Deep neural networks for probability of default modelling

Article *in* Journal of Industrial and Management Optimization · March 2024

DOI: 10.3934/jimo.2024038

CITATIONS
5

READS
447

2 authors, including:

Kyriakos Georgiou
University of Naples Federico II
8 PUBLICATIONS **22** CITATIONS

SEE PROFILE

Journal of Industrial and Management Optimization

doi:10.3934/jimo.2024038

DEEP NEURAL NETWORKS FOR PROBABILITY OF DEFAULT MODELLING

Kyriakos Georgiou* and Athanasios N. Yannacopoulos

Department of Statistics and Stochastic Modeling and Applications Laboratory, Athens University of Economics and Business, Greece

Abstract. In this paper we develop Deep Neural Networks for the approximation of the solution to Partial Integro-Differential Equations (PIDE) that arise in the calculation of Probability of Default functions. We consider a modelling framework in compliance with the spirit and regulations of the International Financial Reporting Standard 9 and use the resulting Deep Learning models to estimate default probabilities that can be used to solve credit risk problems. Detailed comparisons with standard numerical analysis schemes for the solutions to these PIDEs are also reported, enhancing the understanding and adding to the discussion regarding the applicability of the related Machine Learning methodologies.

1. **Introduction.** The International Financial Reporting Standards (IFRS) 9 have brought significant changes and modelling requirements to the field of credit risk. Perhaps one of the most influential changes due to the IFRS 9 is the requirement for financial institutions to consider Expected Lifetime Provisions (ECL), whereby future losses must be forecast using mathematically robust and rigorous methods, for all credit exposures which are considered to have displayed a significant increase in risk. This estimation requires knowledge of the lifetime Probability of Default (PD) for all loan exposures, as well as additional risk parameters such as the Loss Given Default (LGD) and the Exposure at Default (EAD), and finally being able to update these quantities dynamically under changing market conditions. These effects have started appearing in the literature, as well, such as recent papers studying the the ECL calculation ([9, 61]), as well as other modelling tasks associated to IFRS 9 (e.g., [25]).

In general, calculating default probabilities both analytically and numerically is of paramount importance in risk management and a broad range of financial applications. However, particularly under IFRS 9, credit loss forecasting has introduced the need for robust structural models, that can be used for pricing and provisioning purposes. To this end, we will consider stochastic models for the evolution the underlying asset value process, whose default will be studied as an appropriate firs-time-hitting problem. Therefore, it can be assumed that we are working mainly within portfolios of corporate and small business loans, where it is common

2020 *Mathematics Subject Classification.* 60H30, 68T07, 45K05, 91G40, 91G60, 91-08.

Key words and phrases. Stochastic modeling, probability, deep neural networks, default, credit risk, numerical methods, ifrs 9.

*Corresponding author: Kyriakos Georgiou.

practice to consider the company's assets to be governed by stochastic process (see e.g., [6] and [5]). Under this assumption, the PD associated with each loan depends on the underlying asset process and we can define the PD as the probability that the asset process falls below a fixed threshold. More specifically, we will assume that the asset process is governed by an Ornstein - Uhlenbeck (OU) process with a jump component, a member of the family of jump-diffusion processes. We note that practitioners may consider the evolution of asset-dependent processes instead, e.g., returns; such processes can still be described by similar stochastic models, rendering the methods proposed in this paper applicable in these cases, as well. For brevity, hereinafter we will refer to this underlying process as the asset process, with the understanding that is can be replaced with any related dynamics considered appropriate by practitioners. Important theoretical background of such processes and their properties are given in [2] and [47]. The jump process will account for abrupt changes in the asset processes, which are very common in practice and are closely related to loan defaults.

Under this stochastic modelling framework, the PD can be considered as a function of the asset process, as well as additional latent variables. Given the existence of such additional variables, estimating the evolution of the PD values in a consistent and efficient way is of paramount important in order to tackle various modelling tasks which are currently open problems for financial institutions under the IFRS 9 framework. We describe the mathematical framework, which entails deriving a Partial Integro-differential Equations (PIDEs) that describe PD function.

In order to estimate the PD values we propose the use of Deep Neural Network (DNN) models, which can be used to solve the PIDEs. As we will see, in addition to providing an alternative to the standard numerical methods, such as Finite Difference (FD) schemes, DNNs have the important advantage that they do not suffer from the well-known "curse of dimensionality". This allows us to consider asset models of higher dimensions, i.e., with multiple latent variables. The aim of this paper is to study the application of DNNs to credit risk modelling in the setting described above. Our contributions are the following:

1. We describe and implement the development of DNN models for the estimation of PDs.
2. We showcase the DNN models' ability to "learn" high-dimensional functions, by considering two problems which are intractable using standard finite difference schemes:
   - PD calculation for an asset process incorporating two latent variables, a regime-switching and stochastic volatility component.
   - Developing an approach to estimate PD values given a family of stochastic processes the asset value may follow. This is particularly applicable under IFRS 9, since it is reasonable to consider that the underlying parameters of the asset value process themselves change as time evolves. Such changes might be caused by macroeconomic or industry factors, for example. We will see how a DNN model can be trained to estimate the evolution of the PD under this setting.
3. We perform a detailed comparison between the DNN approach and existing numerical methods for solving such equations. We analyse the applicability of each approach when being used in real-life modelling and discuss model architecture, training parameters/algorithms and computational requirements

for the practical implementation of Deep learning models versus the Finite Difference counterparts, thereby providing a holistic analysis for practical and managerial considerations.
4. Furthermore, we describe and discuss findings from the DNN models that warrant further research. These are important considerations since practitioners must take into account a range of factors when deciding which approaches to implement, from latent variables affecting the portfolio to computational power and recalibration periods.

Finally, we note that, even though motivated by credit risk, the approaches detailed in this paper can find applications in other areas of financial mathematics, such as derivatives pricing, where the use of stochastic modelling remains prevalent, e.g., in the pricing of barrier options.

2. **Literature review.** The field of Deep Learning has been developed and applied to many areas in recent years, ranging from financial modelling to production problems (e.g., [40]) and epidemiology ([41]). The literature related to the work herein pertains to the application of Machine Learning (ML) techniques to solve Ordinary and Partial Differential Equations (ODE & PDE), giving rise to a new field of Deep Learning for ODEs and PDEs. Currently, a widely used method is the Physics Informed Neural Network (PINN), developed in [52]. PINNs have been studied and applied in various contexts. See seminal papers such as [14, 49] and [20] and references therein.

At the same time Machine Learning and Deep Learning techniques have recently been applied to the field of credit risk. A very important such application is shown in [24], where a method to learn the solution of PDEs used for credit and insurance risk applications is used. This method relies on the celebrated Feynman-Kac formula to construct the loss function. Additional Deep Learning have been studied in the context of credit risk, e.g.,:

1. in [1] and [36] neural network models are applied to a standard loan dataset with multiple features and a default label. In the former, the authors compare DNN models with standard approaches (e.g., tree-based models) and show that the DNNs are in fact less stable, opening up important questions that we also discuss based on our results in this paper.
2. in [56], the authors consider a deep ensemble model which shows promising results in being able to overcome the class imbalance issue (which is often prevalent in credit risk), and performs well in the prediction of default.
3. in [17] Deep Learning is used to process and analyse long historical sequences of transactional data from credit cards and achieved a significant increase in performance in benchmark problems.

The work outlined above showcases the industry's interest in Deep Learning methods for credit risk; however, most of this work uses existing datasets to test either the performance of Neural Networks versus standard models or to propose improvements in the model development process. Therefore, there exists an important research gap pertaining to the combination of the rigorous stochastic modelling framework which has been applied to credit risk and DNN modelling. This is an important problem both in the context of Machine Learning applications, as well as in the field of numerical methods. We add to the existing literature by focusing on

this combination, and study both the advantages and disadvantages of DNN models in this context.

3. Model for the asset value process.

3.1. **One dimensional model.** As mentioned, it is standard in the field of financial mathematics to consider the evolution of a debtor's assets to be governed by a stochastic process. A well-documented process that is used in various such applications is the Ornstein-Uhlenbeck (OU) process. OU models have been considered in past research and many applications. For example, a well-known special case is the Vasicek model [60]. By including a jump process to the continuous OU asset process, we obtain the Lévy-driven Ornstein-Uhlenbeck process, defined below:

$$dG_u = k(\theta - G_u)du + \sigma dB_u + \int_{\mathbb{R}} zN(du, dz), \quad G_0 = x. \tag{1}$$

The process has two sources of randomness: the continuous Brownian motion B_t and the discontinuous Lévy jump term L_t, determined by the Poisson random measure $N(\cdot, \cdot)$, defined by $N(t, U) = \sum_{0<s\leq t} \chi(\Delta L_s)$ for every Borel set $U \subset \mathbb{R}$, where $\Delta L_s = L_s - L_{s_-}$, that represents the number of jumps of size $\Delta L_s \in U$, which occur up to time t. It can be shown that the jump term is a Compound Poisson process with arrival rate $\lambda = \nu(\mathbb{R})$ and jump distribution $f(dz) = \lambda^{-1}\nu(dz)$. Furthermore, this process is temporally homogeneous, as the the sum of two homogeneous processes (the continuous OU and Compound Poisson processes). For an in depth analysis of integrals with respect to Poisson measures and their properties see e.g., [38].

This is a natural generalization, as significant changes in credit events are often abrupt and unpredictable (particularly a deterioration in creditworthiness), corresponding to a discontinuous component in the driving stochastic process. Indeed, the goal of credit risk requirements under IFRS 9 is to ensure that financial institutions and their customers are protected against such rare and unexpected events and the subsequent losses. It is therefore important to capture the effect of such events mathematically, which is why this model will form the basis of our analysis. To conclude, we note that the use of Lévy processes for financial modeling is well-documented and established. [55] gives an extensive analysis of Lévy processes and their use for asset process modelling, credit derivatives pricing and more. In [43] and [48] the authors consider a Lévy-driven OU process, and Lévy multivariate models for assets processes.Seminal work has also been done in the study of Lévy-driven OU processes in [6].

As discussed, our aim is to develop a stochastic model that incorporates the exogenous variables required when considering asset value processes. To incorporate such effects, we build upon the family of regime switching and stochastic volatility models, as described below.

3.2. **Regime switching and stochastic volatility models.** By definition, changes in the risk profile of an exposure will correspond to changes in the dynamics of the underlying asset process. For example, a debtor may request restructuring, or may be 30 days delinquent. This will trigger a significant increase in credit risk event, which can then affect the underlying asset value process. To capture this dependency we consider a regime switching model for the asset process, whereby the parameters of the stochastic process vary according to the underlying rating (under IFRS 9 these are now referred to as Stages) of the exposure. We can do this

by considering the Continuous Time Markov Chain (CTMC) $(R_t)_{t\geq 0}$ describing the rating at time t, where the set of all loan ratings is denoted by $\mathcal{R}$, with cardinality $|\mathcal{R}| = R$. Therefore, we obtain the following jump diffusion with Markov switching model:

$$dG_u = k(R_u)\big(\theta(R_u) - G_u\big)dt + \sigma(R_u)dB_u + \int_{\mathbb{R}} zN(du, dz), \quad G_0 = x, R_0 = \rho, \quad (2)$$

with $\rho \in \mathcal{R}$. Note that in subsequent sections we adopt the notation $k_\rho, \theta_\rho, \sigma_\rho$, for brevity.

