we determine M coalitions/combinations of feature values for a record $\mathbf{f}^{(i)}$. There are 2^d possible coalitions for feature values and this makes computations to be expensive and as a result we choose M such that $M \ll 2^d$. In order to assess contributions of each feature when explaining a prediction of a record, we make predictions for all M coalitions. Since we are using a subset of coalitions, the contribution of each feature is an estimate given as

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} (\hat{g}(\mathbf{f}_{+j}^{m}) - \hat{g}(\mathbf{f}_{-j}^{m})), \tag{21}$$

where $\hat{g}(\mathbf{f}_{+j}^{m})$ is the prediction for record $\mathbf{f}$, but with a random number of feature values replaced by some feature values from a random data point $\mathbf{z}$ in dataset $\mathbf{F}$ with the exception of the respective value for feature j. The prediction $\hat{g}(\mathbf{f}_{-j}^{m})$ is identical to $\hat{g}(\mathbf{f}_{+j}^{m})$ but the respective value of feature j is taken from a feature value of $\mathbf{z}$. Let

$$\mathbf{f} = (f_1, f_2, \cdots, f_d) \tag{22}$$

and

$$\mathbf{z} = (z_1, z_2, \cdots, z_d) \tag{23}$$

then

$$\mathbf{f}_{+j}^{m} = (f_1, \cdots, f_{j-1}, f_j, z_{j+1}, \cdots, z_d) \tag{24}$$

and

$$\mathbf{f}_{-j}^{m} = (f_1, \cdots, f_{j-1}, z_j, z_{j+1}, \cdots, z_d). \tag{25}$$

In general, Eq. (21) is

$$\phi_j(g) = \sum_{S \subseteq \{f_1, \ldots, f_d\} \setminus \{f_j\}} \frac{|S|!(d-|S|-1)!}{d!} (\hat{g}(S \cup \{f_j\}) - \hat{g}(S)), \tag{26}$$

where $d! = (d) \times (d-1) \times (d-2) \times \cdots \times (3) \times (2) \times (1)$ and S is a subset of feature values for record $\mathbf{f}$ that needs to be explained.

For images, a player is a group of features, for example, pixels can be grouped together to form super-pixels. Hence, each super-pixel represents a player.

V. EXPERIMENTS

A. DATA

Three real world credit datasets were used in this study, i.e. German, Australian [39] and Home Loan Equity (HMEQ) [40]. The number of samples, the number of features and the types of features for each dataset are shown in Table 3. All three datasets are publicly available credit scoring datasets on *UCI* repository and *Kaggle*. The first two datasets are the most used datasets in credit scoring literature [2]. The German credit dataset has 20 features of which 7 are numerical and 13 are categorical. These features include `status of existing checking account`, `duration in month`, `credit history` and `purpose`, to mention

TABLE 3. Credit datasets with sample sizes, number of defaults, number of non-defaults, number of categorical features and number of continuous features.

	Dataset		
	German	Australian	HMEQ
sample size	1,000	690	5,960
# of non-defaults	700	307	4,771
# of defaults	300	383	1,189
# of features	20	14	13
categorical	13	8	2
continuous	7	6	11

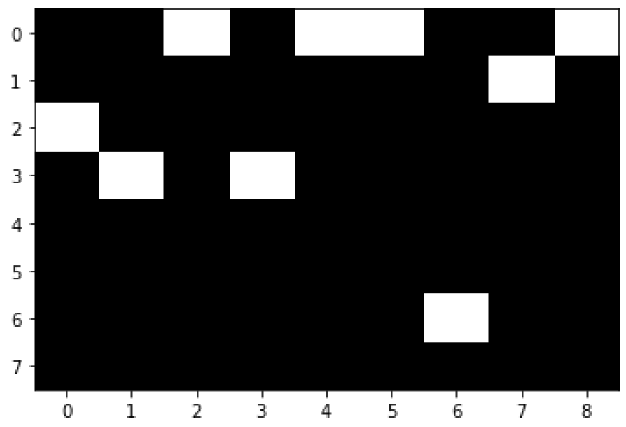

FIGURE 2. An example of applicant information is converted into a binary image for HMEQ tabular dataset. The columns represent features and the rows represent bins for WOE calculations.

a few. The Australian credit dataset has in total 14 features and 6 of the features are numerical and 8 are categorical. The original feature names have been changed for confidential purposes. The HMEQ dataset has 13 features of which 11 are numerical and 2 are categorical. The features include `amount of loan request`, `amount due on existing mortgage`, `value of the current property`, `years at present job`, `number of credit lines` and etc. The `response variable` for all three datasets is binary, i.e. applicants are classified either as "defaults" or "non-defaults". Table 2 shows WOE bins for each feature in German, Australian and HMEQ datasets. Note that unpredictive and weak features were removed according to the information value. Figure 2 shows an example where an applicant information is converted into an image, and in Figure 2 the columns represent features and the rows represent bins for WOE calculations.

B. MODEL

Each of the datasets was split into 70% training set and 30% test set following the common practise in the literature. The samples in each split were shuffled and thereafter stratified random sampling was performed for each split. The training set was used to determine the optimal parameters of the CNN model. The test set was used to assess the performance of the CNN model. Table 4, Table 5 and Table 6 show different architectures of the CNN models for Australian, German and HMEQ datasets, respectively. For Australian dataset, the CNN model was trained using 100 *epochs* and a *batch*

TABLE 4. CNN Architecture #1: Parameters and architectures of the CNN model in (a) Australian dataset, (b) German dataset, and (c) HMEQ dataset. Legend: Conv (Convolutional Layer), PL (Pooling Layer), FC (Fully Connected Layer).

