Pathwise CVA Regressions With Oversimulated Defaults

Lokman A Abbas-Turki, Stéphane Crépey, Bouazza Saadeddine

Pathwise CVA Regressions With Oversimulated Defaults*

Lokman A. Abbas-Turki[1], Stéphane Crépey[2] and Bouazza Saadeddine[3,4]

October 31, 2022

Abstract

We consider the computation by simulation and neural net regression of conditional expectations, or more general elicitable statistics, of functionals of processes (X, Y). Here an exogenous component Y (Markov by itself) is time-consuming to simulate, while the endogenous component X (jointly Markov with Y) is quick to simulate given Y, but is responsible for most of the variance of the simulated payoff. To address the related variance issue, we introduce a conditionally independent, hierarchical simulation scheme, where several paths of X are simulated for each simulated path of Y. We analyze the statistical convergence of the regression learning scheme based on such block-dependent data. We derive heuristics on the number of paths of Y and, for each of them, of X, that should be simulated. The resulting algorithm is implemented on a graphics processing unit (GPU) combining Python/CUDA and learning with PyTorch. A CVA case study with a nested Monte Carlo benchmark shows that the hierarchical simulation technique is key to the success of the learning approach.

Keywords: hierarchical simulation, neural net regression, machine learning, X-valuation adjustment (XVA).

Mathematics Subject Classification: 91B25, 91G40, 62G08, 62M45, 68Q32.

[1] *Sorbonne Université, Laboratoire de Probabilités et Modèles Aléatoires, CNRS UMR 8001, Paris, France*

[2] *Université Paris Cité, Laboratoire de Probabilités et Modèles Aléatoires, CNRS UMR 8001, Paris, France*

[3] *Université Paris-Saclay, Univ Evry, Laboratoire de Mathématiques et Modélisation d'Évry, CNRS UMR 8001, Evry, France*

[4] *Credit Agricole CIB, Quantitative Research GMD/GMT, Paris, France (*`bouazza.saadeddine@gmail.com`*,* **corresponding author***).*

Acknowledgement: We thank an anonymous referee for his useful comments. This research has benefited from the support of the Chair *Capital Markets Tomorrow: Modeling and Computational Issues under the aegis of the Institut Europlace de Finance*, a joint initiative of Laboratoire de Probabilités, Statistique et Modélisation (LPSM) / Université Paris Cité and Crédit Agricole CIB.

*A GPU implementation of our hierarchical simulation and regression learning scheme, as well as single-file python notebook demo for our CVA use case, are available on https://github.com/BouazzaSE/NeuralXVA.

1 Introduction

1.1 Financial Motivation

Greensill defaulted on March 8, 2021, a collapse estimated by British parliamentarians to trigger a cost for UK taxpayers of up to £5bn[1]. Greensill fell short of capital because they lent to Gupta against future invoices which then did not materialize. A projection of Greensill's capital requirements including the tail risk related to Gupta's default would have highlighted a sizable concentrated and unsecured credit risk to a junk-rated counter-party. This example emphasizes the importance of performing proper default risk simulations, as opposed to credit spread simulations simply, as typically done in the industry for the sake of simplicity. Another example (in fact, the motivation for this work) is the path-wise XVA regression approach introduced in Albanese, Crépey, Hoskinson, and Saadeddine (2021), which also requires a hybrid market and credit setup, where the actual defaults of the clients of the bank are simulated.

However, to obtain the required level of accuracy in a simulation setup with defaults, one needs a very large number of simulations—100 times more, say, than in a purely diffusive model, where 100 is in reference to credit spreads that would be of the order of 1%. A factor 100 is not necessarily a sizable amount as far as the simulation of the risk factors is involved. But it also means that the mark-to-market cube of path-wise prices of all trades of the bank becomes 100 times bigger. When applied at the level of a realistic banking portfolio, this becomes prohibitive in terms of both computation time and memory occupancy.

1.2 Contribution of the Paper

To overcome this problem, we introduce an acceleration technique for the computation by simulation and regression of conditional expectations of functions ξ of Markov pairs (X, Y), where Y is an exogenous component, Markov by itself, whose simulation is time-consuming, while the endogenous component X (jointly Markov with Y) is quick to simulate given Y, but also responsible for most of the variance of the simulated payoff. The idea, which we call hierarchical simulation, is then to draw an optimized number of realizations of X conditional on each simulation of Y. For example, in the above-mentioned XVA regression framework, we simulate a few hundred paths of client defaults conditionally on each mark-to-market path. Proceeding in this way, the computational burden of the mark-to-market cube is not amplified by the simulation of the client defaults. We demonstrate, both mathematically and empirically, that the lack of independence of the ensuing simulation setup is not detrimental to the quality of the ensuing learner, i.e. (in the above case) of the regressions of the XVA layers built over the mark-to-market cube and defaults scenarios. In addition, an a posteriori twin Monte Carlo validation technique is introduced to L_2 estimate the error between a targeted conditional expectation and any estimator for the latter.

[1] cf. https://www.theguardian.com/business/2021/apr/28/greensill-collapse-could-cost-uk-taxpayer-up-to-5bn-mps-told, accessed on May 16, 2022.

1.3 Related literature

Supervised learning tasks can be subdivided in two main categories (Murphy, 2012). The first one is classification problems, for which training a neural net allows learning a (conditional) probability that a discrete random variable takes its different possible values (such as 'cat or dog' for an image), by empirical minimization of an entropic criterion [2], based on a number of labeled observations (images and their correct classification, e.g. 'cat or dog'). The second category consists of regression tasks, for which training a neural net allows learning a conditional expectation $\mathbb{E}(V|U)$, by empirical minimization of a least squares criterion based on observations of the pair of random variables (U, V). Our paper falls in the second category, in the special case where the (U, V) are simulated, assuming the data generating process known. For this purpose we rely, as is standard, on the L_2 training error and on out-of-sample validation, represented in our case by companion twin and nested Monte Carlo procedures.

In the last years, such neural net regression-based simulation techniques have rapidly imposed themselves as a worthy player in the field of the numerical methods for PDEs and BSDEs: see e.g. E, Han, and Jentzen (2017) or Huré, Pham, and Warin (2020) (to quote only two). There are good reasons for this evolution, starting with universal approximation theorems, or density results justifying the use of neural network as a versatile parameterization: see (Kidger and Lyons, 2020) for the fixed-width case and (Hornik, 1991; Cybenko, 1989) for the fixed-depth case. Other incentives for using neural networks are the ability to automatically infer a linear regression basis (provided by the trained hidden layers[3]), or the ease of transfer learning (Pan and Yang, 2009; Bozinovski, 2020). However, the control of the error arising from the numerical optimizations for the embedded training tasks (a priori error bounds for (6) in our setup) is a largely open issue: in the nonconvex numerical optimization case typical of neural net training tasks, only idealized versions of stochastic gradient descents with over-parameterized neural nets can be shown to converge to the global minimum of the empirical loss, and only in-sample (Chizat and Bach, 2018; Du, Lee, Li, Wang, and Zhai, 2019); Otherwise, only local minima can be guaranteed, under assumptions such as the ones in Lei, Hu, Li, and Tang (2019). These local minima are often close to global minima (Choromanska, Henaff, Mathieu, Arous, and LeCun, 2015), so that this local vs. global minimization issue is considered by many as a false problem in practice. Still, our aforementioned twin Monte Carlo validation procedure is a useful practical contribution in this regard.

On the XVA side, the use of regression-based Monte Carlo simulations is not a novelty by itself. It was already presented in Cesari, Aquilina, and Charpillon (2010) as a key CVA computational paradigm, intended to avoid nested Monte Carlo. However, from such traditional XVA computations to the neural net regressions of Huge and Savine (2020), the regressions are only used for computing the mark-to-market cube of the prices of all the contracts of the bank with all its clients (or netting sets) at all times of a simulation time-grid, out of which the CVA of the bank at time 0 (and only it)

[2] or Kullback-Leibler divergence, or maximization of a likelihood.