In the sequel, to develop a realistic model we want to capture the effects of macroeconomic variables, which naturally affect the evolution of the asset process. Typically, such latent variables are incorporated by considering stochastic volatility models, whereby the diffusion term of the asset process also evolves according to a stochastic process, as described by the coupled process:

$$\begin{cases} dG_t = \mu_x(G_t, Y_t)dt + \sigma_x(G_t, Y_t)dB_t + \int_{\mathbb{R}} zN(dt, dz), \quad G_s = x, \\ dY_t = \mu_y(Y_t)dt + \sigma_y(Y_t)dW_t, \quad Y_s = y, \end{cases} \quad (3)$$

for $y \in \mathcal{V}$ and where B_t and W_t are independent Brownian motions. Standard cases are the Bates' model, introduced in [7], as well as the Heston model (see [10]), a version of which we consider below. In particular, letting $\mu_x(Y_t) = k(\theta - Y_t)$ and $\sigma_x(Y_t) = \sqrt{Y_t}$, we obtain the asset process driven by a stochastic volatility process, which follows the well-established Cox–Ingersoll–Ross (CIR) model, developed in [19] (note that both processes are time-homogeneous):

$$\begin{cases} dG_u = k(\theta - G_u)dt + \sqrt{Y_t}dB_t + \int_{\mathbb{R}} zN(dt, dz), \quad G_0 = x, \\ dY_u = \kappa(\mu - Y_u)dt + \xi\sqrt{Y_u}dW_t, \quad Y_0 = y. \end{cases} \quad (4)$$

The above models are widely used in mathematical finance and stochastic modelling. Regime switching is a well-documented approach in financial modelling (see e.g., [53, 54] and [32]), with applications ranging from macroeconomics (e.g., [3]) to option pricing (e.g., [21], [31]) and interest rate modelling ([28]). Finally, we refer the reader to [63] for a detailed analysis of more general regime switching jump diffusion processes, where the authors also consider the dynamics of the underlying Markov process to be a function of the initial position of the jump diffusion.

The stochastic volatility model is a natural extension, as it can be seen as the limit process of the regime switching model, as $\mathcal{R} = \mathbb{R}_+$. Such models, in the case of both continuous and jump processes, have also been considered for numerous applications in mathematical finance, particularly in pricing and hedging, such as in [59] and [27].

4. **Mathematical framework for probability of default.**

4.1. **The generelized asset value model.**

We now combine the regime switching and stochastic volatility models to produce a generalized model, which we will use to construct a framework that encapsulates a large family of stochastic processes that can be used for asset value modelling and subsequent credit risk calculations. This framework addresses the strict requirements under IFRS 9, whereby credit risk modelling is required to incorporate multiple appropriate latent variables, whilst adhering to mathematical rigor. A combination of (2) and (4) gives rise to the following.

Definition 4.1. Under the generalized model, the asset value process is defined by the triple $(G_t, R_t, Y_t)_{t\geq 0}$, capturing both the switching and volatility processes, and is given by:

$$\begin{cases} dG_u = k(R_u)\big(\theta(R_u) - G_u\big)dt + \sigma(R_u)\sqrt{Y_t}dB_u + \int_{\mathbb{R}} zN(du, dz), \quad G_0 = x, \\ dY_u = \kappa(\mu - Y_u)dt + \xi\sqrt{Y_u}dW_t, \end{cases} \tag{5}$$

with $G_0 = x, R_0 = \rho$ and $Y_0 = y$.

Before moving on to define the appropriate Probability of Default functions, it will be useful to define some notation.

Notation 4.2. Throughout the remainder of this paper, we employ the notation Z_u^x to represent the stochastic process $(Z_u)_{u\geq 0}$, with $Z_0 = x$, where appropriate. We also generalize this notation to incorporate cases with additional underlying variables $X_t^1, X_t^2, \ldots, X_t^n$, by writing $Z_u^{(x_1,x_2,\ldots,x_n)}$ to represent $(Z_u)_{u\geq 0}$ with $X_0^i = x_i$ for $i = 1, 2, \ldots, n$, (the superscripts are to be understood as indices, i.e., the $i-$th underlying variable is $(X_t^i)_{t\geq 0}$).

4.2. **The probability of default function and PIDEs.** We start by discussing the PD process which, in its most general form, can be written as a function of both the starting time and maturity, as well as of the initial position of the corresponding asset value process. Furthermore, to accurately model real-life dynamics, it is necessary to account for the dependence on latent variables which affect the PD. Incorporating such processes, which in practice are e.g., macroeconomic variables or different market regimes, is of paramount importance as it largely affects PD estimation and subsequent modelling results. To begin, consider compact and bounded sets $\mathcal{D}, \mathcal{D}_i \subset \mathbb{R}$, for $i = 1, \ldots, d$. Then, we have the following definition for the PD function:

Definition 4.3. Consider $x \in \mathcal{D}$ and the vector of (discrete or continuous) stochastic processes $(X_t^i)_{t\geq 0}$ with corresponding state spaces $\mathcal{D}_i$, for $i = 1, \ldots, d$. Furthermore, consider the stochastic asset value process $(G_t)_{t\geq s}$, with initial value $G_s = x$ and which depends on $(X_t^i)_{t\geq 0}$, for $i = 1, \ldots, d$. Then, we define the Probability of Default function $\Psi : \mathcal{D} \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_d \times [0, T] \times [0, T] \to [0, 1]$, for some fixed $T > 0$, by:

$$\Psi(x, x_s^1, x_s^2, \ldots, x_s^d, s, t) = \mathbb{P}\Big(\inf_{s\leq r\leq t} G_r \leq 0 | G_s = x, X_s^1 = x_s^1, X_s^2 = x_s^2, \ldots, X_s^d = x_s^d\Big), \tag{6}$$

and the corresponding survival probability $\Phi : \mathcal{D} \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_d \times [0, T] \times [0, T] \to [0, 1]$ by:

$$\Phi(x, x_s^1, x_s^2, \ldots, x_s^d, s, t) = 1 - \Psi(x, x_s^1, x_s^2, \ldots, x_s^d, s, t). \tag{7}$$

To motivate this definition and its usefulness, notice that by fixing s we obtain the standard finite-horizon ruin probability (see e.g., [45]), whereas by fixing t we obtain the ruin probability with variable starting time, as defined in [46], which can be used to define a martingale. Finally, allowing $t \to \infty$ we obtain the infinite-horizon ruin probability. In general, PD functions of the above form find many applications and have been considered in the field of credit risk, such as in [4], [55] and [62].

Following the definition of the PD function, as given in (6), under the generalized model (5) we will condition on the initial state of the regime switching and stochastic

volatility processes, i.e., ρ and y, to obtain:

$$\Psi(x,\rho,y,s,t) := \mathbb{P}\Big(\inf_{s\le r\le t} G_r \le 0 | G_s = x, R_s = \rho, Y_s = y\Big). \tag{8}$$

Under this assumption, we can utilize the time homogeneity property to write the survival and PD functions more succinctly, whilst still being able to obtain the evolution of the PD, both in the case of a variable maturity and variable starting time. We use this to obtain and analyze appropriate equations for the PD function in the result below.

Proposition 4.4. *Under the generalized model (5), the survival function with variable maturity $\tilde{\Phi}(x,\rho,y,s;t)$ and variable starting time $\tilde{\Phi}(x,\rho,y,s;t)$, can be retrieved from the generalized function $\Psi(x,\rho,y,u)$, where $u = t-s$ represents the remaining time until maturity. Furthermore, the survival probability $\Phi(x,\rho,y,u)$ is a continuous function of (x,y,u) and satisfies (either in the classical or the viscosity sense, depending on smoothness) the PIDE:*

$$\begin{aligned}
&\frac{\partial\Phi}{\partial u}(x,\rho,y,u)\\
&= k_\rho(\theta_\rho - x)\frac{\partial\Phi}{\partial x}(x,\rho,y,u) + \kappa(\mu-y)\frac{\partial\Phi}{\partial y}(x,\rho,y,u) + \frac{1}{2}\sigma_\rho^2 y\frac{\partial^2\Phi}{\partial x^2}(x,\rho,y,u)\\
&\quad + \frac{1}{2}\xi^2 y\frac{\partial^2\Phi}{\partial y^2}(x,\rho,y,u) + \sum_{j\neq\rho} q_{\rho j}\Big(\Phi(x,j,y,u) - \Phi(x,\rho,y,u)\Big)\\
&\quad + \int_{\mathbb{R}}\Big(\Phi(x+z,\rho,y,u) - \Phi(x,\rho,y,u)\Big)\nu(dz),
\end{aligned} \tag{9}$$

for $(x,\rho,y,u)\in\mathcal{D}\times\mathcal{R}\times\mathcal{V}\times[0,T]$, with initial and boundary conditions:

$$\begin{aligned}
\Phi(x,\rho,y,0) &= \mathbb{1}_{\{x>0\}}, \quad (x,\rho,y)\in\mathcal{D}\times\mathcal{R}\times\mathcal{V},\\
\Phi(0,\rho,y,u) &= 0, \quad (\rho,y,u)\in\mathcal{R}\times\mathcal{V}\times[0,T],\\
\Phi(x,\rho,y,u) &\to 1, \ \text{as } x\to\infty, \quad (\rho,y,u)\in\mathcal{R}\times\mathcal{V}\times[0,T],\\
\frac{\partial\Phi}{\partial y}(x,y,u) &= 0, \ \text{as } y\to\infty \quad (x,\rho,u)\in\mathcal{D}\times\mathcal{R}\times[0,T].
\end{aligned} \tag{10}$$

Proof. The first part of the result follows easily from the time-homogeneity of the asset process, since:

$$\begin{aligned}
\Phi(x,\rho,y,s,t) &= \mathbb{P}\Big(\inf_{s\le r\le t} G_r > 0 | G_s = x, R_s = \rho, Y_s = s\Big)\\
&= \mathbb{P}\Big(\inf_{0\le r\le t-s} G_r > 0 | G_0 = x, R_0 = \rho, Y_0 = y\Big) = \Phi(x,\rho,y,0,t-s),
\end{aligned} \tag{11}$$

where we write the survival as an explicit function of both the starting and maturity times. Hence, we can now write $\Phi(x,\rho,y,u)$ with $u := t-s$ representing the remaining time until maturity.

From the theory of stochastic processes we have that the survival function presented above satisfy the Kolmogorov backward equation. Hence, using the fact that the right hand side of (9) is the infinitesimal generator of the generalized asset process we obtain the PIDE. The proof regarding continuity involves various technical details which are outside the scope of this paper. For an in depth proof and analysis we refer the reader to [26]. □

It is important to note that the continuity properties of the solution to the PIDE are essential, from the theoretical point of view, for the Neural Network model to work, as standard Universal Approximation Theorems are usually obtained for classes of continuous functions.

Remark 4.5. It is worth noting that the homogeneity assumption is strong, yet fair. Particularly in the case of corporate and/or small business loans, it is natural to consider such asset processes, since credit risk modelling is often done across complete financial/business cycles (e.g., years or quarters), over which the evolution of the asset process (or related return processes) will have similar dynamics, regardless of the exact point in time. However, even without this assumption, the approaches developed in this paper can be used by fixing the either the starting or maturity time in order to obtain whichever case of the PD process the modeller requires. Hence, this framework is useful for PD modelling under any asset value process.