Layer	Parameters and Architecture
Input	input shape: $k = 14$, $d = 12$, c=1
Conv1	filters:128, kernel size: 3×3, , dropout: 0.25
Conv2	filters: 256, kernel size: 3×3, dropout: 0.25
FC1	neurons: 64, dropout: 0.25
FC2	neurons: 2, activation: softmax
Activation functions	ReLU

(a)

Layer	Parameters and Architecture
Input	input shape:$k = 5$, $d = 7$, c=1
Conv1	filters:64, kernel size: 3×3, dropout: 0.25
Conv2	filters: 128, kernel size: 3×3, dropout: 0.25
Conv3	filters: 256, kernel size: 3×3, dropout: 0.25
PL	type: max, size: 2×2, dropout: 0.25
FC1	neurons: 128, dropout: 0.5
FC2	neurons: 2, activation: softmax
Activation functions	ReLU

(b)

Layer	Parameters and Architecture
Input	input shape:$k = 8$, $d = 9$, c=1
Conv1	filters:128, kernel size: 3×3, dropout: 0.25
Conv2	filters: 256, kernel size: 3×3, dropout: 0.25
FC1	neurons: 64, dropout: 0.25
FC2	neurons: 2, activation: softmax
Activation functions	ReLU

(c)

TABLE 5. CNN Architecture #2: Parameters and architectures of the second CNN model in (a) Australian dataset, (b) German dataset, and (c) HMEQ dataset. Legend: Conv (Convolutional Layer), PL (Pooling Layer), FC (Fully Connected Layer).

Layer	Parameters and Architecture
Input	input shape: $k = 14$, $d = 12$, c=1
Conv1	filters:16, kernel size: 3×3, dropout: 0.25
Conv2	filters: 32, kernel size: 3×3, dropout: 0.25
Conv3	filters: 64, kernel size: 3×3, dropout: 0.25
PL	type: max, size: 2×2
FC1	neurons: 32
FC2	neurons: 64, dropout: 0.5
FC3	neurons: 2, activation: softmax
Activation functions	Tanh

(a)

Layer	Parameters and Architecture
Input	input shape: $k = 5$, $d = 7$, c=1
Conv1	filters:16, kernel size: 3×3, dropout: 0.25
Conv2	filters: 32, kernel size: 3×3, dropout: 0.25
Conv3	filters: 64, kernel size: 3×3, dropout: 0.25
PL	type: max, size: 2×2
FC1	neurons: 64, dropout: 0.5
FC2	neurons: 2, activation: softmax
Activation functions	Tanh

(b)

Layer	Parameters and Architecture
Input	input shape: $k = 8$, $d = 9$, c=1
Conv1	filters:16, kernel size: 3×3, dropout: 0.25
Conv2	filters: 32, kernel size: 3×3, dropout: 0.25
Conv3 #2	filters: 64, kernel size: 3×3, dropout: 0.25
PL	type: max, size: 2×2, dropou=0.25
FC1	neurons: 32
FC2	neurons: 64, dropout: 0.5
FC3	neurons: 2, activation: softmax
Activation functions	Tanh

(c)

TABLE 6. CNN Architecture #3: Parameters and architectures of the third CNN model in (a) Australian dataset, (b) German dataset, and (c) HMEQ dataset. Legend: Conv (Convolutional Layer), PL (Pooling Layer), FC (Fully Connected Layer).

Layer	Parameters and Architecture
Input	input shape: $k = 14$, $d = 12$, c=1
Conv1	filters:8, kernel size: 3×3
Conv2	filters: 16, kernel size: 3×3
Conv3	filters: 32, kernel size: 3×3
FC1	neurons: 16
FC2	neurons: 32
FC3	neurons: 2, activation: softmax
Activation functions	Selu

(a)

Layer	Parameters and Architecture
Input	input shape: $k = 5$, $d = 7$, c=1
Conv1	filters:8, kernel size: 3×3,
Conv2	filters: 16, kernel size: 3×3
Conv3	filters: 32, kernel size: 3×3
FC1	neurons: 16
FC2	neurons: 32
FC3	neurons: 2, activation: softmax
Activation functions	Selu

(b)

Layer	Parameters and Architecture
Input	input shape: $k = 8$, $d = 9$, c=1
Conv1	filters:8, kernel size: 3×3
Conv2	filters: 16, kernel size: 3×3
Conv3	filters: 32, kernel size: 3×3
FC1	neurons: 16
FC2	neurons: 32
FC3	neurons: 2, activation: softmax
Activation functions	Selu

(c)

TABLE 7. CNN Architectures: Parameters and architectures for 1D CNN model: (i) 1D CNN Architecture #1, (ii) 1D CNN Architecture #2, Legend: Conv (Convolutional Layer), PL (Pooling Layer), FC (Fully Connected Layer).

Layer	Parameters for 1D CNN Architecture #1
Conv1	filters:128, kernel size: 1×3, dropout: 0.25
PL1	type: max, size: 1×2
Conv2	filters: 256, kernel size: 1×3, dropout: 0.25
PL2	type: max, size: 1×2
FC1	neurons: 256, dropout: 0.25
FC2	neurons: 2, activation: softmax
Activation functions	ReLU

(i)

Layer	Parameters for 1D CNN Architecture #2
Conv1	filters:16, kernel size: 1×3
Conv2	filters: 32, kernel size: 1×3
Conv3	filters: 64, kernel size: 1×3
FC1	neurons: 64
FC2	neurons: 32
FC3	neurons: 2, activation: softmax
Activation functions	Tanh

(ii)

size of 54. For both German and HMEQ datasets, the CNN model was trained with 100 epochs and a batch size of 64. The 2D CNN model was then compared to a 1D CNN model. The 1D CNN model was trained using 1000 epochs and a batch size of 32 on each dataset. The purpose of using a 1D CNN model is to see the effect of the model when the data is not converted into images. For 1D CNN model, we used the raw tabular dataset. Hence, Table 7 shows two different architectures for a 1D CNN model. Please note that 2D CNNs are not applicable to raw tabular datasets, hence we opted for a 1D CNN model.