[3] see Figure 3.

is obtained by integration of the so-called expected positive exposure relative to each netting set against the credit curve of the corresponding client, and summation over netting sets. By contrast, in this paper, we aim at learning the CVA as a process, i.e. at every node of a simulation for all risk factors, based on a mark-to-market cube computed by model analytics at the forward simulation stage. Regressions could also be used here, but these would be more standard, hence we ignore them in this paper: regressions for the mark-to-market of derivatives à la Cesari, Aquilina, and Charpillon (2010) are typically multiple parametric regressions in diffusive and low-dimensional setups, as opposed to hybrid diffusive / Markov chain setup and high-dimensional neural net regressions in the path-wise CVA case targeted in this work. Recently, Gnoatto, Reisinger, and Picarelli (2020) deep-hedge and learn the CVA and the FVA, but this is again in a purely diffusive setup, after the default of the bank and its (assumed single) counterparty have been eliminated from the model by the reduction of filtration technique of Crépey and Song (2015). This technique of reduction of filtration is not extendible to the realistic case of a bank involved in transactions with several (in practice, many, e.g. several thousands of) clients, the default times of which enter the ensuing FVA (and KVA) equations in a nonlinear fashion, so that there is then no other choice but simulating these defaults and including them in the training. But this requires special care, which is the topic of this work.

1.4 Outline

In Section 2, we introduce a neural net learning framework for conditional expectations, iterated in time as they appear naturally in dynamic pricing problems, taking into account the dynamics of the problem by means of a backward pricing algorithm. A twin Monte Carlo validation technique is also introduced to estimate the error of the estimator. In Section 3, we identify a variance issue raised by the coexistence of risk factors evolving at different paces in the problem (e.g. market risk versus default indicator processes) and we propose a hierarchical simulation approach to address it. We establish the benefit of this approach mathematically by providing associated generalization bounds. Section 4 illustrates our approach with a CVA numerical case study.

Remark 1. *Although our CVA case study only covers quadratic risk minimization (for benchmarking reasons), the approach and the proofs of this paper are valid for more general loss functions and apply to the learning of any elicitable statistics. In particular, via the Rockafellar and Uryasev (2000) representation of value-at-risk and expected shortfall of a given loss (random variable) in terms of "far out-of-the-money call options" on that loss, our hierarchical simulation approach is also relevant for learning value-at-risk and expected shortfall in hybrid mark-to-market and default simulation setups. Such an approach is even particularly relevant in these cases, where the fact that X is responsible for most of the variance of the payoff is then intrinsic to the far out-of-the-money feature of the corresponding "option".*

2 Neural Regression Setup

A reference probability space, with corresponding probability measure and expectation denoted by $\mathbb{Q}$ and $\mathbb{E}$, is fixed throughout the paper. The state spaces of X and Y are taken as $\mathbb{R}^p$ and $\mathbb{R}^q$, for some positive integers p and q. We identify $\mathbb{R}^{p+q}$ with $\mathbb{R}^p \times \mathbb{R}^q$ and write $\phi(z)$ or $\phi(x, y)$ interchangeably, where z is the concatenation of x and y, for every function ϕ defined over $\mathbb{R}^{p+q}$ or $\mathbb{R}^p \times \mathbb{R}^q$, and for every $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$.

In the (default risk) case of a Markov chain like component X, referred to hereafter as the Markov chain X case (but with transition intensities modulated by Y), we assume, without loss of generality in this case, that X evolves on the vertices $\{0, 1\}^p$ of the unit cube in $\mathbb{R}^p$. We take the problem after discretisation of time (if the latter was continuous in the first place), for a time step set to one year for ease of notation.

We then consider $(X_i)_{0 \le i \le n}$ and $(Y_i)_{0 \le i \le n}$ as discrete-time processes on the time grid. Our goal is to estimate, for every i, conditional expectations of the form

$$\Pi_i = \mathbb{E}[\xi_{i,n} \,|X_i, Y_i\,], \tag{1}$$

where

$$\xi_{i,n} = f_i(X_i, \ldots, X_n, Y_i, \ldots, Y_n). \tag{2}$$

Here f_i is a measurable real function such that $\xi_{i,n}$ is a square-integrable random variable.

Conditional expectations such as (1) can be estimated via linear regression using a finite sample. This is ubiquitous in quantitative finance since the Bermudan Monte Carlo papers of Tsitsiklis and Van Roy (2001) and Longstaff and Schwartz (2001). In order to estimate the conditional expectation in (1), one draws i.i.d. samples $\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota)\}_{\iota \in \mathcal{I}}$ of $(X_i, Y_i, \xi_{i,n})$, where $\mathcal{I}$ is a finite set of indices. Then, given a feature map $\phi : \mathbb{R}^{p+q} \to \mathbb{R}^m$ (for some positive integer m), one linearly regresses $\{\xi_{i,n}^\iota\}_{\iota \in \mathcal{I}}$ against $\{\phi(X_i^\iota, Y_i^\iota)\}_{\iota \in \mathcal{I}}$, solving for

$$\hat{w}_i \in \operatorname*{argmin}_{w_i \in \mathbb{R}^m} \sum_{\iota \in \mathcal{I}} (\xi_{i,n}^\iota - w_i^\top \phi(X_i^\iota, Y_i^\iota))^2. \tag{3}$$

One then uses $\hat{w}_i^\top \phi(X_i, Y_i)$ as an approximation for Π_i.

The above procedure is justified by the characterization, in the square integrable case, of conditional expectations as orthogonal projections, i.e.

$$\mathbb{E}[\xi_{i,n} \,|X_i, Y_i\,] = \varphi_i^\star(X_i, Y_i) \quad \text{a.s.},$$

where, denoting by $\mathcal{B}(E)$ the set of Borel measurable real functions on a metric space E,

$$\varphi_i^\star \in \operatorname*{argmin}_{\varphi_i \in \mathcal{B}(\mathbb{R}^p \times \mathbb{R}^q)} \mathbb{E}[(\xi_{i,n} - \varphi_i(X_i, Y_i))^2]. \tag{4}$$

One recovers the linear regression formulation (3) by approximating the expectation by an empirical mean and restricting the search space to the functions of the form $\mathbb{R}^{p+q} \ni (x, y) \mapsto w_i^\top \phi(x, y)$, where $w_i \in \mathbb{R}^m$.

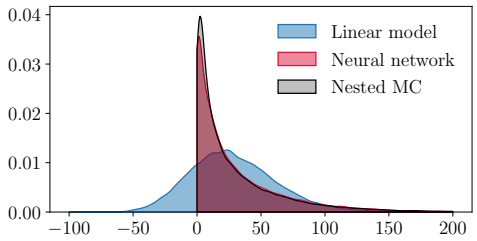

Figure 1: Density plot of the CVA of a vanilla call, at mid-life of the option.

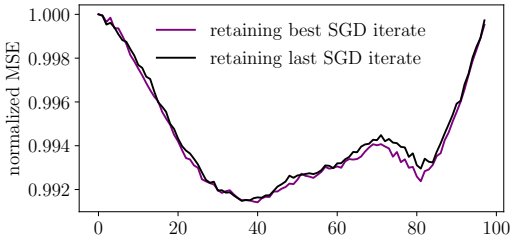

Figure 2: Out-of-sample MSEs against labels $\xi_{i,n}$ at different time-steps divided by the variance of the labels in the context of the CVA case study of Section 4.