Throughout the remainder of this paper, we will therefore use the following formulation of the PD and corresponding survival process:

$$\Psi(x,\rho,y,u) := \mathbb{P}\Big(\inf_{r\leq u} G_r \leq 0 | G_0 = x, R_0 = \rho, Y_0 = y\Big) \equiv \mathbb{P}\Big(\inf_{r\leq u} G_r^{(x,\rho,y)} \leq 0\Big), \quad (12)$$

$$\Phi(x,\rho,y,u) := 1 - \Psi(x,\rho,y,u) = \mathbb{P}\Big(\inf_{r\leq u} G_r^{(x,\rho,y)} > 0\Big). \quad (13)$$

It is important to not that when considering real-life credit risk modelling tasks the state of the regime (e.g., the IFRS 9 Stage), and/or the value of any underlying macroeconomic factors may be observable and can therefore be inserted explicitly into the generalized model (5), thereby obtaining the regime switching or stochastic volatility model, with PD functions given by:

$$\Psi(x,\rho,u) = \mathbb{P}\Big(\inf_{r\leq u} G_r^{(x,\rho)} > 0\Big), \quad (14)$$

$$\Psi(x,y,u) = \mathbb{P}\Big(\inf_{r\leq u} G_r^{(x,y)} > 0\Big), \quad (15)$$

respectively, and corresponding PD (or survival) functions $\Psi(x,\rho,u)$ and $\Psi(x,y,u)$ (or $\Phi(x,\rho,u)$ and $\Phi(x,y,u)$). These models can then be used for the tasks we consider under IFRS 9, and credit risk more generally. It is straightforward to obtain the corresponding PIDEs these survival functions satisfy as simplified versions of (9). For completeness, these are included in Appendix A. For more details on the generators of regime switching and stochastic volatility models see e.g., [30, 63].

5. Deep Neural Networks.

Neural Network models, commonly referred to as simply Neural Networks, have risen to prominence in the Machine Learning and Artificial Intelligence fields in recent years, having been used in a vast amount of fields, spanning both theoretical and practical problems. Accounts of their vast applications can be found in e.g., [23, 50] and [37]. In turn, the study of such models has lead to many different architectures such as Convolutional (CNN), which have been widely used in image recognition and classification (see e.g., [29] and [35]), or Recurrent (RNN) ([44]), which can handle multiple other forms of data, such as text ([58]). In general, the structure of any NN model relies on the Multi-Layer Perceptron (MLP) architecture, described in the following definition.

Definition 5.1. Consider a multi-dimensional input vector $\mathbf{x} \in \mathbb{R}^n$, with dependent variable $y \in \mathbb{R}$. Then, a MLP feed-forward Neural Network, commonly referred to as a Deep Neural Network (DNN), with L hidden layers and the activation function $\mathbf{g}^{(i)} : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$, for each layer i consists of iterations of the following equations:

$$\begin{aligned} \mathbf{x}^{(0)} &= \mathbf{x} \\ \mathbf{x}^{(i)} := \mathbf{a}^{(i)} &= \mathbf{g}^{(i)} \left(W^{(i-1)} \mathbf{x}^{(i-1)} + b^{(i-1)} \right), \text{for all } 1 \leq i \leq L \\ y &= W^{(L)} \mathbf{x}^{(L)} + b^{(L)}, \end{aligned} \tag{16}$$

where the i-th hidden layer $\mathbf{x}^{(i)}$ is a vector of length n_i for $1 \leq i \leq L$, n_i is the size of the i-th hidden layer and the activation function $g^{(i)}$ for each layer i is applied to each coordinate of its input, i.e.,

$$\mathbf{g}^{(i)}(\mathbf{x}^{(i)}) = \left(g\left(x_1^{(i-1)}\right), g\left(x_2^{(i-1)}\right), \ldots, g\left(x_{n_{i-1}}^{(i-1)}\right) \right).$$

An illustration of the DNN architecture is given in Fig. 1. In the DNN models developed in this paper the input vector consists of the spacial, temporal and (if applicable) latent variable values at which we estimate the solution to the PIDE. Of great importance is the selection of appropriate activation functions. It can be shown that non-linear activation functions are required in order to train DNNs to solve non-trivial problems, and, depending on the setting and framework, there exist many such functions that can be used. Some of the most commonly used sigmoid, hyperbolic tangent, Rectified Linear Unit (ReLU) and Gaussian. In depth studies on the selection of activation function can be found in e.g., [51] and [33]. Finally, we remark that the other hyperparameters, such as the number of hidden layers and the nodes per layer have also been shown to affect model performance. Empirical and statistical studies can be found in e.g., [39] and [22]. Despite their obvious importance, selecting activation functions and model hyperparameters still largely remains an "art", relying heavily on the experience of the modeler, rather than a science in itself, given that robust techniques have not yet been established. We will discuss this further in subsequent sections in the context of credit risk modeling.

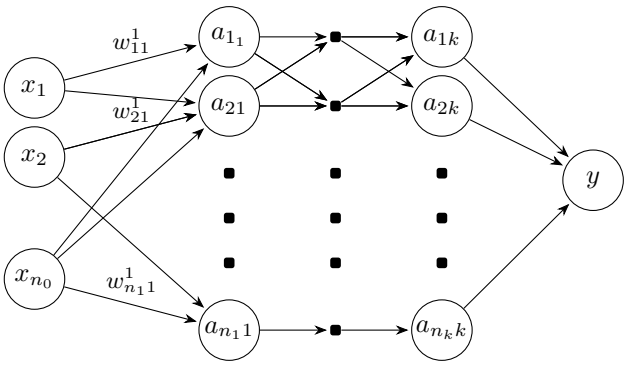

FIGURE 1. Fully connected Neural Network with an n-dimensional input layer, k dense layers.

The architecture and mathematical foundation of Neural Networks as machine learning models has been established since the first half of the 20th century. However, the lack of an efficient training algorithm, i.e., mathematical framework for

the estimation of the trainable parameters $W^{(i)}$ and $b^{(i)}$, for these complex models meant that they could not be used for the wide range of problems for which they are theoretically applicable. The backpropagation algorithm ([34]) was developed and able to efficiently solve the problem of training such a model. Simply put, backpropagation is used to train the model by calculating the errors in its predictions and updating the parameters in order to minimize the error. This result, coupled with the exponential increase in computational power in recent years have lead to significant breakthroughs in DNNs, their applications and research.

Backpropagation itself utilizes a gradient descent algorithm, which depends on the learning rate parameter, which affects the amount by which the parameters are altered during the training process. For most applications, decaying, step-wise learning rates are used, which is the approach we adopt in the following sections.

5.1. **DNN models for solving PDEs.** In this section we present the method that will be used to train DNNs to calculate solutions of PDEs. In [12] a seminal overview of this and other approaches is given and applied to various PDEs. We will follow the notation in this paper in what follows. As already mentioned, PINNs are perhaps the most widely used such method. However, for the given equations, the development of PINNs may prove costly, due to the need to explicitly encode and calculate each term in the equation, including the integral terms. A different approach to training Neural Networks to solve PDEs has been studied in [8] and further in the case of PIDEs in [24]. As mentioned in the overview provided by [12], this method is applicable to the well-known class of PDEs that arise as solutions to the Kolmogorov backward or forward equations of Itô processes satisfying the SDE:

$$dG_t = \mu(G_t, t)dt + \sigma(G_t, t)dW_t, \quad G_0 = x. \tag{17}$$

The corresponding backward equation is then given by:

$$\frac{\partial u}{\partial t} + \mu(x,t)\frac{\partial u}{\partial x} + \frac{1}{2}\sigma^2(x,t)\frac{\partial^2 u}{\partial x^2} = 0, \quad t \in [0,T], x \in \mathcal{D}, \tag{18}$$

$$u(x,T) = g(x,T). \tag{19}$$

To obtain the forward problem we simply use the change of variables $u := T - t$ to define the time until maturity. Then, we can apply the Feynman-Kac formula in order to establish the relationship between the solution of the PDE and the underlying SDE, using the terminal condition. The Feynman-Kac representation can then be used to construct an appropriate loss function. To this end, consider the random variable $Y = g(G_T^x, T)$. Following [24], the solution of (18), $u(x,t)$ can be written as:

$$u(x,t) = \mathbb{E}\big[g(G_T,T)|\mathcal{F}_t\big] = \mathbb{E}\big[g(G_T,T)|X_t = x\big], \tag{20}$$

where the equality is a result of the Markov property, and therefore we can write:

$$u(x,t) = \mathbb{E}[Y|\mathcal{F}_t]. \tag{21}$$

It then follows that, for a fixed time t, $u(x,t)$ is the solution of the minimization problem:

$$\min_u \mathbb{E}[|Y - u(x)|^2]. \tag{22}$$

Hence, to train a DNN model we can use the estimator of the expectation above as the loss function:

$$L_\theta(x) = \frac{1}{M}\sum_{i=1}^{M}\big(Y_i - u_\theta(x,t)\big)^2, \tag{23}$$

where $u_\theta(x,t)$ is the Neural Network approximation of the solution and Y_i is the random variable representing the payoff function, as described above, corresponding to the $i-th$ simulated asset process path. Training therefore consists of simulating M paths of the asset value process and calculating the payoff in order to construct the loss function and estimate the trainable model parameters. Notice how this construction only takes into account different values of the initial position x we therefore estimate the solution to the PDE only as a function of x, i.e., $u(x;t)$. We will extend this in order to produce the PDs as functions of time, as well, in the following sections.

When studying Physics Informed Neural Networks and the Feynman-Kac approach as described above, one can easily observe important differences between the two approaches presented for training DNN models for the solutions of PDEs. Notably, in what we will hereinafter refer to as the the Feynman-Kac approach, we rely on stochastic simulation of the underlying process to generate training data. This approach is also quite practical, as it only requires simulating paths of the underlying stochastic processes. Therefore, a DNN can be easily developed for all the asset value models we have examined (with appropriate changes in the neural network architecture).

6. **Problem statement and approach.** In this Section we describe the mathematical framework of the modelling task and the method of developing the simulation-based Neural Network models.

The DNN models will be used to estimate PD values under various stochastic asset models, with the method described above. For clarity in the implementation of the method, we will considering the backward formulation of the PIDEs obtained, with appropriate terminal condition at the time of maturity $t = T$, via the payoff function h. Hence, the resulting PDs being functions of the starting time t (considering fixed maturity T) rather than the remaining time until maturity u. For clarity, we recall this version of the definition of the PD process below.

Definition 6.1. Consider the Lévy-driven OU stochastic asset process $(G_t)_{t\geq 0}$, depending on variables $X^1, X^2, \ldots, X^N$ (which could be discrete or continuous). The corresponding survival probability function is then:

$$\Phi(x, x_0^1, \ldots, x_0^N, t) = \mathbb{P}\Big(\inf_{t\leq r\leq T} G_r > 0 | G_t = x, X_t^1 = x^1, \ldots, X_t^N = x^N\Big), \tag{24}$$

with corresponding Probability Default process:

$$\Psi(x, x_0^1, \ldots, x_0^N, t) = 1 - \Phi(x, x_0^1, \ldots, x_0^N, t). \tag{25}$$

We will be tackling the problem of training and testing DNN models to estimate the PD functions under all the asset process models previously presented, as well as showcasing additional capabilities that these models have when considering PD estimation.

6.1. **Feynman-Kac DNNs for PD functions.** Training the Neural Network now consists of simulating paths of the asset process $G_t, t \in [0, T]$, for various values of the initial position $x \in \mathcal{D}$ in order to calculate the payoff function, which is given by $Y := g(T, G_T^x) = \mathbb{1}_{\{\inf_{t \leq r \leq T} G_r^x \leq 0\}}$. Hence, we obtain the set of random variables Y, from many initial positions $x \in \mathcal{D}$, which will be used to train the Neural Network using the loss function given in (23).

Notice that, in the training process described above, fixed starting and maturity times t and T are considered and, hence, the resulting model will be able to estimate the PD only as a function of the initial position; this is similar to the process that one must apply when performing a Monte Carlo estimation, and is therefore sub-optimal, as we want to estimate PD values with variable time until maturity. We can therefore improve this method by simulating paths for various values of the time until maturity $u = T - t$ (by changing either the starting or maturity time) when creating the training dataset and considering the DNN with a two-dimensional input layer (initial position and maturity time). In what follows, we start by considering a fixed maturity time to train a model with only one input, which we will use to gauge the best choice of model hyperparameters, and will subsequently extend the Neural Networks to account for the additional variables. We summarize the DNN development process in Algorithm 1. For clarity, Fig. 2 also provides a general illustration of how such models can be constructed and trained.