C. PREDICTION PERFORMANCE MEASURES

For each of the datasets, we calculated "Accuracy" which measures the proportion of the correctly classified examples and is defined as

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + FN + TN)}. \tag{27}$$

We also calculated other performance metrics which are defined next. The "Area Under the Curev" (AUC) [15] which is a metric that measures the predictive power of a model is calculated as follows

$$\text{AUC} = \frac{1}{2}\Big(1 + \frac{TP}{TP + FN} - \frac{FP}{FP + TN}\Big). \tag{28}$$

The higher the AUC, the better the predictive power of the model. The "Brier Score", named after Glenn Brier [41], calculates the mean squared error between the predicted probabilities and their respective true class values and is calculated as follows

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2, \tag{29}$$

where $\hat{y}_i$ is the predicted probability of the true class, y_i is the true class and N is the number of records. The Brier Score takes values between 0 and 1. The lower the Brier Score, the better the predictions are calibrated. On the other hand the "H-measure" proposed by Hand [42] assesses the cost of misclassification and it uses a severity ratio (SR) given as

$$\text{SR} = \frac{c_0}{c_1}, \tag{30}$$

where $c_0 > 0$ is the cost of misclassifying a class 0 record as class 1. Hence, the higher the H-measure the lower the misclassification cost.

D. EXPLANATIONS

After training the CNN models and assessing their performances, explanation methods such as LIME, SHAP values, Grad-CAM and Saliency Map were used to explain individual predictions. For LIME, we set the kernel width $\sigma^2 = 0.25$ and the number of perturbed images $K = 150$. This study focused on *local explanations* (i.e. individual explanation) as opposed to *global explanations* (i.e. explaining holistically how the model makes predictions).

VI. RESULTS

A. MODEL PERFORMANCE

We created three CNN architectures (please refer to Table 4, Table 5 and Table 6) to perform contrast experiments on the design of the architectures. For each architecture we used different activation functions, different number of layers and different number of filters in each layer. Also, Table 8, Table 9 and Table 10 show confusion matrices, AUC, Brier Score and H-measure for each of the CNN architectures on all three

TABLE 8. Performance metrics resulted from the CNN Architecture #1 model for all three datasets for both training and test sets.

Dataset	Test (30%)		Train (70%)	
German	TP = 72	FN = 18	TP = 173	FN = 37
	FP = 19	TN = 191	FP = 36	TN = 454
	Accuracy = 0.88		Accuracy = 0.90	
	AUC = 0.80		AUC =0.79	
	Brier Score = 0.21		Brier Score = 0.22	
	H-measure = 0.42		H-measure = 0.42	
Australian	TP = 111	FN = 4	TP = 260	FN = 8
	FP = 6	TN = 86	FP = 8	TN = 207
	Accuracy = 0.95		Accuracy = 0.97	
	AUC = 0.96		AUC = 0.96	
	Brier Score = 0.54		Brier Score = 0.54	
	H-measure = 0.89		H-measure = 0.88	
HMEQ	TP = 1181	FN = 250	TP =2742	FN = 598
	FP = 69	TN = 288	FP = 145	TN = 687
	Accuracy = 0.82		Accuracy = 0.82	
	AUC = 0.83		AUC = 0.82	
	Brier Score = 0.16		Brier Score = 0.18	
	H-measure = 0.44		H-measure = 0.44	

TABLE 9. Performance metrics resulted from the CNN Architecture #2 model for all three datasets for both training and test sets.

Dataset	Test (30%)		Train (70%)	
German	TP = 41	FN = 49	TP =147	FN =63
	FP = 34	TN =176	FP = 18	TN = 472
	Accuracy = 0.72		Accuracy = 0.88	
	AUC = 0.65		AUC = 0.83	
	Brier Score = 0.21		Brier Score = 0.20	
	H-measure = 0.11		H-measure =0.54	
Australian	TP = 98	FN = 17	TP = 250	FN = 18
	FP = 14	TN = 78	FP = 6	TN = 209
	Accuracy = 0.85		Accuracy = 0.95	
	AUC = 0.85		AUC = 0.95	
	Brier Score = 0.50		Brier Score = 0.50	
	H-measure = 0.54		H-measure = 0.85	
HMEQ	TP =1398	FN =33	TP =	FN = 42
	FP = 208	TN = 149	FP = 411	TN = 421
	Accuracy = 0.87		Accuracy = 0.89	
	AUC = 0.70		AUC = 0.75	
	Brier Score = 0.13		Brier Score = 0.11	
	H-measure = 0.41		H-measure = 0.41	

datasets. Amongst the three tables, Table 8 which corresponds to CNN Architecture #1 shows better performances in almost all three datasets. Hence, our proposed method is based on CNN Architecture #1. The results show that the proposed method achieves similar performances on both training and test set, hence it does achieve a good generalisation performance. We compared the 2D CNN model with 1D CNN model to see the impact of converting data into images. Table 11 and Table 12 show the results of the 1D CNN model. It can be seen that when comparing the results in Table 8 with the results in Table 11 and Table 12, the 2D CNN model performs better than the 1D CNN model on German and Australian datasets when looking at all the performance metrics. However, on the HMEQ datasets, both 2D CNN and 1D CNN perform similarly regarding accuracies and brier scores, but 2D CNN show better results when looking at the AUC and the H-measure. These results support the conversion of tabular datasets into images in order to leverage the high performance nature of 2D CNNs.