2.1 Neural Net Parameterization

Linear regression by means of a priori, explicit factors has a reasonable chance of success when $\varphi_i^\star$ is simple enough and the feature mapping ϕ can be judiciously chosen, usually from expert knowledge. This is however not always the case, e.g. when considering portfolio-wide XVA metrics, which exhibit non-trivial dependencies on the many risk factors being regressed against. It is then impossible to manually devise a satisfactory feature mapping ϕ. Figure 1 shows how a linear regression with the raw risk factors as features fails in learning the (conditional) CVA of an elementary portfolio made of a single call option, while the neural net estimator almost matches with the nested Monte Carlo estimator (see Section 4 for more numerical details). In the Markov chain X case, we face the additional peculiarity of a hybrid regression setting, with discrete and continuous natures of the X and Y model components.

Neural networks (Bengio et al., 2016) propose an alternative way to parameterize and learn the feature map. Let $\mathcal{NN}_{p+q,h,u,\varsigma}$ denote the set of functions of the form

$$\mathbb{R}^{p+q} \ni z \mapsto \zeta(z; W^{[h+1]}, \dots, W^{[1]}, b^{[h+1]}, \dots, b^{[1]}) = \zeta^{[h+1]}(z; W, b),$$

where

$$W^{[h+1]} \in \mathbb{R}^{1\times u}, \dots, W^{[\ell]} \in \mathbb{R}^{u\times u}, \dots, W^{[1]} \in \mathbb{R}^{u\times(p+q)}$$

are weight matrices,

$$b^{[h+1]} \in \mathbb{R}, \dots, b^{[\ell]} \in \mathbb{R}^u, \dots, b^{[1]} \in \mathbb{R}^u$$

are bias offsets, W and b are the respective concatenations of the $W^{[\ell]}$ and of the $b^{[\ell]}$, ς is a scalar nonlinearity applied element-wise and, for every $z \in \mathbb{R}^{p+q}$,

$$\begin{aligned}
&\zeta^{[0]}(z; W, b) = z \\
&\zeta^{[\ell]}(z; W, b) = \varsigma(W^{[\ell]}\zeta^{[\ell-1]}(z; W, b) + b^{[\ell]}), \quad \ell = 1, \dots, h \\
&\zeta^{[h+1]}(z; W, b) = W^{[h+1]}\zeta^{[h]}(z; W, b) + b^{[h+1]}.
\end{aligned}$$

The function $z \mapsto \zeta^{[h]}(z; W, b)$ can be seen as a nonlinear feature mapping from $\mathbb{R}^{p+q}$ to $\mathbb{R}^u$, parameterized by $W^{[h]}, \dots, W^{[1]}, b^{[h]}, \dots, b^{[1]}$ (for a given activation function ς): see Figure 3.

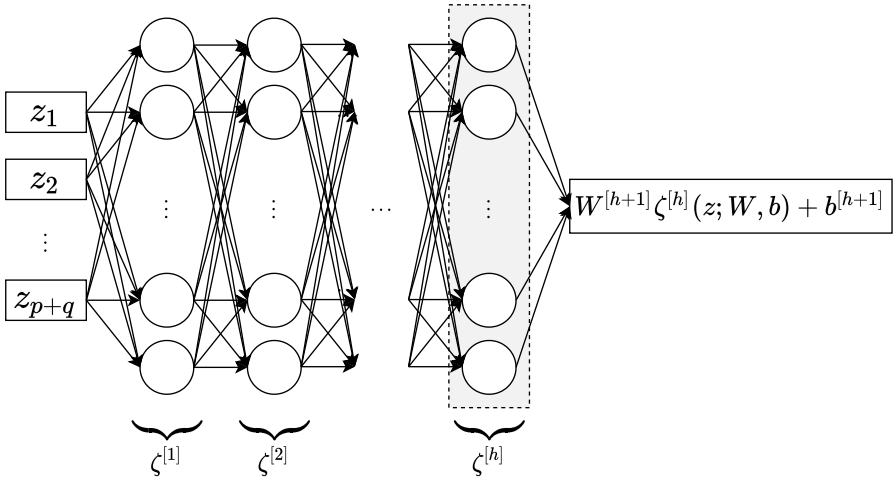

Figure 3: The last hidden layer of our network, $\zeta^{[h]}$, can be seen as a parameterization of an endogenous, implicit linear regression basis.

On top of the set $\mathcal{NN}_{p+q,h,u,\varsigma}$ of real-valued neural networks taking inputs from $\mathbb{R}^{p+q}$, with h hidden layers, u units per hidden layer (and one output neuron), and ς as the activation function, we also define

$$\mathcal{NN}^{+}_{p+q,h,u,\varsigma} = \{\mathbb{R}^{p+q} \ni z \mapsto (f(z))^{+} + \mu, f \in \mathcal{NN}_{p+q,h,u,\varsigma}, \mu \in \mathbb{R}\}. \tag{5}$$

This specification ensures positivity of the output when the additive constant μ is non-negative and is useful for learning positive (e.g. XVA) functions. The additive constant μ is introduced in order to improve the fit of the first moment of the target function.

2.2 Local Training Algorithm

Learning the conditional expectation (1) in a positive neural net search space consists in applying the same empirical risk minimization (3) approximation as in linear regression, using this time $\mathcal{NN}^{+}_{p+q,h,u,\varsigma}$ as the search space, i.e. solving for

$$\hat{\varphi}_i \in \underset{\varphi_i \in \mathcal{NN}^{+}_{p+q,h,u,\varsigma}}{\operatorname{argmin}} \sum_{\iota \in \mathcal{I}} (\xi^{\iota}_{i,n} - \varphi_i(X^{\iota}_i, Y^{\iota}_i))^2. \tag{6}$$

This is achieved by a mini-batch stochastic gradient descent.

For learning a positive output (e.g. an XVA), the addition of a ReLU activation $\cdot^{+}$ at the output layer in (5) can jeopardize the learning as the gradient may vanish at a certain SGD iteration and the parameters are then frozen irrespective of the number of subsequent iterations. Thus, for more stability of the learning procedure, we first perform the first half of SGD steps on the network without the ReLU at the output layer. Then, still without the ReLU, we fine-tune the weights of the output layer by optimizing with respect to those weights only (freezing the weights of the hidden layers), which can be done in closed form in the case of quadratic risk minimization.

Remark 2. *This fine-tuning step is not achievable in closed-form in the case of, for example, quantile regression*[4]*. However, even in this case, the optimization problem is still convex and as such easier to solve numerically.*

Finally, we restore the ReLU at the output layer to proceed with the second half of the SGD iterations.

We also chose to retain the best set of parameters among those explored during the SGD iterations. Figure 2 shows the corresponding improvement in generalization when applied in the context of the CVA case study of Section 4.

The ensuing learning scheme is detailed in Algorithm 0. Note that we presented vanilla SGD iterations only for the sake of simplicity. In practice, accelerated SGD methods like Adam (Kingma and Ba (2014)) are used instead.