Algorithm 1 DNN model development algorithm

1. Generation of training set
   - Simulate M paths of the asset value process, starting with an initial values of the input variables.
   - Calculate the payoff variables by "killing" the process if it falls below the default threshold (in our examples, zero), which will comprise the training set.
2. Training the Neural Network
   - Use the payoff variables to construct the loss function (23).
   - Calculate trainable parameters for the DNN using appropriate optimization methods
3. Approximate the solution of the PIDE using the trained DNN model at suitable inputs for the independent variables of PD (or survival) function.

We will now implement Algorithm 1 for the various PIDEs for the calculation of the PD function introduced in Section 4. For completeness, we gather all related variables and quantities in Table 1.

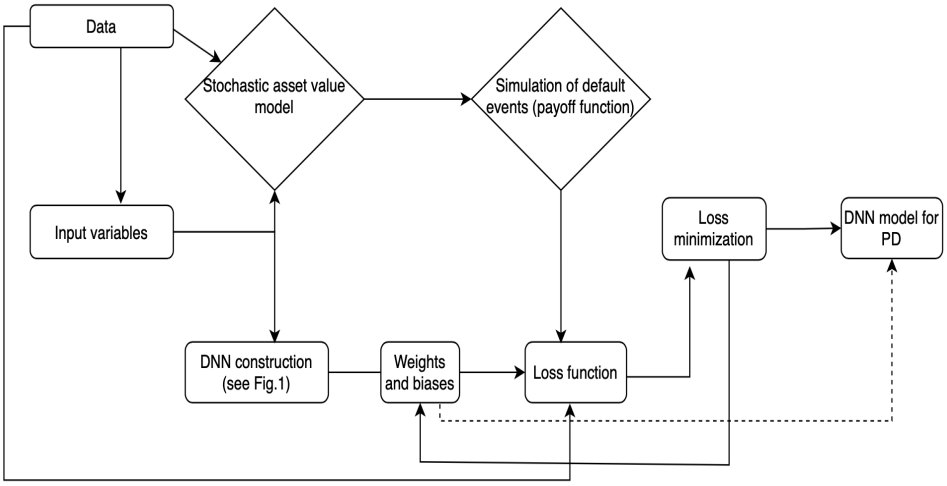

FIGURE 2. Graphical illustration of the proposed model.

Quantity (type)	Notation
Probability of Default (function)	Ψ
Probability of Survival (function)	Φ
Initial position (variable)	x
Initial volatility (variable)	y
Initial regime (variable)	ρ
Asset value (random variable)	G
Input of model (5) (parameters)	$k, \theta, \sigma, \kappa, \mu, \xi$
Starting time (variable)	t
Maturity time (variable)	T

TABLE 1. Quantities of interest.

7. **Model results.**

7.1. **One dimensional input layer (initial position).** We begin by considering a fixed $t = 0$ and $T = 1.0$, and training a DNN model as a function of only the initial position. This requires a Neural Network model with a one-dimensional input and output layer (the initial position and probability of default, respectively), as shown in Fig. 3.

We do this for the one dimensional, as well as for the regime switching and stochastic volatility models (considering a fixed value of the initial regime and volatility value, respectively). Finding the best choice of hidden layers for this model, will allow us to make an educated estimate for the optimal selection of hidden layers when training the extended DNN that estimates the PD as a function of both initial position and maturity. Recall that in its simplest, one dimensional, form we consider a jump-diffusion asset value process:

$$dG_t = k(\theta - x)dG_t + \sigma dW_t + \int_{\mathbb{R}} zN(dt, dz), G_0 = x,$$

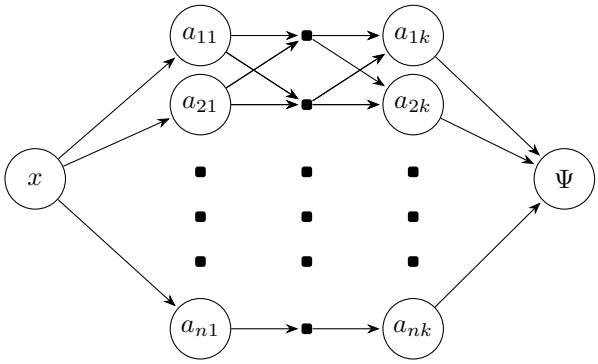

FIGURE 3. Fully connected Neural Network with a one-dimensional input layer and a one-dimensional output layer.

with the corresponding regime switching and stochastic volatility models given by (2) and (4), respectively. For illustrative purposes we consider specific values of the asset process parameters. This analysis is gathered in the example below.

Example 7.1. Consider the one dimensional asset value process (16), with $(k, \theta, \sigma) = (0.5, 3.5, 2.0)$, a Compound Poisson Process with normally distributed size $Z \sim N(0.0, 2.0)$ and jump rate $\lambda = 1.0$. Furthermore, consider an asset process governed by the regime switching model with three regimes, $\mathcal{R} = \{1, 2, 3\}$, representing the IFRS 9 Stages (performing, significant increase in credit risk and defaulted) and corresponding parameters $(k_1, k_2, k_3) = (0.3, 0.2, 0.0), (\theta_1, \theta_2, \theta_3) = (0.8, 0.5, 0.0), (\sigma_1, \sigma_2, \sigma_3) = (0.3, 0.5, 0.0)$, We further assume normally distributed jumps with $Z \sim N(0.0, 0.5)$ and rate $\lambda = 1.0$ and a generator matrix of the underlying Continuous Time Markov Chain given by:

$$Q = \begin{pmatrix} -0.5 & 0.3 & 0.2 \\ 0.3 & -0.6 & 0.3 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

Lastly, we consider parameters $(k, \theta, \kappa, \mu, \xi) = (2.0, 2.0, 0.05, 0.1, 0.07)$ for the stochastic volatility model, and normally distributed jumps, with size $Z \sim N(0.3, 0.5)$ and rate $\lambda = 1.0$.

Fig. 4 shows the PD functions resulting from DNN models trained with hidden layers ranging from 1 to 5. For all three models, we can easily see that a single hidden layer is insufficient, yielding solutions that do not adequately follow the desired properties that characterize the PD process, such as monotonicity and boundedness. However, adding only one more hidden layer vastly improves the results from all models, and when using $L \geq 3$ hidden layers we obtain nearly indistinguishable solutions.

7.2. **Two dimensional input layer (initial position and maturity).** We now extend the DNN models by creating models with both spatial and temporal inputs, as shown in Fig. 5. Using the estimations above, we will consider 5 hidden layers and 30,000 training points. Note that each training point now consists of an initial position, maturity time (equivalently, time until maturity u, since we keep $t = 0$; we proceed with this terminology hereinafter) and payoff value, allowing the networks to "learn" how the maturity time affects the PD values. Fig. 6 displays the resulting

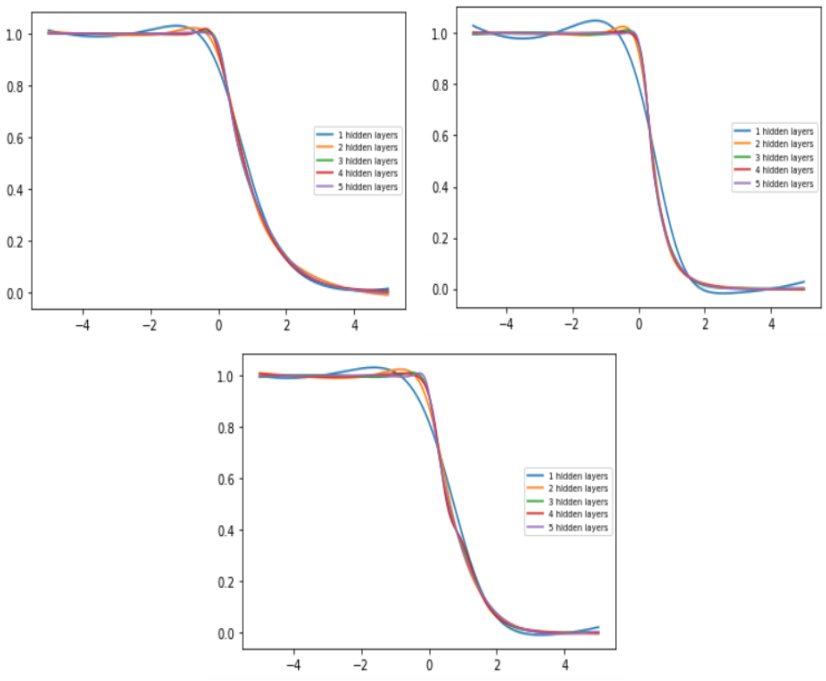

FIGURE 4. Graphs of the PD as a function of the initial position for (top left) the DNN model with $\Psi(x, 0)$, (top right) the DNN model with $\Psi(x, \rho, 0)$ and $\rho = 2$, and (bottom) the DNN model with $\Psi(x, y, 0)$ and $y = 0.2$.

survival, as functions of the initial position and time until maturity, estimated by the DNN models (note that in many of the figures throughout the remainder of the paper we display the survival rather than the default function, which is more commonly used in the literature. Naturally, the methods are identical and one can easily obtain the one function from the other). Despite the additional computational power required for the training of DNN model with 2 inputs (due to the increased number training data and model parameters), their advantages over the corresponding DNNs with a single input are obvious; with a single model we obtain the full term structure of the PD function that can be used for the many applications described. The extra computational cost is inconsequential relative what would be required for training multiple models for different maturity times.

7.3. **Extending the input dimension (latent variables).** We have seen that the model can be trained on paths with variable initial position as well as maturity time, thereby resulting in a Neural Network with two inputs. However, in the case of the regime switching and stochastic volatility models it is important to also account for the the latent variables. We can do this by considering another model input representing the initial regime or volatility, respectively. Indeed, a Neural Network with such an input is more appropriate for comparisons with the Finite Difference approach, where we obtain solutions across the regimes and volatility grid.

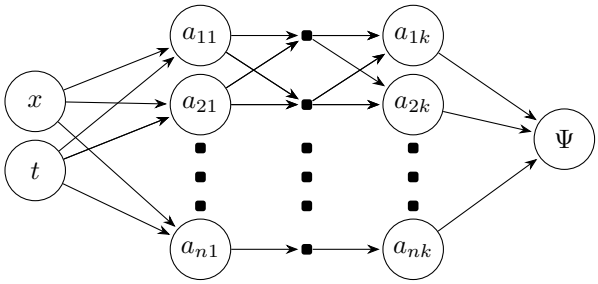

FIGURE 5. Fully connected Neural Network with a two dimensional input layer, k hidden layers and a one dimensional output layer.