TABLE 10. Performance metrics resulted from the CNN Architecture #3 model for all three datasets for both training and test sets.

Dataset	Test (30%)		Train (70%)	
German	TP = 49	FN = 41	TP =181	FN =29
	FP = 40	TN =170	FP = 15	TN = 475
	Accuracy = 0.73		Accuracy = 0.94	
	AUC = 0.68		AUC = 0.92	
	Brier Score = 0.26		Brier Score = 0.25	
	H-measure = 0.15		H-measure = 0.75	
Australian	TP =103	FN =12	TP =260	FN =8
	FP =13	TN =79	FP =3	TN =212
	Accuracy = 0.88		Accuracy = 0.98	
	AUC =0.88		AUC =0.98	
	Brier Score =0.52		Brier Score = 0.53	
	H-measure = 0.62		H-measure = 0.93	
HMEQ	TP =1370	FN = 61	TP =3296	FN = 44
	FP = 184	TN =173	FP = 284	TN = 548
	Accuracy = 0.86		Accuracy = 0.92	
	AUC = 0.72		AUC = 0.82	
	Brier Score = 0.13		Brier Score = 0.08	
	H-measure = 0.57		H-measure = 0.57	

TABLE 11. Performance metrics resulted from the 1D CNN Architecture #1 model for all three datasets for both training and test sets.

Dataset	Test (30%)		Train (70%)	
German	TP = 48	FN = 42	TP = 116	FN = 94
	FP = 41	TN = 169	FP = 64	TN = 426
	Accuracy = 0.72		Accuracy = 0.77	
	AUC = 0.67		AUC = 0.71	
	Brier Score = 0.28		Brier Score = 0.25	
	H-measure = 0.10		H-measure = 0.27	
Australian	TP = 96	FN = 19	TP = 250	FN = 18
	FP = 23	TN =69	FP = 14	TN = 201
	Accuracy = 0.79		Accuracy = 0.93	
	AUC = 0.79		AUC = 0.93	
	Brier Score = 0.20		Brier Score = 0.06	
	H-measure = 0.41		H-measure = 0.74	
HMEQ	TP = 1415	FN = 16	TP = 3272	FN = 68
	FP = 310	TN = 47	FP = 684	TN = 148
	Accuracy = 0.82		Accuracy = 0.82	
	AUC = 0.56		AUC = 0.56	
	Brier Score = 0.18		Brier Score = 0.18	
	H-measure = 0.07		H-measure = 0.08	

The results that were found in literature which are shown in Table 13 were compared with the proposed method results. Hence, the proposed method shows in most cases better performances compared to the methods from the literature considered in this paper. The results in Table 13 and Table 8 support the efficacy of the proposed 2D CNN model in accurately assessing the credit applicants. However, we also want to make sure that we have meaningful and good explanations for the proposed model's decision. In the following, we qualitatively and quantitatively evaluate the explanations of the proposed method generated by four different explanation methods (i.e LIME, Saliency Map, Grad-CAM and SHAP values).

B. EXPLANATIONS

For each of the datasets, we randomly selected an image representing the corresponding record in the tabular dataset. Thereafter, we predicted each class for each of the randomly

TABLE 12. Performance metrics resulted from the 1D CNN Architecture #2 model for all three datasets for both training and test sets.

Dataset	Test (30%)		Train (70%)	
German	TP = 32	FN = 58	TP = 75	FN =135
	FP = 22	TN = 188	FP = 22	TN = 468
	Accuracy = 0.73		Accuracy = 0.78	
	AUC = 0.63		AUC = 0.66	
	Brier Score = 0.27		Brier Score = 0.22	
	H-measure = 0.13		H-measure = 0.19	
Australian	TP = 92	FN = 23	TP = 226	FN = 42
	FP = 28	TN = 64	FP = 39	TN = 176
	Accuracy = 0.75		Accuracy = 0.83	
	AUC = 0.75		AUC = 0.83	
	Brier Score = 0.25		Brier Score = 0.17	
	H-measure = 0.34		H-measure = 0.66	
HMEQ	TP =1422	FN = 9	TP = 3294	FN =46
	FP =323	TN = 34	FP = 739	TN = 93
	Accuracy = 0.81		Accuracy = 0.81	
	AUC = 0.54		AUC = 0.55	
	Brier Score = 0.19		Brier Score = 0.19	
	H-measure = 0.06		H-measure = 0.05	

TABLE 13. Performances of related models in credit scoring literature.

Source	Year	German	Astralian	HMEQ
		Accuracy	Accuracy	Accuracy
[43]	2000	0.76	0.87	-
[19]	2016	0.75	0.86	-
[44]	2016	0.78	0.88	-
[45]	2017	0.77	0.88	-
[46]	2018	0.77	0.87	-
[22]	2018	0.86	0.93	
[23]	2018	0.87	**0.95**	
[21]	2020	0.81	0.90	
Our Study	2021	**0.88**	**0.95**	0.82