Algorithm 0: Baseline learning scheme for training at a given time-step i

---

name : BaseAlg

input : $\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota), \iota \in \mathcal{I}\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E \in \mathbb{N}^\star$, a learning rate $\eta > 0$, initial values for the network parameters W, b and μ

output: Trained parameters W_{best}, b_{best} and μ_{best}

define

$$\mathcal{L}(W, b, \mu, \text{batch}, \text{pos}) = \begin{cases} \dfrac{1}{|\text{batch}|} \displaystyle\sum_{\iota \in \text{batch}} (\zeta^{[h+1]}(X_i^\iota, Y_i^\iota; W, b) + \mu - \xi_{i,n}^\iota)^2 & \text{if pos} = 0 \\ \dfrac{1}{|\text{batch}|} \displaystyle\sum_{\iota \in \text{batch}} ((\zeta^{[h+1]}(X_i^\iota, Y_i^\iota; W, b))^+ + \mu - \xi_{i,n}^\iota)^2 & \text{if pos} = 1 \end{cases}$$

$\mathcal{L}_{\text{best}} \leftarrow \infty$, pos $\leftarrow 0$

for *epoch* $= 1, \ldots, E$ **do** `// loop over epochs`

  for *batch* $\in \mathcal{B}$ **do** `// loop over batches`

    for $\ell = 1, \ldots, h+1$ **do**

      $W^{[\ell]} \leftarrow W^{[\ell]} - \eta \nabla_{W^{[\ell]}} \mathcal{L}(W, b, \mu, \text{batch}, \text{pos})$

      $b^{[\ell]} \leftarrow b^{[\ell]} - \eta \nabla_{b^{[\ell]}} \mathcal{L}(W, b, \mu, \text{batch}, \text{pos})$

    end

    $\mu \leftarrow \mu - \eta \partial_\mu \mathcal{L}(W, b, \mu, \text{batch}, \text{pos})$

  end

  if *epoch* $= \left\lfloor \frac{E}{2} \right\rfloor$ **then** `// tune weights of last layer`

    $(W^{[h+1]}, b^{[h+1]}) \leftarrow \underset{\widetilde{W}^{[h+1]}, \widetilde{b}^{[h+1]}}{\text{argmin}} \mathcal{L}(\{W^{[0]}, \ldots, W^{[h]}, \widetilde{W}^{[h+1]}\}, \{b^{[0]}, \ldots, b^{[h]}, \widetilde{b}^{[h+1]}\}, \mu, obs, 0)$

    pos $\leftarrow 1$

  end

  if $\mathcal{L}(W, b, \mu, \mathcal{I}, 1) < \mathcal{L}_{best}$ **then** `// keep track of best parameters`

    $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}(W, b, \mu, obs, 1)$

    $W_{\text{best}} \leftarrow W$

    $b_{\text{best}} \leftarrow b$

    $\mu_{\text{best}} \leftarrow \mu$

  end

end

---

[4] cf. Remark 1.

2.3 Backward Learning

In the setup of the path-wise pricing problem (1), at each pricing time i, a separate learning problem is solved by Algorithm 0. Since the algorithm returns for each problem a local minimum, it is possible to end up with an approximation of the pricing function $\mathbb{E}[\xi_{i,n}\,|X_i = x, Y_i = y]$ (cf. (1)) with noisy paths (i.e. with respect to time i) if the local minima are not close to each other, even for fixed x and y. Yet, for two consecutive time-steps i and $i+1$, the learning problems are similar. One possible refinement is, after having learned Π_{i+1}, to initialize the parameters of the network at time i with the parameters of the network trained at time $i+1$. This not only smoothes the results across regression times, but also accelerates convergence.

We obtain an algorithm which starts the learnings at the last time step n and, proceeding backward in time until time step 1, reuses each time the previously trained weights and biases as an initialization for the next learning. The ensuing backward learning scheme is detailed in Algorithm 1. This process of reusing knowledge from a different but related learning task can be seen as a form of transfer learning (Pan and Yang, 2009; Bozinovski, 2020).

Algorithm 1: Backward learning scheme

input : $\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota), \iota \in \mathcal{I}, 1 \le i \le n\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E \in \mathbb{N}^\star$, a learning rate $\eta > 0$

output: $\hat{\varphi}_1, \dots, \hat{\varphi}_n$

initialize parameters W_{n+1}, b_{n+1} and μ_{n+1} of the network at terminal time-step n

for $i = n \dots 1$ **do**

  $W_i, b_i, \mu_i \leftarrow \text{BaseAlg}(\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota), \iota \in \mathcal{I}\}, \mathcal{B}, E, \eta, W_{i+1}, b_{i+1}, \mu_{i+1})$

  $\hat{\varphi}_i \leftarrow \{x \mapsto \zeta^{[h+1]}(x, y; W_i, b_i) + \mu_i\}$

end

Remark 3. *A variation on the above would be forward learning. We favor the backward learning scheme because it is the only one that is amenable to more general backward stochastic differential equations, such as the equations for the FVA and the KVA in Crépey (2022). In addition, in these XVA applications, the labels/features corresponding to times i closer to the final maturity n of the portfolio have a lower/higher variance. Hence the training task corresponding to a lower time step i is harder. Proceeding backward in time, we thus solve successively harder and harder problems, but problems which benefit from the knowledge acquired solving the previous ones: a virtuous divide-and-conquer strategy.*

2.4 A Posteriori Twin Monte Carlo Validation Procedure

As part of the validation of our approach, we provide an a posteriori Monte Carlo L^2 error estimation procedure, which can be used for assessing the quality of any estimator of a conditional expectation, without any prior knowledge on the latter (and without heavy nested Monte Carlo). At a given time step i, let $\xi_{i,n}^{(1)}$ and $\xi_{i,n}^{(2)}$ denote two independent

copies of $\xi_{i,n}$ conditional on (X_i, Y_i)[5]. For any Borel function $\varphi : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ such that $\varphi(X_i, Y_i)$ is square integrable (e.g. a neural net estimate of $\mathbb{E}[\xi_{i,n}|X_i, Y_i]$), we have:

$$\mathbb{E}[(\varphi(X_i, Y_i) - \mathbb{E}[\xi_{i,n}|X_i, Y_i])^2] = \mathbb{E}[\varphi(X_i, Y_i)^2 - (\xi_{i,n}^{(1)} + \xi_{i,n}^{(2)})\varphi(X_i, Y_i) + \xi_{i,n}^{(1)}\xi_{i,n}^{(2)}]. \quad (7)$$

The equality stems from the fact that, by conditional independence,

$$(\mathbb{E}[\xi_{i,n}|X_i, Y_i])^2 = \mathbb{E}[\xi_{i,n}^{(1)}|X_i, Y_i]\mathbb{E}[\xi_{i,n}^{(2)}|X_i, Y_i] = \mathbb{E}[\xi_{i,n}^{(1)}\xi_{i,n}^{(2)}|X_i, Y_i], \quad (8)$$

followed by an application of the tower rule. Thus, one can approximate the L^2 error of any estimator for the conditional expectation, without any knowledge on the latter, using only two inner paths. This can be used as a very fast validation procedure and as a safeguard in a production environment before using the learned values. A slower but more complete nested Monte Carlo approach is then only needed periodically, e.g. after significant changes in the risk factor models, or to perform more elaborate checks (e.g. tail behavior).

3 Hierarchical Simulation and its Analysis

Learning tasks involving defaults such as (6) may be challenging, even with optimized training schemes. As should always be first scrutiny with machine learning, a difficulty can come from the data, i.e. from the simulation part in our case. Specifically, a large variance of the estimated population loss function can jeopardize the learning approach, which we address in what follows by a suitable hierarchical simulation approach.

3.1 Identification of the Variance Contributions Using Automatic Relevance Determination

In this part we show how to hierarchize the variance impact of explanatory variables using automatic relevance determination (ARD). As detailed in Rasmussen and Williams (2006, Sections 5.1, 5.4.3, 6.6, and 8.3.7), ARD is a Bayesian procedure for feature selection and consists in estimating the relevance of the features by maximizing a marginal likelihood. In our case, we apply a Gaussian process regression based ARD to quantify empirically the impact of the variances of X and Y on that of ξ.

Toward this aim, we treat the vector of the parameters of the data generating process (DGP, e.g. a financial model), denoted by ν, as a latent variable endowed with some instrumental distribution.