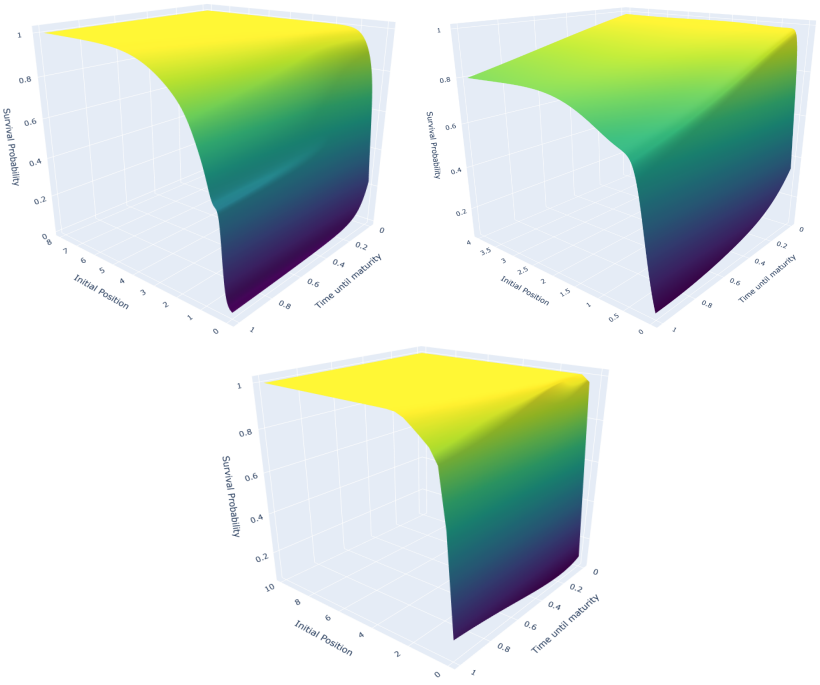

FIGURE 6. (Left) DNN model for $\Phi(x, u)$ (Middle) DNN model for $\Phi(x, \rho, u)$ with $\rho = 2$ (Right) DNN model for $\Phi(x, y, u)$ with $y = 2.0$.

The approach is similar to that described above. We generate paths of the (regime switching or stochastic volatility) model by drawing random initial values of the latent variables, in combination with the initial position and maturity time. Training is then done in the same fashion, using the appropriate payoff function. This extension requires only a simple addition to the input layer of the Neural Network, which now becomes three-dimensional.

It is worth noting that various architectures for the Neural Network may need to be considered before arriving at an appropriate and satisfactory model. In particular, when considering 5 hidden layers, each with 10 neurons, as above, the resulting PD functions do not satisfy standard properties such as monotonicity or stability. It is possible that such occurrences are a case of overfitting, which Neural Networks are often prone to, due to the large number of trainable parameters. For the estimation of the PD functions, we concluded on 3 hidden layers each with 10 neurons for the regime switching model and 3 hidden layers with 7 neurons for the stochastic volatility model. For these higher dimensional models, 50,000 path samples were used for training. Fig. 7 displays the resulting PD functions for the given choices of latent variables (which are now an input for the Neural Network model, rather that fixed parameters as in the models previously developed). Hence, the resulting PD functions are now directly comparable to those obtained from the FD scheme, which we will discuss in the sequel.

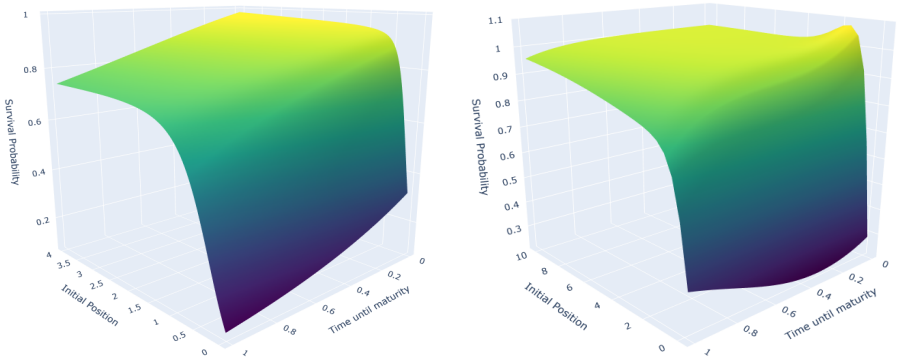

FIGURE 7. (Left) DNN model for $\Phi(x, \rho, u)$ with $\rho = 2$. (Right) DNN model for $\Psi(x, y, u)$ with $y = 2.0$.

7.4. **DNN model for the generalized probability of default.** We are now in the position to utilize the DNN framework to address one of the modelling problems presented in the beginning of the paper: calculating the PD function under the generalized asset value model. A prolific issues when considering more complex models for PD value estimations is the "curse of dimensionality", as faced when considering the Finite Difference method for the PIDE arising from the generalized asset process, given by (9). This issue is exacerbated when considering equations that have an integral term, since the matrices in the FD schemes are then non-sparse, further increasing the computationally complexity. Using the DNN framework we can overcome this issue for such complex models. Indeed, Neural Networks are considered superior in precisely such cases, i.e., in the existence of large, high dimensional data, and we will take advantage of this to estimate the PD function under the generalized model $\Psi(x, \rho, y, u)$.

The extension is relatively straightforward, as the only structural change that is required is adding another dimension to the input layer, with the resulting architecture shown in Fig. 8. Naturally, additional complexity arises due to the simulation process, as both latent variables must now be simulated in order to generate the required training data. A detailed example with the corresponding parameters is

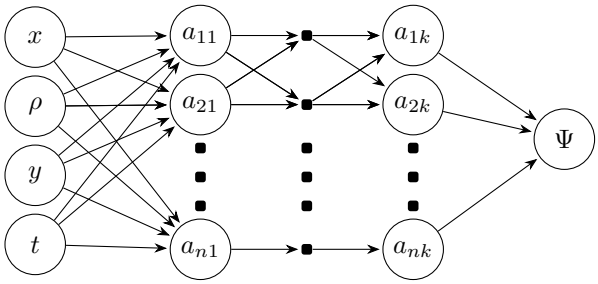

FIGURE 8. Fully connected Neural Network with a four-dimensional input layer, k hidden layers and a one-dimensional output layer.

given below, as this model has not been explicitly dealt with in the previous numerical methods.

Example 7.2. Consider the generalized model (5), with parameters of the asset value process (now depending on the underlying regime) given by:

$$(k_\rho, \theta_\rho, \sigma_\rho) = \begin{cases} (0.3, 0.8, 0.3) & \text{if } \rho = 0, \\ (0.2, 0.5, 0.5) & \text{if } \rho = 1, \\ (0.1, 0.6, 0.4) & \text{if } \rho = 2, \end{cases}$$

parameters of the volatility process $(\kappa, \mu, \xi) = (0.05, 0.1, 0.07)$, and jumps are again normally distributed, with size $Z \sim N(0.3, 0.5)$ with rate $\lambda = 1.0$. We train a Neural Network with 3 hidden layers and 7 neurons per layer (further discussion on hyperparameter selection is given in Section 7.5.1). For the training, 60,000 paths of the generalized asset process are simulated. The resulting PD function for a selected pair of initial regime and volatility values is given in Fig. 9.

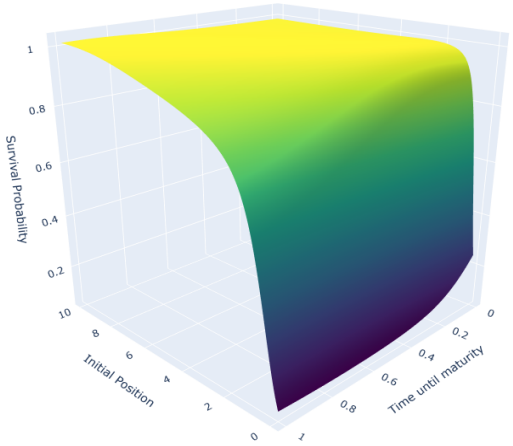

FIGURE 9. DNN model for $\Psi(x, \rho, y, u)$ with $\rho = 1$ and $y = 2.0$.

This application displays the utility of using DNNs to estimate the PD functions. The additional variables do not create undue effort when constructing the model,

as the only additional effort comes from simulating the corresponding asset value process. On the other hand, a Finite Difference scheme to solve PIDE (9) and obtain PD function under the generalized model would be extremely complicated due to the number of terms, the structure of the scheme required to ensure monotonicity and stability properties hold (i.e., utilizing the ADI scheme as detailed for the stochastic volatility model), the combination of terminal conditions and, finally, the dimension of the resulting matrices constructed during the iterations of the numerical scheme. In practice, such a scheme would require significantly more computational power than the training of the corresponding DNN.

7.5. **Comparisons and errors.** In this Section we will quantify and further discuss the main difficulties and errors that arise in estimating PD functions from the Neural Network models, using the estimates obtained from Finite Difference methods as a baseline for the comparisons, as these are staples in the field of numerical methods. Finite Difference schemes entail an appropriate discretization of the spatial and temporal variables. Furthermore, we must also discretize the integral term. Using an appropriate implicit scheme, often referred to as Backward Time Central Space implicit, we can estimate the solutions to the PDEs by solving the matrix equations that arise from the discetization. Such methods for PIDEs have been studied in e.g., [18], with schemes specific to the PIDEs arising in this paper given in [26] (practitioners should also note that it is important to ensure the selection of the grid is such that stability and monotonicity of the FD schemes hold. This may cause significant increase in the computational power required to implement these methods, but is necessary for correct estimations. For brevity, we omit further details pertaining to these FD schemes and refer the reader to additional).

The imposition of strict requirements for the grid, coupled with the high dimensionality of the regime switching and stochastic volatility models, makes it extremely difficult to estimate the PD function under the generalized model using FD schemes, due to the curse of dimensionality. Therefore, we will mainly focus on comparisons of the PD values obtained under the three models for which we are able to examine the solutions using both methods. Furthermore, we give details pertaining to the computational time required for both approaches, which is also directly related to the Neural Network's ability to overcome the issues pertaining to the computational cost of increased dimensionality, thereby shedding light on the considerations that practitioners must take into account when deciding on a methodology for their credit risk modelling framework.

7.5.1. *Errors in the PD functions.* To compare the solutions we will follow [24] and consider the relative error, using the Finite Difference survival probability $\Phi(x,t)$ value as a benchmark (we will work with the survival probabilities). Hence, we measure the error function as the quantity:

$$\epsilon(x,t) = \frac{|\Phi^{NN}(x,t) - \Phi^{FD}(x,t)|}{\Phi^{FD}(x,t)}.$$

For the vast majority of credit modelling tasks have discussed we are mainly interested in the error within a domain sufficiently far away from the boundary. It is within this space that possible errors would have the largest effect when using the PD function in practice and it is therefore of great importance to ensure that the two methods deliver PD values as close as possible. Fig. 10 displays the relative difference function for the three main models we have considered, from which

we obtain maximum errors of approximately 1.6%, 6.0% and 15.8% for the one dimensional, regime switching and stochastic volatility models respectively, when examined within selected domains illustrated in the corresponding graphs. The increase in relative error as we consider more complicated models is expected, given the complexity of the underlying PIDE, which creates the need for significantly more asset value path simulations upon which the Neural Network will be trained.

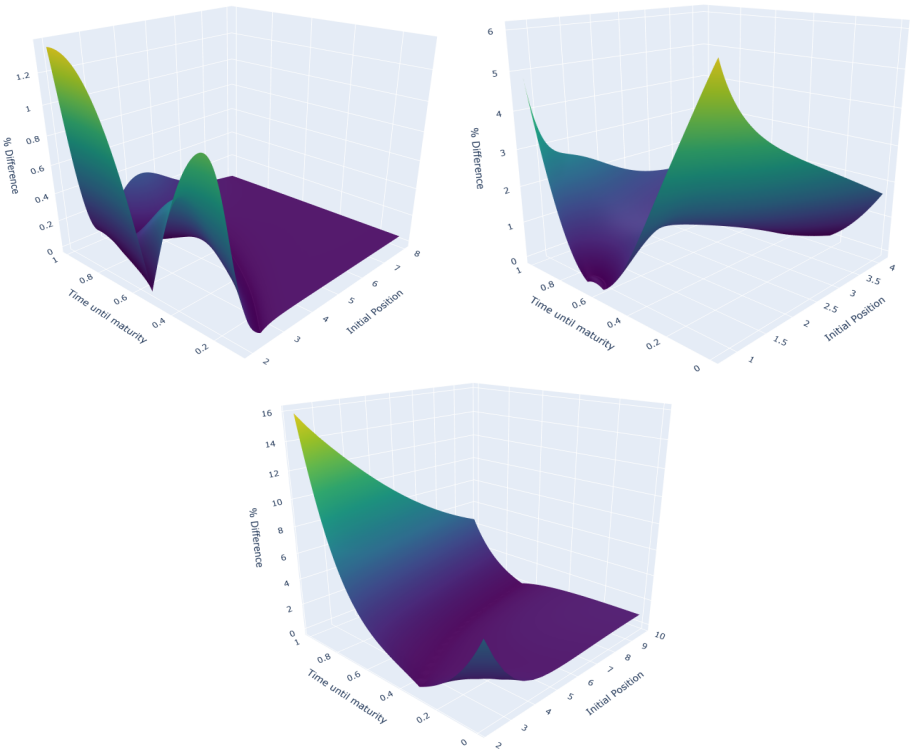

FIGURE 10. Relative error of the survival probability from the NN models under the (top left) one-dimensional, (top right) regime switching and (bottom) stochastic volatility asset value model.