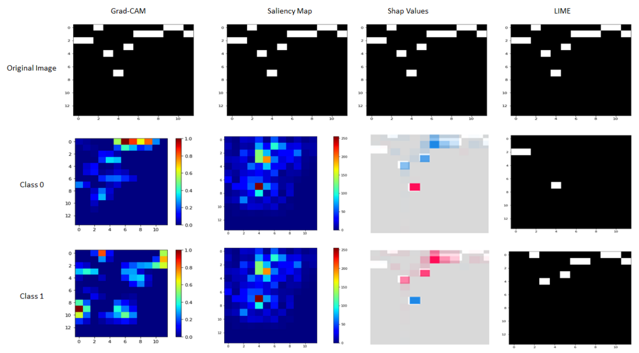

FIGURE 3. The rows in this figure refer to the original image, class 0 and class 1 and the columns refer to the Grad-CAM, Saliency Map, Shap values and LIME. Prediction explanation for a random image belonging to class 0 on Australian dataset. Legend: class 0 (non-defaults), class 1 (defaults), column 0 (`A2`), column 1 (`A3`), column 2 (`A4`), column 3 (`A5`), column 4 (`A6`), column 5 (`A7`), column 6 (`A8`), column 7 (`A9`), column 8 (`A10`), column 9 (`A12`), column 10 (`A13`) and column 11 (`A14`). All the (A's) are feature names. All columns are features from tabular dataset and the rows are bins for WOE calculations. See Table 2 for further details on features.

selected images. We then used Grad-CAM, LIME, SHAP values and Saliency Map to explain each predicted class (i.e. defaults = 1 and non-defaults = 0). Figure 3, Figure 4 and Figure 5 show explanations for each of the predictions. The white blocks/pixels in LIME are super-pixels that are

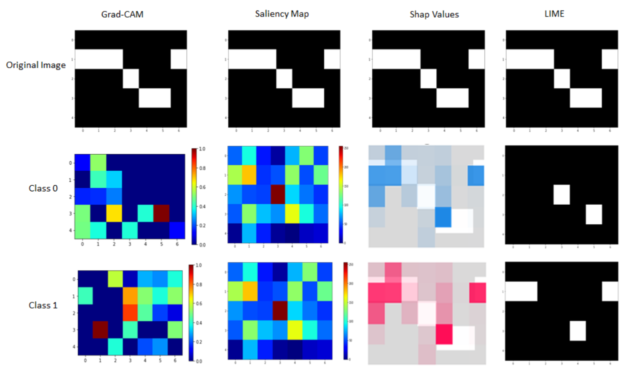

FIGURE 4. The rows in this figure refer to the original image, class 0 and class 1 and the columns refer to the Grad-CAM, Saliency Map, Shap values and LIME. Prediction explanation for a random image belonging to class 1 on German dataset. Legend: class 0 (non-defaults), class 1 (defaults), column 0 (`Acc Bal`), column 1 (`Age`), column 2 (`DCredit`), column 3 (`LEmploy`), column 4 (`Asset`), column 5 (`VStocks`) and column 6 (`PStatus`). All columns are features from tabular dataset and the rows are bins for WOE calculations. See Table 2 for further details on features.

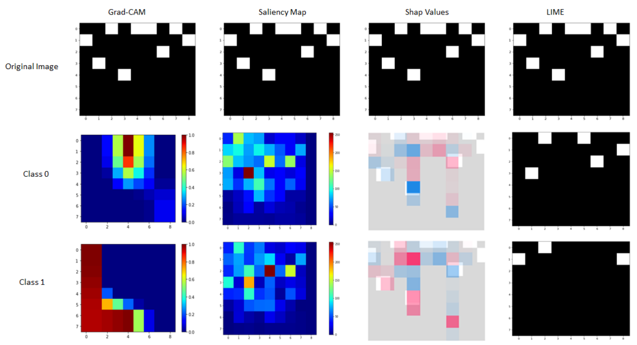

FIGURE 5. The rows in this figure refer to the original image, class 0 and class 1 and the columns refer to the Grad-CAM, Saliency Map, Shap values and LIME. Prediction explanation for a random image belonging to class 1 on HMEQ dataset. Legend: class 0 (non-defaults), class 1 (defaults), column 0 (`Loan`), column 1 (`Mortdue`), column 2 (`Value`), column 3 (`Job`), column 4 (`Derog`), column 5 (`Delinq`), column 6 (`Clage`), column 7 (`Ninq`) and column 8 (`Debtinc`). All columns are features from tabular dataset and the rows are bins for WOE calculations. See Table 2 for further details on features.

important in making a prediction of a class. For Saliency Map, the color-bar shows which pixels are important in making a prediction of a class. For SHAP values, the red pixels contribute more in making a prediction of a class and the blue pixels contribute less. To explain in an understandable form to a human, we use Table 2 where each bin corresponds to a pixel.

For example, LIME in Figure 5 explains the reason why a customer is flagged as a defaulted customer. The reason is that the amount due on existing mortgage is at least \$88, 210, the value of the current property is at least \$106, 450, the number of credit default lines is at most 1, the age of the oldest credit line is between 130 and 166 months and the debt to income ratio is at least 31%. We can interpret this explanation as follows: the customer has defaulted once, he/she still owes a huge amount on his/her mortgage, almost a third of his/her salary/income pays debt and he/she has one credit line that is older than 11 years. Saliency Map in Figure 5 explains that the oldest credit line for the same customer is between 130 and 166 months, the amount due on existing mortgage is less than $42, 000 and the value of the current property is less than $72, 000. SHAP values in Figure 5 explain the following: the amount due on existing mortgage is at least $88, 210 for the customer and the customer is doing an office work, the value of the existing property is at least $106, 450 and the oldest credit line is between 279 and 1168 months. Grad-CAM in Figure 5 explains that the customer requested a loan of atleast $13, 000 and the amount due on existing property is atleast $88, 210.