Example 1. *In the CVA setup of Sections 4.1-B, ν is the collection of the drift and diffusion parameters of the SDEs* (45)-(46) *for all e and c.* ∎

[5]The conditional independence means that for any Borel bounded functions ϕ_1 and ϕ_2, we have $\mathbb{E}[\phi_1(\xi_{i,n}^{(1)})\phi_2(\xi_{i,n}^{(2)})|X_i, Y_i] = \mathbb{E}[\phi_1(\xi_{i,n}^{(1)})|X_i, Y_i]\mathbb{E}[\phi_2(\xi_{i,n}^{(2)})|X_i, Y_i]$.

Given ν, we sample the following time-averages of the variances of $X_1, \ldots, X_n$, $Y_1, \ldots, Y_n$ and $\xi_{1,n}, \ldots, \xi_{n,n}$:

$$\begin{aligned} V(X|\nu) =& \frac{1}{n+1} \sum_{i=0}^{n} \widehat{\mathbb{E}}^{\nu}[(X_i - \widehat{\mathbb{E}}^{\nu}[X_i])^2] \\ V(Y|\nu) =& \frac{1}{n+1} \sum_{i=0}^{n} \widehat{\mathbb{E}}^{\nu}[(Y_i - \widehat{\mathbb{E}}^{\nu}[Y_i])^2] \\ V(\xi|\nu) =& \frac{1}{n+1} \sum_{i=0}^{n} \widehat{\mathbb{E}}^{\nu}[(\xi_{i,n} - \widehat{\mathbb{E}}^{\nu}[\xi_{i,n}])^2], \end{aligned}$$

meant componentwise in the vector cases of X and Y, where $\widehat{\mathbb{E}}^{\nu}$ is an empirical average over paths sampled for a given realization ν of the DGP parameters. Then, based on a finite sample of ν and on the corresponding realizations of the triple $(V(X|\nu), V(Y|\nu), V(\xi|\nu))$, we perform a Gaussian process regression (Rasmussen and Williams, 2006) of $V(\xi|\nu)$ against $V(X|\nu)$ and $V(Y|\nu)$. In this procedure we use an anisotropic kernel $\exp(-\sum_{j=1}^{p} \frac{(v_j - v'_j)^2}{2\lambda_{x,j}^2} - \sum_{j=1}^{q} \frac{(w_j - w'_j)^2}{2\lambda_{y,j}^2})$, where the hyperparameters $\lambda_{x,1}, \ldots, \lambda_{x,p}$ and $\lambda_{y,1}, \ldots, \lambda_{y,q}$ are characteristic length-scales for the corresponding components of $V(X|\nu)$ and $V(Y|\nu)$. Maximizing the marginal likelihood on the dataset allows to recover those length-scales and these can then be interpreted as relevance estimates for the different input variables. The higher the inverse length-scale gets, the more the corresponding variable influences the output (payoff variance, in our case).

This procedure is then itself randomized, i.e. run multiple times on the restricted datasets corresponding to different sub-samplings of ν. This provides a distribution of the fitted hyper-parameters λ in the above, while being also less prone to over-fitting and local minima issues. A similar analysis was used in Bergstra and Bengio (2012) to study the relevance of different neural network hyper-parameters with respect to the validation loss.

Example 2. *Figure 4 reveals the dominance of the impact of the variance of X on that of ξ in the context of the CVA case study of Section 4 (here for a single-client portfolio of the bank), where Y is the vector of mark-to-market risk factor processes and X is the default indicator process of the client.* ∎

3.2 Learning on Hierarchically Simulated Paths

If X contributes more to the variance of ξ than Y, then, in order to deal with the resulting variance issue regarding the associated simulation/learning scheme, an idea is to simulate more realizations of X than Y, even if this means giving up the independence of the simulation setup. More precisely, we simulate M i.i.d paths $Y^1, \ldots, Y^M$ of Y and, for every $k \in \{1, \ldots, M\}$ and $i \in \{1, \ldots n\}$, we simulate N i.i.d realizations $X_i^{k,1}, \ldots, X_i^{k,N}$ of X_i conditional on Y^k. For every i, this yields a sample

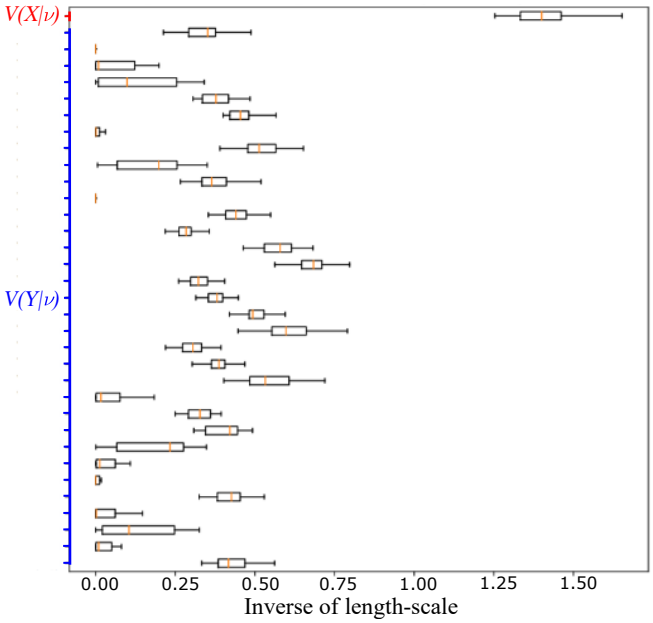

Figure 4: CVA case with a single client in Section 4.2: Box-plot of the inverse length-scales obtained by randomized Gaussian process regressions of the conditional variances of the cash flows ξ against the conditional variances of the risk factors X (default indicator process of the client) and Y, where conditional here is in reference to the parameters of the financial model (45)-(46) for all e and c, treated as a random vector with a postulated distribution.

$(X_i^{k,l}, Y_i^k, \xi_{i,n}^{k,l})$, $k \in \{1,\dots,M\}, l \in \{1,\dots,N\}$ of $(X_i, Y_i, \xi_{i,n})$ of size MN, where, within each block k, independence between the $X^{k,l}$ only holds conditionally on Y^k.

Algorithm 1 is then run on the resulting hierarchically simulated dataset by taking $\mathcal{I} = \{1,\dots,M\} \times \{1,\dots,N\}$, with $Y^{k,l} := Y^k$ for all l. For implementation efficiency reasons pertaining to memory contiguity, the set of indices of the $\imath$-th batch, with $1 \le \imath \le |\mathcal{B}|$, is chosen to be $\{(k,l) \in \{1,\dots,M\} \times \{1,\dots,N\} : (\imath-1)\frac{|\mathcal{I}|}{|\mathcal{B}|} \le (l-1)N + (k-1) + 1 < \imath\frac{|\mathcal{I}|}{|\mathcal{B}|}\}$.

Hierarchical simulation in the above sense can be thought of as a form of data augmentation procedure (Shorten and Khoshgoftaar, 2019), but in a simulation setup where one knows how to generate the data perfectly. The main question is then to which extent one should augment the data, i.e. the choice of the hierarchical simulation parameters M and N, which is the focus of the sequel of this section.

Remark 4. *Hierarchical simulation is different in nature from importance sampling that favors particular events, e.g., in a credit risk setup, default versus survival (see e.g. Carmona and Crépey (2010)). In an XVA setup, some metrics, like the CVA, need default events for being properly estimated, whereas others, like the FVA, require survival events. Hence what one needs is richness regarding both default and survival events, which is what hierarchical simulation provides.*

It is also unrelated to nested Monte Carlo as per Gordy and Juneja (2010) or (in an XVA setup) Abbas-Turki, Diallo, and Crépey (2018), which optimizes the complexity of the simulation with respect to the number of trajectories on each Monte Carlo layer.