However, to fully compare the DNN model to the FD scheme, we must also consider points near and on the boundaries. To the best of our knowledge the effects of the initial and boundary conditions have not been carefully examined. In [24], the DNN model is compared to Monte Carlo methods only on the interior of the domain, as previously described. Fig. 11 displays the absolute error functions, $|\Phi^{NN}(x,t) - \Phi^{FD}(x,t)|$, in the whole domain. It can be seen that the approximation provided by the DNN is very satisfactory in the interior of the domain under consideration. However, it is worth emphasizing that this does not remain true near the boundary, thus raising concerns with respect to the effectiveness of DNNs in handling boundary conditions. This is clearly a point deserving further theoretical and numerical investigation. At present we can only postulate concerning the reasons of this behavior. From detailed experiments we have conducted (not reported here) the inclusion of the boundary terms in the loss function does not seem to significantly improve this behavior. Additionally, we a) increased the number of total

simulated asset value paths used for training, and b) increased the points generated from the initial and boundary conditions, however no significant improvements in the errors were observed. We posit that this could be explained by the fundamental difference in handling the boundary and initial conditions in the two methods; in the FD schemes recall that we manually impose the boundary and initial conditions by appropriately constructing the coefficient matrices, but on the other hand, no such conditions are directly imposed when training the DNN model. This means that the model must "learn" these solutions using the paths and corresponding payoff values. Hence, it is expected that the differences are largest near this set of points, and the effect of these errors decreases further within the spatial and temporal domain, leading to the insignificant differences observed in Fig. 10.

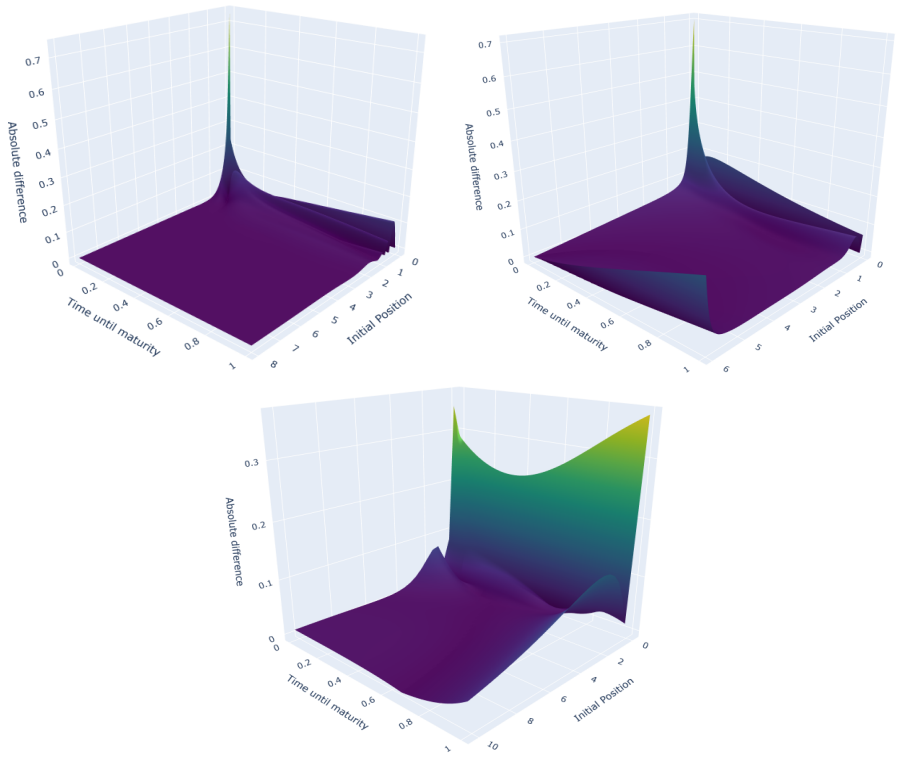

FIGURE 11. Absolute error of the survival probability from the NN models under the one-dimensional (top left), regime switching (top right) and stochastic volatility (bottom) asset value model.

However, it is worth emphasizing that the fact the DNN approximation yields large errors on the boundary does not affect its applicability for practitioners. In particular, the domain of the relevant variables in the model which are of interest and use to practitioners for scenario analysis and decision making are well within the interior of the domain, which is in the region where the DNN provides a perfectly acceptable (almost indistinguishable as compared with the FD) approximation of the solution. Nonetheless, from the theoretical and numerical analysis aspect we are compelled to report these errors as a potential disadvantage of the Neural Network

framework for PD estimation, particularly given the fact that these models are still "black-boxes", making it difficult to understand the reason the large errors arise. Furthermore, estimating the magnitude of the errors may also be difficult as it requires having a benchmark, such as the FD solution, which will often suffer from the "curse of dimensionality", thereby inducing a large risk by the DNN for more complex asset value models, since the errors cannot be estimated. Despite this risk, as we have now validated that the solutions in the interior the domains (i.e., sufficiently far from the boundaries) are nearly identical for a large family of stochastic models, the Neural Networks models can be powerful tools to efficiently deal with increasingly complex asset value models and predict the corresponding PD functions.

7.5.2. *Applicability of the PD functions and computational requirements.* In addition to the numerical, it is important for practitioners to discuss the structural differences between the two approaches examined. In both methods we have been able to obtain the PD estimates under the regime switching and stochastic volatility models as a functions of the spatial, temporal and latent variables (regime or volatility, respectively), i.e., $\Psi(x, \rho, t)$ and $\Psi(x, y, t)$. However, there exists a significant difference in the resulting PD functions: the FD method provides PD estimates covering values of the variables only at the specific grid points created by the discretization (x_i, t_j) for $(i, j) \in \{0, 1, \dots, N\} \times \{0, 1, \dots, M\}$, whereas the Neural Network leads to a solution for every $(x, t) \in \mathcal{D} \times [0, T]$. This is an important advantage of the DNN models since it means that we are able to obtain the entire term structure of the PD function without requiring interpolation or similar approximation techniques, which would be necessary when using the FD scheme. Naturally, this advantage is accompanied with an increase in computational cost, as we will see below. Depending on the application, business needs and setting, practitioners may prefer one of these methods based on this trade-off. For example, if the application requires estimating the PD at any possible values of the initial position and maturity it would be preferable to use the DNN model and take advantage of the ability to predict the PD value for any given (x, t). On the other hand, if we are interested in estimations with a predetermined initial value and maturity input, such as in e.g., derivatives pricing, the FD is useful due to its computational accessibility.

To further illustrate these points, we will compare the time required to estimate the PD function with the FD and DNN methods. Table 2 displays these results for each of the three asset value models. When applying the Finite Difference methods $N \times M = 1001 \times 101$ was used for the one dimensional model, where N and M are the sizes of the spatial and temporal discetizations, respectively. For the regime switching and stochastic volatility models we used grids of sizes $N \times T \times R = 1001 \times 1001 \times 3$ and $N \times T \times V = 201 \times 1001 \times 201$, respectively. On the other hand, as outlined in Table 3, 30,000 paths were used to train the DNN under the one-dimensional model and 50,000 under the regime switching and stochastic volatility models. All experiments were conducted in *Python* 3.8.8, run on a Windows 10 Pro with Intel(R) Core(TM) i5-1035G1 CPU Processor; the Finite Difference methods were implemented using *Pandas, NumPy, SciPy* and the Neural Network were trained using *TensorFlow*.

Model	FD approximation time (seconds)	DNN training time (seconds)
One dimensional	209	66,243
Regime switching	1,141	96,699
Stochastic volatility	154,542	190,734
Generalized	Intractable	194,208

TABLE 2. Comparison of computational time required for FD and NN methods.

Note that the DNN training times given in Table 2 also account for the time required to simulate the asset value paths. These results provide a quantification of various points previously addressed; firstly, we can see that indeed the FD schemes are easier to implement in terms of computational cost. Particularly, the PD estimations under the one dimensional and regime switching can be obtained very easily. However, the effects of the "curse of dimensionality" are evident, as the time required to obtain the PD estimations under the stochastic volatility model drastically increases. It is easy to see the increase in computational power required to develop the DNN models, compared to the FD solution, for the one dimensional and regime switching asset processes is significant. Interestingly, the same does not apply in the case of the stochastic volatility model, where the computational time for the two methods are very similar, attesting to the Neural Network's ability for efficient generalization. This is further validated when observing that the DNN for the PD under the generalized model requires approximately 194,208 seconds, a rather insignificant difference compared to the stochastic volatility case, whereas the corresponding FD scheme is practically intractable.

8. **DNN models for scenario analysis in credit risk.** As shown above, DNNs can be largely beneficial when considering problems in high dimensional spaces. This stems from the remarkable property that such models enjoy regarding their ability to "learn" complicated functions via the architecture and activation functions used. From a theoretical standpoint the this ability is described by the Universal Approximation Theorem (see e.g., [16, 42] for details). DNN architectures rely on this mathematical framework, and fascinating results surrounding this field have been studied in depth in the context of Hilbert's famous 13th Problem and celebrated results by Kolmogorov and Arnold ([13]). In this section we will consider how DNNs can be used for more general credit risk modeling tasks, namely scenario analysis. We will see that this problem can be tackled by considering a DNN model with a high dimensional input corresponding to the asset process parameters. We begin by discussing the use and applicability of scenario analysis in credit risk.

8.1. **Scenario analysis in credit risk modeling.** Understanding the construction of the DNN allows us to also understand its benefits and how these can be maximized by practitioners. Specifically in the field of credit risk, we are often interested in various (optimistic, pessimistic) scenarios which can largely affect individual and portfolio provisions. One way of capturing such cases is to incorporate a regime switching model, as we have already seen. However, it is also important to consider the cases in which credit managers want to account for the effect of non-observable factors that cause changes in the underlying model. In such cases, it is more appropriate to consider a family of stochastic process to which the asset

value process belongs. This can be done by considering a model for the asset value process with parameter vector $\mathbf{\Theta} = (\theta_1, \dots, \theta_n)$, which follows an appropriate multidimensional distribution function, $\mathbf{\Theta} \sim F$, i.e., $\theta_i \sim F_i$ for all i and F_i is defined on an appropriate support $\mathcal{S}_i \subset \mathbb{R}$. Naturally, significant differences can then occur in credit risk modeling tasks, such as provisioning. To incorporate such dynamics into a single model we can use a DNN where the input now consists of the parameter vector $\mathbf{\Theta}$.