It is worth to mention that for Australian credit dataset all feature names and values were anonymized to protect confidentiality of the data. Hence, we can't really explain Figure 3. Next, we explain Figure 4 where the correct class is a defaulted customer. Starting with LIME, the defaulted customer is 34 years or older, owns a real estate, his/her account is in arrears and his/her account balance is less than 200 Deutsche Marks. The same defaulted customer is explained by the Saliency Map as 34 years or older and unemployed. For SHAP values, the customer is 34 years and older, account in arrears, owns a real estate and the account balance is less than 200 Deutsche Marks. For Grad-CAM in Figure 4, the customer has a balance that is less than 200 Deutsche Marks and she/he has been working for atleast 4 years but less than 7 years.

This is an illustration of how one would explain the predictions using the corresponding bins in Table 2. Please note that we did padding on the images to compensate for empty bins, if an explanation highlights a pixel that falls on a padding, we ignore the importance of that pixel because it carries no meaning.

C. SANITY CHECK FOR EXPLANATIONS

Once an explanation is determined for a prediction, the next thing to do is to assess its validity. To do this, we followed a process suggested in [32]. Firstly, a model is trained on the original labels and an explanation is provided to explain a prediction of image **x**. Secondly, another model having the exact architecture as the previous model is trained on a copy of the original data but the labels are randomly permuted. The reason for randomizing the labels is to break the relationship that exists between the input features and the labels [32]. An explanation is then provided to explain a prediction of a randomly permuted label for image **x**. Thirdly, the explanations on both scenarios for image **x** are compared. If the explanation for image **x** did not change after randomly permuting the labels, then the explanation on the original labels did not explain anything about the relationship between the inputs and the prediction. The visual comparison of the explanations could be misleading, to ascertain that the explanations are the same/differ, a quantitative measure such as the *Spearman rank correlation* is used. The Spearman rank correlation is calculated as

TABLE 14. Spearman rank correlations to quantitatively perform sanity check for Figure 3, Figure 4 and Figure 5.

	Dataset		
Explanation	Australian	German	HMEQ
Saliency Map	$\rho = 0.85$	$\rho = 0.21$	$\rho = 0.90$
LIME	$\rho = -0.04$	$\rho = 0.69$	$\rho = 0.06$
SHAP values	$\rho = -0.25$	$\rho = 0.11$	$\rho = -0.60$
Grad-CAM	$\rho = -0.35$	$\rho = 0.13$	$\rho = -0.50$

$$\rho = 1 - \frac{6 \times \sum_i^l m_i^2}{l(l^2-1)} \tag{31}$$

where l is a number of categories that need to be ranked and $m = Rank(\mathbf{x})_o - Rank(\mathbf{x})_p$ is the difference between the rankings of the explanations on the original and the permuted labels, respectively.

Table 14 shows correlations for each explanation technique for Figure 3, Figure 4 and Figure 5 on each dataset. For Australian data, LIME shows a weaker correlation. For German data, SHAP values and Grad-CAM show a weaker correlation. For HMEQ, LIME shows a weaker correlation. According to [32], weaker correlations mean that the explanations do pass the sanity check.

D. QUANTITATIVE COMPARISON OF EXPLANATION METHODS

This section looks at comparing all four explanation methods quantitatively. Suppose $\mathcal{L}_R$ is an xgboost model trained on raw dataset and $\mathcal{L}_E$ is another xgboost model trained on explanations of predictions from the model. The performance of $\mathcal{L}_R$ on raw test set is denoted as P_R and the performance on explanations test set is denoted as P_E. We hypothesise that for good explanations, we expect $P_E > P_R$. Note that the performance metric we are using is the accuracy which is a proportion of the correctly classified records. We tested our hypothesis in all three datasets and we found that for all four explanations $P_{E_i} > P_R, \forall i \in$ {Grad-CAM, Saliency Map, LIME, SHAP values}. Thereafter, we compared performances of the explanations methods used in this study. The best performing explanation method will have $P_{E_i} > P_{E_j}, \forall j$ where $j \neq i$. To conduct the comparisons, firstly we split each raw dataset into 70% training and 30% test sets. To avoid confusion, for each dataset we only performed a single data split, where we randomly shuffled the records and performed stratified sampling. Secondly, we predict the class of each record for each of the datasets (both training and test sets) using our CNN model. Thirdly, we explain each of the predictions using Grad-CAM, Saliency Map, LIME and SHAP values. Fourthly, we train $\mathcal{L}_R$ on 70% raw dataset and assess its performance P_R on 30% raw test set. Thereafter, we train $\mathcal{L}_E$ on explanations of the predictions from the 70% raw dataset and assess the performance P_E on explanations of the predictions from the 30% raw test set. Table 15 shows performances of the explanation methods together with performances on the raw datasets

TABLE 15. Performances on the 30% test sets based on raw dataset and explanations of predictions from the CNN model. Accuracy is denoted as Acc.

	Performance				
Dataset	Raw	Saliency Map	LIME	SHAP values	Grad-CAM
German	Acc = 0.70	Acc = 0.76	Acc = 0.74	Acc = 0.99	Acc = 0.91
	AUC = 0.63	AUC= 0.65	AUC = 0.64	AUC = 0.98	AUC = 0.87
	Brier Score = 0.31	Brier Score = 0.25	Brier Score = 0.26	Brier Score= 0.02	Brier Score = 0.09
	H-measure = 0.08	H-measure= 0.13	H-measure = 0.12	H-measure = 0.93	H-measure = 0.63
Australian	Acc=0.83	Acc = 0.90	Acc = 0.91	Acc = 1.00	Acc = 0.93
	AUC = 0.82	AUC = 0.85	AUC = 0.87	AUC =1.00	AUC = 0.93
	Brier Score = 0.17	Brier Score = 0.15	Brier Score =0.13	Brier Score =0.00	Brier Score = 0.07
	H-measure = 0.48	H-measure = 0.54	H-measure =0.61	H-measure = 1.00	H-measure = 0.78
HMEQ	Acc = 0.81	Acc = 0.84	Acc = 0.84	Acc = 1.00	Acc = 0.94
	AUC = 0.76	AUC = 0.63	AUC = 0.64	AUC = 1.00	AUC = 0.93
	Brier Score = 0.13	Brier Score = 0.15	Brier Score = 0.16	Brier Score = 0.00	Brier Score = 0.05
	H-measure = 0.40	H-measure= 0.20	H-measure = 0.21	H-measure = 1.00	H-measure=0.80