In these contributions, the inner conditional trajectories are not assumed to be much less complex to simulate than the outer one. Thus, in contrast to the asymptotic results obtained for all layers in (Gordy and Juneja, 2010; Abbas-Turki, Diallo, and Crépey, 2018), Heuristic 2 separates the asymptotic strategy for N from the non asymptotic strategy proposed for M.

3.3 Choosing the Hierarchical Simulation Factor

Assume that simulating Y_i costs P times more than simulating X_i given a path $\{Y_j\}_{j\leq i}$ in terms of computation time. The hierarchical simulation factor N can be chosen so as to minimize the variance ($\mathbb{V}\mathrm{ar}$) of the loss $\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} g_i(\theta, X_i^{k,l},\ldots,X_n^{k,l},Y_i^k,\ldots,Y_n^k)$ with respect to N, under a budget constraint $M(N+P)=B$, where g_i is the point-wise loss of our learning task at time-step i, e.g. $g_i(\theta,X_i,\ldots,X_n,Y_i,\ldots,Y_n)=(\xi_{i,n}-\varphi_i^\theta(X_i,Y_i))^2$, where φ_i^θ is a neural net (element of $\mathcal{NN}^+_{p+q,h,u,\varsigma}$) with parameters collectively denoted by θ (cf. (6)).

For ease of notation in this and the next part, we write $g_i(\theta,X^{k,l},Y^k)$ and $g_i(\theta,X,Y)$ instead of $g_i(\theta,X_i^{k,l},\ldots,X_n^{k,l},Y_i^k,\ldots,Y_n^k)$ and $g_i(\theta,X_i,\ldots,X_n,Y_i,\ldots,Y_n)$ (it is then implied that X and Y play formally the role of vectors containing their path from time-step i up to n).

Proposition 1. *The hierarchical simulation factor that minimizes the variance of the loss $\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} g_i(\theta,X^{k,l},Y^k)$ with respect to N, subject to the budget constraint $M(N+P)=B$, is*

$$N_i^\theta=\sqrt{\frac{Q_i^\theta P}{R_i^\theta}}, \tag{9}$$

where

$$R_i^\theta=\mathbb{C}\mathrm{ov}\big(g_i(\theta,X^{1,1},Y^1),g_i(\theta,X^{1,2},Y^1)\big)=\mathbb{V}\mathrm{ar}\big(\mathbb{E}(g_i(\theta,X^{1,1},Y^1)|Y^1)\big)$$
$$Q_i^\theta=\mathbb{E}\big(\mathbb{V}\mathrm{ar}(g_i(\theta,X^{1,1},Y^1)|Y^1)\big)=\mathbb{V}\mathrm{ar}(g_i(\theta,X^{1,1},Y^1))-R_i^\theta.$$

Proof. After rearranging terms, one can show that

$$\mathbb{V}\mathrm{ar}\Big(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} g_i(\theta,X^{k,l},Y^k)\Big)=\frac{R_i^\theta}{B}\Big(\frac{1}{N}\big(N-\sqrt{\frac{Q_i^\theta P}{R_i^\theta}}\big)^2+\big(\sqrt{\frac{Q_i^\theta}{R_i^\theta}}+\sqrt{P}\big)^2\Big),$$

where

$$Q_i^\theta=\mathbb{E}[(g_i(\theta,X^{1,1},Y^1))^2]-\mathbb{E}[g_i(\theta,X^{1,1},Y^1)g_i(\theta,X^{1,2},Y^1)]$$
$$R_i^\theta=\mathbb{E}[g_i(\theta,X^{1,1},Y^1)g(\theta,X^{1,2},Y^1)]-(\mathbb{E}[g_i(\theta,X^{1,1},Y^1)])^2.\ \blacksquare$$

The quotient

$$\frac{Q_i^\theta}{R_i^\theta}=\frac{\mathbb{E}\big(\mathbb{V}\mathrm{ar}(g_i(\theta,X^{1,1},Y^1)|Y^1)\big)}{\mathbb{V}\mathrm{ar}\big(\mathbb{E}(g_i(\theta,X^{1,1},Y^1)|Y^1)\big)}$$

in (9) measures the relative contributions of Y and X to the variance of the loss estimator (note that $Q_i^\theta + R_i^\theta = \mathbb{V}\mathrm{ar}(g_i(\theta, X^{1,1}, Y^1))$, by the total variance formula). To estimate the values of Q_i^θ and of R_i^θ therein, one only needs to simulate $(X^{1,1}, X^{1,2}, Y^1)$, i.e., with respect to the bare simulation of (X, Y), one extra simulation of X conditional on each realization of Y.

As a fixed value of N has to be chosen throughout all the simulation and training task, for the above result to be of practical use, N_i^θ has to be reasonably stable with respect to both pricing time steps i and SGD iterations (the transfer learning scheme of Section 2.3 is advantageous in this respect in that it stabilizes the learning). If so, it leads to the following:

Heuristic 1. *Choose for N the average of the values N_i^θ obtained during the SGD iterations and the time steps. Make for M the corresponding choice deduced from the budget constraint, i.e. $M = \frac{B}{N+P}$.*

Note that N depends only on P, and M on P and B. If P is not analytically known, it can be deduced from simulation times of experiments corresponding to the same M but different N. Namely, let B and B' the budgets corresponding to configurations (M, N) and (M, N'). We have

$$\frac{B}{B'} = \frac{P+N}{P+N'}. \tag{10}$$

One can deduce P by identifying the ratio in (10) to that of the execution times of (M, N) and (M, N'). For doing so, it is preferable to choose M large enough to avoid time measurement noise that may be due to caching or parallelization of the simulations.

3.4 Statistical Convergence Analysis

In this part we completely omit the index i from the notation. For every possible parameterization $\theta \in \Theta \subset \mathbb{R}^d$ of our neural network (with d parameters), define

$$\begin{aligned} G(\theta) &= \mathbb{E}[g(\theta, X, Y)] \\ \hat{G}_{M,N}(\theta) &= \frac{1}{MN} \sum_{k=1}^{M} \sum_{l=1}^{N} g(\theta, X^{k,l}, Y^k) \end{aligned}$$

and, for all $\epsilon > 0$ and non-empty subsets E of Θ:

$$\begin{aligned} S^\epsilon(E) &= \{\theta \in E : G(\theta) \leq \min_E G + \epsilon\} \\ \hat{S}^\epsilon_{M,N}(E) &= \{\theta \in E : \hat{G}_{M,N}(\theta) \leq \min_E \hat{G}_{M,N}. + \epsilon\}. \end{aligned} \tag{11}$$

Let $\mathcal{X}$ and $\mathcal{Y}$ be the state spaces of X and Y. For all $\theta, \theta' \in \Theta$ and $t \in \mathbb{R}, y \in \mathcal{Y}$, denote:

$$\begin{aligned} \Gamma(\theta, \theta', X, Y) &= g(\theta', X, Y) - g(\theta, X, Y) \\ \mathcal{M}(\theta, \theta', t, y) &= \mathbb{E}[\exp(t\Gamma(\theta, \theta', X, Y))|Y = y]. \end{aligned} \tag{12}$$

We are interested in the event

$$\{\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)\} = \bigcup_{\theta \in E \setminus S^{\epsilon}(E)} \bigcap_{\theta' \in E} \{\hat{G}_{M,N}(\theta) \leq \hat{G}_{M,N}(\theta') + \delta\}, \tag{13}$$

where E is a non-empty subset of Θ and $\epsilon, \delta > 0$. This is the event that a close to minimum of the finite-sample problem is far from being a minimum of the population mean minimization problem. Theorem 1 provides a bound on the probability of this event when E is finite.