Considering such a model in the framework of credit risk has important implications. Given that the parameters of the model themselves now vary, in order to reach deterministic results for provisions one would have to be able to accurately estimate the point-in-time value of the parameters $\theta_1, \dots, \theta_n$ or, alternatively, average across the support of their distributions in order to obtain the expected value of the required results. By introducing some simple notation, these cases can easily be represented mathematically. To this end, consider, as above, the asset value process G_t, with $G_0 = x$ and a *risk function* $R : \mathcal{D} \times \mathcal{S} \to \mathbb{R}$, such that $R(x, \mathbf{\Theta})$ represents the result of the credit risk modelling task. Hence, if $\mathbf{\Theta}$ is observable then it is obvious that the risk function is also calculable. On the other hand, if the parameters of the asset value process can not be directly observed (as is the case in most real-life applications) we can use the distribution of the parameter vector to calculate the expected value of the risk function. This gives:

$$R(x) := \mathbb{E}[R(x, \mathbf{\Theta})] = \int_{\mathbb{R}^n} R(x, \mathbf{\Theta}(\mathbf{z}))d\mathbf{F}(\mathbf{z}). \tag{26}$$

The above estimation can be used for all risk modelling problems, under a model uncertainty framework. In particular, provisioning tasks under IFRS 9 now depend directly on the models used for forecasting future losses and therefore, incorporating the more general family of stochastic processes is a way to account for infinitely many scenarios which affect the evolution of individual exposures and, by extension, the credit portfolio. The Deep Learning framework used to estimate PD processes can be used to address these problems; as mentioned, this will require appropriate changes in the input layer of the network (see Fig. 12) as well as the subsequent training process.

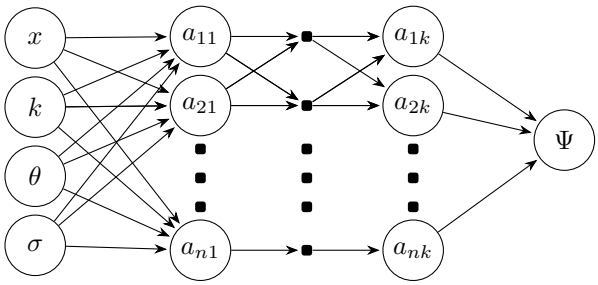

FIGURE 12. Fully connected Neural Network with a 4-dimensional input layer and k hidden layers.

8.2. **DNN model for PDs under a family of stochastic asset processes.** In the setting described above, we consider a DNN with 4 inputs: the initial position

and the elements of the parameter vector of the asset process (for illustrative purposes we do not consider the temporal input, but this can be easily added, as seen above). Hence, in order to create the training data we simulate from the family of OU processes by randomly generating values of the parameter vector and then simulating the evolution of the process with the given combination of parameters. Naturally, adequate training of the DNN may require a significant increase in the volume of training data, as well as in the time required to train the model; this however is counteracted by the advantage of obtaining a model capable of producing the PD process for an entire family of asset value processes, without the need to re-train the DNN when considering different asset value models. Even though the application of such a model is straightforward given the general DNN architecture, for completeness, we give an example below.

Example 8.1. Consider the one dimensional asset value model (16) where the stochastic coefficients $\boldsymbol{\Theta} = (k, \theta, \sigma)$ are each randomly sampled from the same distribution $Unif(0.0, 5.0)$, the Lévy jumps are normally distributed with $N(0.0, 0.5)$ and the jump rate is $\lambda = 1.0$. For simplicity, we also consider a fixed maturity, $T = 1.0$. To illustrate the usefulness of this approach, we tested various number of simulations and model architectures, from which we found that satisfactory models can be developed with as low as 30,000 simulations of the the asset value process and a Neural Network with 3 hidden layers and 7 neurons per layer. The resulting model has a four-dimensional input layer (x, k, θ, σ), and, by fixing the parameter set we can obtain the PD as a function of the initial position, as required for the various IFRS 9 modelling tasks previously addressed. Fig. 13 displays the resulting PD functions predicted by the DNN for four parameter sets:

$$(k, \theta, \sigma) = \begin{cases} (0.5, 0.5, 0.5), \\ (2.0, 2.0, 2.0), \\ (3.0, 3.0, 3.0), \\ (4.0, 4.0, 4.0). \end{cases}$$

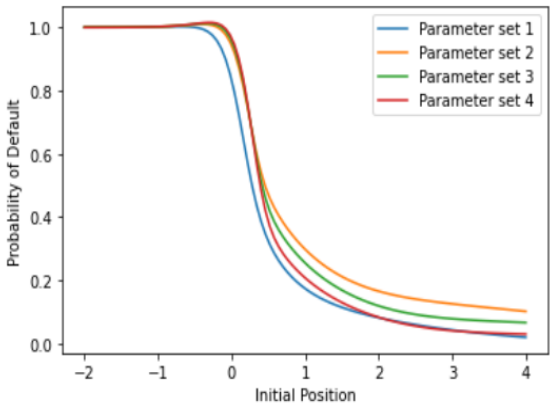

FIGURE 13. PD functions for various choices of asset value process parameters.

Naturally, if any of the parameters are observable these can be considered constant and the remaining can be used for the risk function calculation. We believe that this approach can be useful to practitioners as it also provides the necessary tools to calculate the probabilities of various scenarios occurring, by simply using the distribution of the parameter set.

Remark 8.2. It is important to remark on the selection of hyperparameters for training the Neural Networks. For the models developed above, many different combinations of layers and neurons per layer were tested, with many choices leading to nonsensical PD functions. Multiple tests showed that, in such cases, choosing different hyperparameters in combination with increasing the number of sample paths for training proved to be most effective. For all experiments a softplus activation function and Glorot uniform kernel initializer are used (changing these did not improve the resulting models). Furthermore, following [24], the models were trained over 10,000 epochs with an Adam optimizer and a piecewise decaying learning rate of 0.1, 0.01, 0.001 and 0.0001 up until 2000, 4000, 6000 and 10,000 epochs, respectively. Finally, Table 3 shows the choices of training path simulations, layers and neurons per layer for these DNN models.

Asset value model	Simulated paths	Layers	Neurons per layer
One dimensional	30,000	5	10
Regime switching	50,000	3	10
Stochastic volatility	50,000	3	7
Generalized	60,000	3	7

TABLE 3. Neural Network architectures for the three asset value models.

An interesting observation is that, for the more complex asset value processes, the best models were those with less trainable parameters than the model for the PD in the one dimensional case, i.e., less layers and/or neurons per layer. Simultaneously, more asset value paths simulations were used for training as the complexity of the asset value process increases. Therefore, one possible explanation is that the combination of these two changes assists in avoiding overfitting, which Neural Networks are known to be prone to.

9. **Managerial insights and practical implications.** This research shows that the DNN framework for PD estimation can be extremely useful for practitioners for a wide range of tasks, including provisioning and scenario analysis, which become more demanding under IFRS 9. Their applicability lies not only in the capabilities for generalization, but also in the advantages compared to FD difference methods, such as the fact that they can produce PD values for every (x, t), not only at points on a pre-defined grid. Furthermore, we have shown that we can train DNNs to produce the PD as a function of the input parameters. Therefore, it is expected that such novel approaches will be not only applicable but necessary as the complexity of credit risk modelling, and the corresponding regulations it must adhere to, increase in the years to come. Management and practitioners will need to take such considerations into account in order to create robust credit risk modelling frameworks.

On the other hand, our results show that large discrepancies between the two approaches can arise near the boundaries of the spatial and temporal domains, which

are still difficult to analyze due to the lack of rigorous explainability techniques for complex ML models with a large number of trainable parameters. Therefore, given the importance of the tasks entrusted to these models, practitioners must also weigh these current difficulties associated to the DNN models. Particularly in the case of systemic financial institutions, models with such characteristics may not adhere to the regulatory requirements.

10. **Conclusion and outlook.** We have seen in this paper that the problem of estimating PDs under various stochastic asset value models can be efficiently tackled using Machine Learning approaches and particularly Neural Networks. Specifically, we have shown that:

- We can train DNN models to produce the PD functions under sophisticated dynamics, such as the generalized model (5).
- Furthermore, we can harness the DNN's capabilities for generalization and handling high-dimensional data to train models where the parameters of the stochastic processes are handled as inputs, thereby creating a framework that can estimate PD functions under a family of asset value model, as described in Section 8.2.
- Important difficulties can arise in the development of these models and have to be carefully considered and implemented by practitioners.

One of the most consequential difficulty that need be addressed in future work is the selection of model architecture, since, as seen, the best choices may be counter-intuitive, in the sense that more trainable parameters do not always result in better models. It is probable that this observation is closely related to the overfitting problem exhibited by Neural Networks in various settings (see e.g., [15, 11]). Note that new approaches have been suggested to rectify this problem; for example, in [57], the authors develop the "Dropout" method for training NNs which is applied to supervised learning problems. Future research could consider such overfitting-prevention techniques in the context of the unsupervised learning problem of solving PDEs and PIDEs. Furthermore, comparisons of the DNN models with the standard FD approaches indicate that future research should focus on bridging the numerical, as well as conceptual gaps between the novel ML and established FD methods, in order to safely deploy such models to solve a multitude of credit risk modelling tasks.

Appendix A. **PIDEs for the PD functions under the simplified models.** For completeness, in this section we explicitly write the PIDEs under the simplified models we examine in this paper.

A.1. **One dimensional model.** Under model (1) the survival probability $\Phi(x,u)$ satisfies the PIDE:

$$\frac{\partial \Phi}{\partial u} = k(\theta - x)\frac{\partial \Phi}{\partial x} + \frac{1}{2}\sigma^2 \frac{\partial^2 \Phi}{\partial x^2} + \int_{\mathbb{R}} \Big(\Phi(x+z,u) - \Phi(x,u)\Big)\nu(dz) = 0, \tag{27}$$

for $(x,u) \in \mathcal{D} \times [0,T]$ and with initial and boundary conditions:

$$\begin{aligned} \Phi(x,0) &= \mathbb{1}_{\{x>0\}}, \quad x \in \mathcal{D}, \\ \Phi(0,u) &= 0, \quad u \in [0,T], \\ \Phi(x,u) &\to 1 \text{ as } x \to \infty, \quad u \in [0,T]. \end{aligned} \tag{28}$$

A.2. **Regime switching model.** Under the regime-switching model (2), the survival probability $\Phi(x,\rho,u)$ satisfies the PIDE:

$$\begin{aligned}&\frac{\partial\Phi}{\partial u}(x,\rho,u) \\ &= k_\rho(\theta_\rho - x)\frac{\partial\Phi}{\partial x}(x,\rho,u) + \frac{1}{2}\sigma_\rho^2\frac{\partial^2\Phi}{\partial x^2}(x,\rho,u) + \sum_{j\neq\rho} q_{\rho j}\Big(\Phi(x,j,u) - \Phi(x,\rho,u)\Big) \\ &\quad + \int_{\mathbb{R}}\Big(\Phi(x+z,\rho,u) - \Phi(x,\rho,u)\Big)\nu(dz), \quad (x,\rho,u)\in\mathcal{D}\times\mathcal{R}\times[0,T],\end{aligned} \tag{29}$$

with initial and boundary conditions:

$$\begin{aligned}\Phi(x,\rho,0) &= \mathbb{1}_{\{x>0\}}, \quad (x,\rho)\in\mathcal{D}\times\mathcal{R},\\ \Phi(0,\rho,u) &= 0, \quad (\rho,u)\in\mathcal{R}\times[0,T],\\ \Phi(x,\rho,u)\to 1 \text{ as } x\to\infty, &\quad (\rho,u)\in\mathcal{R}\times[0,T],\end{aligned}$$

where q_{ij} are the elements of the generator Q of the Continuous time Markov chain representing the switching pocess, R_t.