when using xgboost model. Note that the performances were based on 30% test set. In all three datasets, the SHAP values improve accuracy to 99% and 100% and these are state-of-the-art results for all three datasets. This shows that the SHAP values may be used for explanations as well as for feature selection for improving model performances.

VII. CONCLUSION AND FUTURE WORK

In this paper, tabular datasets were converted into images using bins that were used to calculate weights of evidence. Each pixel of a feature image corresponds to a feature bin. The purpose of using weights of evidence was to create meaningful bins that are monotonic to the response variable. For instance, in credit scoring, a feature such as `age` should decrease monotonically with a default risk response variable and this means that the younger applicants are more riskier than the older applicants in terms of defaulting on their loans. Hence, the risk of defaulting decreases monotonically with increasing age. The other purpose for using bins that were used to calculate weights of evidence in this current study is that both continuous and categorical features were accommodated. The images were then used to train 2D convolutional neural networks. The performances of the trained convolutional neural networks were compared with literature results. We found that the trained convolutional neural networks performed better when compared to the literature results. However, the aim of our study was not only on model performance but rather on explaining predictions. The predictions of the models were then explained using Grad-CAM, LIME, SHAP values and Saliency Map. Each of these explanation methods highlight important regions/pixels in an image that correspond to the output/prediction class. To explain in an understandable form, all important pixels were linked to their respective bins that were used to calculate weights of evidence. The explanation techniques were validated using sanity check methodology. Furthermore, performances of explanation methods were compared quantitatively by using an xgboost model. The explanation method which performed better compared to other methods in all three datasets was SHAP values. The purpose of the xgboost model was to evaluate quantitatively the performances of the explanation methods. However, the future research should focus on validating and evaluating performances of the explanations by using domain experts in the field of credit scoring such as the credit risk analysts or managers. Also, future work should focus on comparing the performances of the methods that perform tabular data conversion into images. Further, for future work the proposed framework can be extended to other deep learning architectures such as ResNet, Deep Graph Neural Networks and others.

REFERENCES

[1] D. C. Sowers and D. Durand, "Risk elements in consumer instalment financing," *J. Marketing*, vol. 6, no. 4, p. 407, Apr. 1942.

[2] X. Dastile, T. Celik, and M. Potsane, "Statistical and machine learning models in credit scoring: A systematic literature survey," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106263.

[3] X. Liu, X. Wang, and S. Matwin, "Interpretable deep convolutional neural networks via meta-learning," *CoRR*, vol. abs/1802.00560, pp. 1–5, Jul. 2018.

[4] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017, *arXiv:1702.08608*. [Online]. Available: http://arxiv.org/abs/1702.08608

[5] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *Proc. IEEE 5th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2018, pp. 80–89.

[6] D. Rothman, *Hand-on Explainable AI (XAI) with Python*. Birmingham, U.K.: Packt, 2020.

[7] *Basel II: International Convergence of Capital Measurement and Capital Standards: A Revised Framework—Comprehensive Version, Bank for International Settlements*, Basel Committee on Banking Supervision, Basel, Switzerland, 2006.

[8] GDPR. (2018). *Reform of Data Protection Rules*. Accessed: Sep. 17, 2020. [Online]. Available: https://www.legislation.gov.uk/ukpga/2018/12/part/2/chapter/2/enacted

[9] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Mülle, Eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, doi: 10.1007/978-3-030-28954-6.

[10] L. Breiman, "Statistical modeling: The two cultures," *Stat. Sci.*, vol. 16, no. 3, pp. 199–215, 2001.

[11] W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 5–22, doi: 10.1007/978-3-030-28954-6_1.

[12] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Mach. Intell.*, vol. 1, no. 5, pp. 206–215, May 2019.

[13] S. Chari, D. M. Gruen, O. Seneviratne, and D. L. McGuinness, "Directions for explainable knowledge-enabled systems," 2020, *arXiv:2003.07523*. [Online]. Available: https://arxiv.org/abs/2003.07523