Theorem 1. *Let E be a finite and non-empty subset of Θ and let $0 < \delta < \epsilon$. Assume that $E \setminus S^{\epsilon}(E) \neq \emptyset$ and that there exist $b_1, b_2 > 0$ such that for every $t \in \mathbb{R}$ and $\theta, \theta' \in E$:*

$$\mathcal{M}(\theta, \theta', t, Y) \leq \exp(\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y]t + \frac{b_1^2 t^2}{2}) \quad a.s. \tag{14}$$

$$\mathbb{E}[\exp(t\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y])] \leq \exp(\mathbb{E}[\Gamma(\theta, \theta', X, Y)]t + \frac{b_2^2 t^2}{2}). \tag{15}$$

Then

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)) < |E| \exp(\frac{-M(\epsilon - \delta)^2}{2(b_1^2/N + b_2^2)}). \tag{16}$$

Proof. See Section A.1. ∎

Theorem 2 yields a similar bound valid for a possibly infinite parameter space, under additional assumptions of compactness and convexity of this space and Lipschitz continuity of the point-wise loss function. For brevity we write $S^{\epsilon}(\Theta) = S^{\epsilon}$, $\hat{S}^{\delta}_{M,N}(\Theta) = \hat{S}^{\delta}_{M,N}$.

Theorem 2. *Assume that Θ is a compact and convex and let $0 < \delta < \epsilon$. Let $D = \sup_{\theta, \theta' \in \Theta} \|\theta - \theta'\|$ and assume that there exists a mapping $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{\star}_{+}$ such that $\mathbb{E}[\exp(tL(X, Y))] < \infty$ for all t in some neighbourhood of 0 and for all $\theta, \theta' \in \Theta$:*

$$|g(\theta, X, Y) - g(\theta', X, Y)| \leq L(X, Y)\|\theta - \theta'\| \quad a.s. \tag{17}$$

Let $\bar{L} = \mathbb{E}[L(X, Y)]$ and assume that there exist $\ell_1, \ell_2 > 0$ such that:

$$|L(X, Y) - \mathbb{E}[L(X, Y)|Y]| \leq \ell_1 \quad a.s. \tag{18}$$

$$|\mathbb{E}[L(X, Y)|Y] - \bar{L}| \leq \ell_2 \quad a.s. \tag{19}$$

and that there exist $b_1, b_2 > 0$ such that for every $t \in \mathbb{R}$ and $\theta, \theta' \in \Theta$:

$$\mathcal{M}(\theta, \theta', t, Y) \leq \exp(\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y]t + \tfrac{b_1^2 t^2}{2}) \quad a.s. \tag{20}$$

$$\mathbb{E}[\exp(t\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y])] \leq \exp(\mathbb{E}[\Gamma(\theta, \theta', X, Y)]t + \tfrac{b_2^2 t^2}{2}). \tag{21}$$

Then

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}) \leq \inf_{L' > \bar{L}} \{\big((\frac{8L'D}{\epsilon - \delta} + 1)^d + 1\big) \exp(\frac{-M(\epsilon - \delta)^2}{8(b_1^2/N + b_2^2)}) + \exp(\frac{-M(L' - \bar{L})^2}{2(\ell_1^2/N + \ell_2^2)})\}. \tag{22}$$

Proof. See Section A.2. ∎

The Lipschitz assumptions of Theorem 2 are reasonable in our case since our neural network is Lipschitz with respect to its parameters, and its composition with the loss function remains Lipschitz if we assume that the parameters are bounded. In particular, these Lipschitz assumptions are satisfied in our learnings if we assume that *(i)* the processes X and Y are bounded (natively or after numerical truncation), *(ii)* the payoff function f (cf. (2)), which is embedded in the loss function g, is Lipschitz continuous or bounded, and *(iii)* Lipschitz continuous activation functions are used in the neural networks.

The following result can help in selecting M for reaching a target confidence level $(1-\alpha)$.

Corollary 1. *Let $0 < \alpha < 1$ and assume the conditions of Theorem 2. Choose any weight $u \in (0,1)$ and $L' > \bar{L}$. Then, for*

$$M = \max\{\frac{8(b_1^2/N + b_2^2)}{(\epsilon-\delta)^2}\log(\frac{1}{u\alpha}\big((\frac{8L'D}{\epsilon-\delta}+1)^d+1\big)), \frac{8(\ell_1^2/N+\ell_2^2)}{(L'-\bar{L})^2}\log(\frac{1}{(1-u)\alpha})\}, \quad (23)$$

we have

$$\hat{S}^{\delta}_{M,N} \subset S^{\epsilon} \text{ with probability at least } (1-\alpha). \quad (24)$$

Proof. By Theorem 2, for (24) to hold, it suffices that

$$\big((\frac{8L'D}{\epsilon-\delta}+1)^d+1\big)\exp(\frac{-M(\epsilon-\delta)^2}{8(\frac{b_1^2}{N}+b_2^2)}) < u\alpha \text{ and}$$

$$\exp(\frac{-M(L'-\bar{L})^2}{2(\frac{\ell_1^2}{N}+\ell_2^2)}) < (1-u)\alpha,$$

which is verified by choosing M as (23). ∎

As the formula (23) for M is decreasing in N, no matter how large N is, M has to be greater than the limit of (23) as $N \to \infty$, which provides a lower bound for M. This is natural as we do not expect to get an efficient sampling and good generalization just by increasing the number of realizations of X only.

Corollary 1 suggests the following:

Heuristic 2. *In order to satisfy a constraint $\mathbb{Q}(\hat{S}^{\delta}_{M,N} \subset S^{\epsilon}) \geq 1-\alpha$ (instead of a target budget B in Heuristic 1), choose N as in Heuristic 1 (since it is independent of the budget value), then deduce M from the formula* (23)*.*

In the data augmentation mindset framed before Remark 4, one could then set the size M of the market data Y as a function of the hierarchical simulation factor N and of the confidence level $(1-\alpha)$. In a context where collecting the market data Y is expensive, Heuristic 2 would thus allow the user to benefit from the augmentation factor N through a reduction of the size M of the dataset for Y. However, making Heuristic 2 really practical would require to estimate the parameters b_1, b_2, ℓ_1, ℓ_2 and $\bar{L}$ in Corollary 1.

4 CVA Case Study

We illustrate our approach numerically by a CVA case study. In this context the probability measure $\mathbb{Q}$ represents a risk-neutral measure chosen by the market, to which a model of financial risk factors is calibrated.

4.1 Market and Credit Model

We consider a bank trading derivative contracts in different economies e with various clients c. The currency corresponding to the economy labeled by 0 is taken as the reference currency. Let there be given the short rate process $r^{\langle e \rangle}$ in each economy e, as well as the exchange rate process $\chi^{\langle e \rangle}$ from the currency of each economy $e \neq 0$ to the reference currency. Each client c of the bank has a stochastic default intensity process $\gamma^{\langle c \rangle}$ and a default-time $\tau^{\langle c \rangle}$. For notational convenience we also define $\chi^{\langle 0 \rangle} = 1$ and we denote by $\gamma^{\langle 0 \rangle}$ the default intensity of the bank itself. We consider an Euler-Marayama time-discretization of the model in Section B. We use the same notation for the continuous-time processes and their discrete-time approximations (with time-step equal to 1 year to alleviate the notation).