A.3. **Stochastic volatility model.** Under the stochastic volatility model (4), the survival probability $\Phi(x,y,u)$ satisfies the PIDE:

$$\begin{aligned}&\frac{\partial\Phi}{\partial u}(x,y,u) \\ &= k(\theta - x)\frac{\partial\Phi}{\partial x}(x,y,u) + \kappa(\mu - y)\frac{\partial\Phi}{\partial y}(x,y,u) + \frac{1}{2}y\frac{\partial^2\Phi}{\partial x^2}(x,y,u) + \frac{1}{2}\xi^2 y\frac{\partial^2\Phi}{\partial y^2}(x,y,u) \\ &+ \int_{\mathbb{R}}\Big(\Phi(x+z,y,u) - \Phi(x,y,u)\Big)\nu(dz), \quad (x,y,u)\in\mathcal{D}\times\mathcal{V}\times[0,T],\end{aligned} \tag{30}$$

subject to the initial and boundary conditions:

$$\begin{aligned}\Phi(x,y,0) &= \mathbb{1}_{\{x>0\}}, \quad (x,y)\in\mathcal{D}\times\mathcal{V},\\ \Phi(0,y,u) &= 0, \quad (y,u)\in\mathcal{V}\times[0,T],\\ \Phi(x,y,u)\to 1 \text{ as } x\to\infty, &\quad (y,u)\in\mathcal{V}\times[0,T],\\ \frac{\partial\Phi}{\partial y}(x,y,u) = 0 \text{ as } y\to\infty &\quad (x,u)\in\mathcal{D}\times[0,T].\end{aligned} \tag{31}$$

REFERENCES

[1] P. M. Addo, D. Guegan and B. Hassani, Credit risk analysis using machine and deep learning models, *Risks*, **6** (2018), 38.

[2] D. Applebaum, *Lévy Processes and Stochastic Calculus*, Cambridge university press, Cambridge, 2009.

[3] C. Aristidou, The meta-phillips curve: Modelling us inflation in the presence of regime change, *Journal of Macroeconomics*, **57** (2018), 367-379.

[4] L. Ballotta and G. Fusai, Counterparty credit risk in a multivariate structural model with jumps, *Finance*, **36** (2015), 39-74.

[5] O. E. Barndorff-Nielsen and N. Shephard, Modelling by lévy processess for financial econometrics, *Lévy Processes*, Birkhäuser Boston, Inc., Boston, MA, 2001, 283-318.

[6] O. E. Barndorff-Nielsen and N. Shephard, Non-gaussian ornstein–uhlenbeck-based models and some of their uses in financial economics, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63** (2001), 167-241.

[7] D. S. Bates, Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options, *The Review of Financial Studies*, **9** (1996), 69-107.

[8] C. Beck, S. Becker, P. Grohs, N. Jaafari and A. Jentzen, Solving the kolmogorov pde by means of deep learning, *Journal of Scientific Computing*, **88** (2021), 1-28.
[9] D. Beerbaum, Significant increase in credit risk according to ifrs 9: Implications for financial institutions, *International Journal of Economics & Management Sciences*, **4** (2015), 1-3.
[10] E. Benhamou, E. Gobet and M. Miri, Time dependent heston model, *SIAM Journal on Financial Mathematics*, **1** (2010), 289-325.
[11] I. Bilbao and J. Bilbao, Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks, In *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, IEEE, 2017, 173-177.
[12] J. Blechschmidt and O. G. Ernst, Three ways to solve partial differential equations with neural networks-a review, *GAMM-Mitteilungen*, **44** (2021), e202100006.
[13] V. Brattka. From Hilbert's 13th Problem to the theory of neural networks: Constructive aspects of Kolmogorov's Superposition Theorem, *Kolmogorov's Heritage in Mathematics*, Springer, Berlin, 2007, 253-280.
[14] S. Cai, Z. Wang, S. Wang, P. Perdikaris and G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, *Journal of Heat Transfer*, **143** (2021), 060801.
[15] R. Caruana, S. Lawrence and C. Giles, Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping, *Advances in Neural Information Processing Systems*, **13** (2000).
[16] T. Chen and H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, *IEEE Transactions on Neural Networks*, **6** (1995), 911-917.
[17] J. M. Clements, D. Xu, N. Yousefi and D. Efimov, Sequential deep learning for credit risk monitoring with tabular financial data, arXiv preprint, arXiv:2012.15330, 2020.
[18] R. Cont and E. Voltchkova, A finite difference scheme for option pricing in jump diffusion and exponential lévy models, *SIAM Journal on Numerical Analysis*, **43** (2005), 1596-1626.
[19] J. C. Cox, J. E. Ingersoll Jr and S. A. Ross, A theory of the term structure of interest rates, In *Theory of Valuation*, World Scientific, (2005), 129-164.
[20] S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi and F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, *Journal of Scientific Computing*, **92** (2022), Paper No. 88, 62 pp.
[21] J.-C. Duan, I. Popova and P. Ritchken, Option pricing under regime switching, *Quantitative Finance*, **2** (2002), 116-132.
[22] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, K. Leyton-Brown, et al., Towards an empirical foundation for assessing bayesian optimization of hyperparameters, In *NIPS Workshop on Bayesian Optimization in Theory and Practice*, **10** (2013).
[23] A. Fadlalla and C.-H. Lin, An analysis of the applications of neural networks in finance, *Interfaces*, **31** (2001), 112-122.
[24] R. Frey and V. Köck, Deep neural network algorithms for parabolic pides and applications in insurance mathematics, In *Mathematical and Statistical Methods for Actuarial Sciences and Finance: MAF 2022*, Springer, (2022), 272-277.
[25] K. Georgiou, G. N. Domazakis, D. Pappas and A. N. Yannacopoulos, Markov chain lumpability and applications to credit risk modelling in compliance with the international financial reporting standard 9 framework, *European Journal of Operational Research*, **292** (2021), 1146-1164.
[26] K. Georgiou and A. N. Yannacopoulos, Probability of default modelling with l\'evy-driven ornstein-uhlenbeck processes and applications in credit risk under the ifrs 9, arXiv preprint, arXiv:2309.12384, 2023.
[27] S. Goutte, Pricing and hedging in stochastic volatility regime switching models, *Journal of Mathematical Finance*, **3** (2013), 70-80.
[28] S. Goutte and A. Ngoupeyou, Conditional laplace formula in regime switching model: Application to defaultable bond, Preprint, 2011.
[29] T. Guo, J. Dong, H. Li and Y. Gao, Simple convolutional neural network on image classification, In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, IEEE, (2017), 721-724.
[30] D. Hainaut, Financial modeling with switching lévy processes, ESC, 2011.
[31] D. Hainaut and O. L. Courtois, An intensity model for credit risk with switching lévy processes, *Quantitative Finance*, **14** (2014), 1453-1465.
[32] J. D. Hamilton, Regime switching models, In *Macroeconometrics and time series analysis*, Springer, (2010), 202-209.

[33] S. Hayou, A. Doucet and J. Rousseau, On the selection of initialization and activation function for deep neural networks, arXiv preprint, arXiv:1805.08266, 2018.
[34] R. Hecht-Nielsen, Theory of the backpropagation neural network, In *Neural Networks for Perception*, Elsevier, (1992), 65-93.
[35] S. Hijazi, R. Kumar, C. Rowen, et al., Using convolutional neural networks for image recognition, *Cadence Design Systems Inc.: San Jose, CA, USA*, **9** (2015), 1.
[36] A. Khashman. Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications*, **37** (2010), 6233-6239.
[37] K. Kumar and G. S. M. Thakur, Advanced applications of neural networks and artificial intelligence: A review, *International journal of information technology and computer science*, **4** (2012), 57.
[38] A. E. Kyprianou, *Introductory Lectures on Fluctuations of Lévy Processes with Applications*, Springer Science & Business Media, 2006.
[39] L. Liao, H. Li, W. Shang and L. Ma, An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, **31** (2022), 1-40.
[40] R. Lotfi, A. Gholamrezaei, M. Kadłubek, M. Afshar, S. S. Ali and K. Kheiri, A robust and resilience machine learning for forecasting agri-food production, *Scientific Reports*, **12** (2022), 21787.
[41] R. Lotfi, K. Kheiri, A. Sadeghi and E. B. Tirkolaee, An extended robust mathematical model to project the course of covid-19 epidemic in iran, *Annals of Operations Research*, (2022), 1-25.
[42] L. Lu, P. Jin, G. Pang, Z. Zhang and G. E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, *Nature Machine Intelligence*, **3** (2021), 218-229.
[43] E. Luciano and W. Schoutens, A multivariate jump-driven financial asset model, *Quantitative Finance*, **6** (2006), 385-402.
[44] L. R. Medsker and L. C. Jain, Recurrent neural networks, *Design and Applications*, **5** (2001), 64-67.
[45] Y. Mishura and O. Ragulina, *Ruin Probabilities: Smoothness, Bounds, Supermartingale Approach*, ISTE Press, London; Elsevier, Inc., Oxford, 2016.
[46] C. M. Møller, Stochastic differential equations for ruin probabilities, *Journal of Applied Probability*, **32** (1995), 74-89.
[47] B. K. Øksendal and A. Sulem, *Applied Stocstic Control of Jump Diffusions*, volume 498. Springer, Berlin, 2007.
[48] O. Onalan, Financial modelling with ornstein-uhlenbeck processes driven by lévy process, In *Proceedings of the World Congress on Engineering*, **2** (2009), 1-3.
[49] G. Pang, L. Lu and G. E. Karniadakis, fpinns: Fractional physics-informed neural networks, *SIAM Journal on Scientific Computing*, **41** (2019), A2603-A2626.
[50] K. Papik, B. Molnar, R. Schaefer, Z. Dombovari, Z. Tulassay and J. Feher, Application of neural networks in medicine-a review, *Med Sci Monit*, **4** (1998), 538-546.
[51] H. Pratiwi, A. P. Windarto, S. Susliansyah, R. R. Aria, S. Susilowati, L. K. Rahayu, Y. Fitriani, A. Merdekawati and I. R. Rahadjeng, Sigmoid activation function in selecting the best model of artificial neural networks, In *Journal of Physics: Conference Series*, IOP Publishing, **1471** (2020), 012010.
[52] M. Raissi, P. Perdikaris and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, **378** (2019), 686-707.
[53] E. Savku and G.-W. Weber, A regime-switching model with applications to finance: Markovian and non-markovian cases, *Dynamic Economic Problems with Regime Switches*, **25** (2021), 287-309.
[54] E. Savku and G.-W. Weber, Stochastic differential games for optimal investment problems in a markov regime-switching jump-diffusion market, *Annals of Operations Research*, **312** (2022), 1171-1196.
[55] W. Schoutens and J. Cariboni, *Lévy Processes in Credit Risk*, volume 519. John Wiley & Sons, 2010.
[56] F. Shen, X. Zhao, G. Kou and F. E. Alsaadi, A new deep learning ensemble credit risk evaluation model with an improved synthetic minority oversampling technique, *Applied Soft Computing*, **98** (2021), 106852.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research*, **15** (2014), 1929-1958.
[58] I. Sutskever, J. Martens and G. E. Hinton, Generating text with recurrent neural networks, In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, (2011), 1017-1024.
[59] J. Toivanen, A componentwise splitting method for pricing american options under the bates model, In *Applied and Numerical Partial Differential Equations*, Springer, **15** (2010), 213-227.
[60] O. Vasicek, An equilibrium characterization of the term structure, *Journal of Financial Economics*, **5** (1977), 177-188.
[61] X. Xu, Estimating lifetime expected credit losses under ifrs 9, Available at SSRN 2758513, 2016.
[62] C. Zhou, A jump-diffusion approach to modeling credit risk and valuing defaultable securities, Available at SSRN 39800, 1997.
[63] C. Zhu, G. Yin and N. A. Baran, Feynman–kac formulas for regime-switching jump diffusions and their applications, *Stochastics An International Journal of Probability and Stochastic Processes*, **87** (2015), 1000-1032.

Received December 2023; revised February 2024; early access March 2024.