[14] B. Zhu, W. Yang, H. Wang, and Y. Yuan, "A hybrid deep learning model for consumer credit scoring," in *Proc. Int. Conf. Artif. Intell. Big Data (ICAIBD)*, May 2018, pp. 205–208.
[15] J. M. Tomczak, "Classification restricted Boltzmann machine for comprehensible credit scoring model," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1789–1796, Mar. 2015.
[16] Y. Li, X. Lin, X. Wang, F. Shen, and Z. Gong, "Credit risk assessment algorithm using deep neural networks with clustering and merging," in *Proc. 13th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2017, pp. 173–176.
[17] V.-E. Neagoe, A.-D. Ciotec, and G.-S. Cucu, "Deep convolutional neural networks versus multilayer perceptron for financial prediction," in *Proc. Int. Conf. Commun. (COMM)*, Jun. 2018, pp. 201–206.
[18] S. Hamori, M. Kawai, T. Kume, Y. Murakami, and C. Watanabe, "Ensemble learning or deep learning? Application to default risk analysis," *J. Risk Financial Manage.*, vol. 11, no. 1, p. 12, Mar. 2018.
[19] S. Ha and H.-N. Nguyen, "Credit scoring with a feature selection approach based deep learning," *MATEC Web Conf.*, vol. 54, p. 05004, Dec. 2016.
[20] J. Sirignano, A. Sadhwani, and K. Giesecke, "Deep learning for mortgage risk," *SSRN Electron. J.*, 2018, doi: 10.2139/ssrn.2799443.
[21] D. Tripathi, D. R. Edla, V. Kuppili, and A. Bablani, "Evolutionary extreme learning machine with novel activation function for credit scoring," *Eng. Appl. Artif. Intell.*, vol. 96, Nov. 2020, Art. no. 103980.
[22] D. R. Edla, D. Tripathi, R. Cheruku, and V. Kuppili, "An efficient multi-layer ensemble framework with BPSOGSA-based feature selection for credit scoring data analysis," *Arabian J. Sci. Eng.*, vol. 43, no. 12, pp. 6909–6928, Dec. 2018.
[23] D. Tripathi, D. R. Edla, and R. Cheruku, "Hybrid credit scoring model using neighborhood rough set and multi-layer ensemble classification," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1543–1549, Mar. 2018.
[24] M. J. Ariza-Garzon, J. Arroyo, A. Caparrini, and M. Segovia-Vargas, "Explainability of a machine learning granting scoring model in peer-to-peer lending," *IEEE Access*, vol. 8, pp. 64873–64890, 2020.
[25] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Sci. Rep.*, vol. 9, no. 1, Dec. 2019, Art. no. 11399.
[26] C.-L. Yang, Z.-X. Chen, and C.-Y. Yang, "Sensor classification using convolutional neural network by encoding multivariate time series as two-dimensional colored images," *Sensors*, vol. 20, no. 1, p. 168, Dec. 2019.
[27] L. Buturović and D. Miljković, "A novel method for classification of tabular data using convolutional neural networks," Cold Spring Harbor Lab., Cold Spring Harbor, NY, USA, May 2020, doi: 10.1101/2020.05.02.074203.
[28] M. S. Singh, V. Pondenkandath, B. Zhou, P. Lukowicz, and M. Liwicki, "Transforming sensor data to the image domain for deep learning—An application to footstep detection," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 2665–2672.
[29] M. Tamajka, W. Benesova, and M. Kompanek, "Transforming convolutional neural network to an interpretable classifier," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jun. 2019, pp. 255–259.
[30] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proc. 2nd Int. Conf. Learn. Represent.*, 2014, pp. 1–8.
[31] X. Liu, X. Wang, and S. Matwin, "Interpretable deep convolutional neural networks via meta-learning," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2018, pp. 1–9.
[32] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2018, pp. 9525–9536.
[33] N. Siddiqi, *Credit Risk Scorecards: Developing And Implementing Intelligent Credit Scoring*. Chennai, India: SAS, 2005.
[34] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[35] F. Chollet, *Deep Learning With Python*, 1st ed. Greenwich, CT, USA: Manning, 2017.
[36] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.
[37] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?' Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, pp. 1135–1144, Aug. 2016.
[38] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, pp. 1–10, May 2017.
[39] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[40] Kaggle. (2017). *Home Loan Equity Data*. Accessed: Jun. 3, 2020. [Online]. Available: https://www.kaggle.com/ajay1735/hmeq-data
[41] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Rev.*, vol. 78, no. 1, pp. 1–3, Jan. 1950.
[42] D. J. Hand, "Measuring classifier performance: A coherent alternative to the area under the ROC curve," *Mach. Learn.*, vol. 77, no. 1, pp. 103–123, Oct. 2009.
[43] D. West, "Neural network credit scoring models," *Comput. Oper. Res.*, vol. 27, nos. 11–12, pp. 1131–1152, Sep. 2000.
[44] M. Ala'raj and M. F. Abbod, "Classifiers consensus system approach for credit scoring," *Knowl.-Based Syst.*, vol. 104, pp. 89–105, Jul. 2016.
[45] Y. Xia, C. Liu, Y. Li, and N. Liu, "A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring," *Expert Syst. Appl.*, vol. 78, pp. 225–241, Jul. 2017.
[46] L. Munkhdalai, O.-E. Namsrai, and K. Ryu, "Credit scoring with deep learning," in *Proc. 4th Int. Conf. Inf. Syst. Appl.*, 2018, pp. 1–6.

XOLANI DASTILE received the bachelor's degree (Hons.) in mathematics and the bachelor's (Hons.) and M.Sc. degrees in mathematical statistics from Rhodes University, South Africa. He is currently pursuing the Ph.D. degree in computer science in the field of deep learning with the University of the Witwatersrand, Johannesburg, South Africa. He is also a Senior Data Scientist with Fuel Management Company in South Africa. He previously worked in major banks in South Africa under credit risk departments as a Quantitative Analyst as well as a Data Scientist.

TURGAY CELIK (Member, IEEE) received the Ph.D. degree from The University of Warwick, Coventry, U.K., in 2011. He is currently a Professor of Digital Transformation and the Director of the Wits Institute of Data Science, University of the Witwatersrand, Johannesburg, South Africa. His research interests include signal and image processing, machine intelligence, robotics, data science, and remote sensing. He is also an Associate Editor of the *IET Electronics Letters* (ELL), the IEEE Geoscience and Remote Sensing Letters (GRSL), and *Signal, Image and Video Processing* (SIVP, Springer).