Remark 5. *In practice, the time-discretizations are stepping through a refined simulation time grid. This simulation grid is also used when integrating numerically some of the above diffusions, e.g. the default intensities in* (47)*, or for defining risk-neutral discount factors β_i associated with the reference currency by approximating $(-\ln \beta_i)$ using numerical integration*[6] *of $r^{\langle 0 \rangle}$ on $[0, i]$. Learning, pricing and checking for default events, instead, are only done at the coarser pricing time steps. Hence, although we step through the fine time grid in our discretized diffusions, we only need to store the values of the processes at the pricing time steps.*

We define X as the collection of all the default indicator processes of the clients c and Y as the collection of all the short interest rate, FX, and default intensity processes r, χ and γ (except for the instrumental $\chi^{\langle 0 \rangle} = 1$), endowed with the filtration generated by the innovation in the model, i.e. the collection of all the Gaussian and exponential variables involved at the increasing time steps $i \in \{1, \ldots, n\}$. Note that both Y (by itself) and (X, Y) (jointly) are Markov processes with respect to this filtration.

4.2 Learning the CVA

We denote by $\mathrm{MtM}_i^{\langle c \rangle}$ the mark-to-market at time i, from the point of view of the bank and in units of the reference currency, of all the contracts with the client c. By mark-to-market we mean trade additive counterparty-risk-free valuation, i.e. the risk-neutral conditional expectation of the future contractually promised cash flows, expressed in units of the reference currency and discounted at the risk-free rate $r^{\langle 0 \rangle}$. We restrict ourselves to interest-rate derivatives for which mark-to-market valuation at i is a function

[6] It is also possible to jointly simulate exactly $r^{\langle 0 \rangle}$ and its integral without the need for numerical integration, see for example Glasserman (2004).

of Y_i, by the nature of the cash-flows and the Markov property of Y[7]. The CVA of the bank then corresponds to the risk-neutral conditional expectation of its future risk-free discounted client default losses. Namely, the CVA of the bank at the time step i is given by[8]

$$\mathrm{CVA}_i = \sum_c \mathrm{CVA}_i^{\langle c\rangle} \mathbb{1}_{\{i<\tau^{\langle c\rangle}\}}, \tag{25}$$

for a CVA of the client c such that

$$\mathrm{CVA}_i^{\langle c\rangle} = \mathbb{E}[\sum_{j=i}^{n-1} \beta_i^{-1}\beta_{j+1}(\mathrm{MtM}_{j+1}^{\langle c\rangle})^+ \mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1}\Big| X_i, Y_i]. \tag{26}$$

One can rewrite the CVA of the bank using a single expectation conditional on the default states of all clients, as

$$\mathrm{CVA}_i = \mathbb{E}\Big[\sum_c \sum_{j=i}^{n} \beta_i^{-1}\beta_{j+1}(\mathrm{MtM}_{j+1}^{\langle c\rangle})^+ \mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1}\Big| X_i, Y_i\Big]. \tag{27}$$

Hence $\mathrm{CVA}_i = \varphi_i^\star(X_i, Y_i)$, where

$$\varphi_i^\star \in \operatorname*{argmin}_{\varphi\in\mathcal{B}(\mathbb{R}^{p+q})} \mathbb{E}\big[(\sum_c \sum_{j=i}^{n} \beta_i^{-1}\beta_{j+1}(\mathrm{MtM}_{j+1}^{\langle c\rangle})^+ \mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1} - \varphi(X_i, Y_i))^2\big]. \tag{28}$$

We also mention the following default intensity-based formula for the CVA of the bank (cf. Albanese et al. (2021, Eq. (60))):

$$\widetilde{\mathrm{CVA}}_i = \mathbb{E}\Big[\sum_c \sum_{j=i}^{n-1} \beta_i^{-1}\beta_j(\mathrm{MtM}_j^{\langle c\rangle})^+ \gamma_j^{\langle c\rangle} \exp\big(-\sum_{s=i}^{j-1}\gamma_s^{\langle c\rangle}\big)\mathbb{1}_{\{i<\tau^{\langle c\rangle}\}}\Big| X_i, Y_i\Big], \tag{29}$$

which converges to the same continuous-time limit as CVA_i when the time discretisation step[9] goes to zero. Hence $\widetilde{\mathrm{CVA}}_i = \tilde\varphi_i(X_i, Y_i)$, where (cf. (4))

$$\tilde\varphi_i \in \operatorname*{argmin}_{\varphi\in\mathcal{B}(\mathbb{R}^{p+q})} \mathbb{E}\Big[\sum_c \sum_{j=i}^{n-1} \beta_i^{-1}\beta_{j+1}(\mathrm{MtM}_i^{\langle c\rangle})^+ \gamma_j^{\langle c\rangle} \exp(-\sum_{s=i}^{j-1}\gamma_s^{\langle c\rangle})\mathbb{1}_{\{i<\tau^{\langle c\rangle}\}} - \varphi(X_i, Y_i)\Big]^2. \tag{30}$$

We reiterate that Algorithm 1 with hierarchical simulation of (X, Y) is generically applicable to all the XVA metrics. The focus on the CVA in our case study is for benchmarking purposes. Were it for the CVA only, the regression learning scheme with minimal variance is obviously the one based on (29). Also note that learning the CVA client by client, exploiting the linearity of the conditional expectation for this purpose, would lead to as many regressions as there are clients of the bank, which for realistic banking portfolios would be extremely inefficient.

[7] see Remark 7.

[8] Assuming that the netting set for a given client is the whole set of transactions with this client.

[9] Conventionally set to one in this paper.

On top of the learning schemes (30) and (28) associated with the formulations (29) and (27), another computational alternative in each case is nested Monte Carlo as detailed in Abbas-Turki, Diallo, and Crépey (2018). This variety of approaches is useful for benchmarking purposes. Specifically, we implemented the learning procedure of Algorithm 1 in PyTorch with custom CUDA kernels for label generation during the backward iterations, the way detailed in Section C (and in the accompanying github repository). In addition we implemented an optimized CUDA benchmark involving nested simulations, using the intensity-based formulation (29) for the inner CVA computations. For the nested Monte Carlo, in consideration of the square-root rule recalled in Abbas-Turki, Diallo, and Crépey (2018, Section 3.3), we used 128 inner paths. The nested Monte Carlo CVA is only computed at few pricing times due to the heavy calculation.

4.3 Preliminary Learning Results Based on IID Data

In the following experiments, we assume that the bank is trading derivatives in 10 economies with 8 clients. Implementing the discretized market and default model, we get a total of 10 interest rates, 9 cross-currency rates, and 8 default intensities. This yields 27 diffusive market risk factors and 8 default indicator processes. For time-stepping, we use $n = 100$ pricing time steps and 25 simulation sub-steps per pricing time step (see Remark 5). We consider a portfolio of 500 interest rate swaps with random characteristics (notional, currency and counterparty), the $\mathrm{MtM}^{\langle c\rangle}$ are thus analytic. All swaps are priced at par at inception. For all the runs of the simulations in this section, whether they be for training or testing, we use $M = 16384$ paths for the market risk factors Y.

The comparison between the two panels of Figure 5 reveals a difficulty with the neural net learning approach of Algorithm 1 applied to the defaults-based formulation (27) on the basis of i.i.d. simulated data. In this case, represented by the left panel in Figure 5, the network only learns a rather crude and noisy approximation of the CVA conditional to each training time: it is only on the mean that the learned CVA agrees with the nested Monte Carlo estimator; on the tails it largely fails. As visible from the right panel, the CVA learned using the intensity-based formulation, instead, yields satisfactory results on a wide range of quantiles of the targeted distribution.

4.4 Learning Results Based on Hierarchically Simulated Data

In order to improve the learning (28) of the defaults-based CVA (27), we apply to it the hierarchical simulation technique of Section 3. Let $(r^1, \chi^1, \gamma^1), \ldots, (r^M, \chi^M, \gamma^M)$, be i.i.d sample paths of the triple of processes (r, χ, γ). Let $\{\epsilon^{k,l}, 1 \le k \le M, 1 \le l \le N\}$ be i.i.d samples of ϵ, the vector defined by the right-hand side in (47) where c ranges over clients. Then we can define MN samples of the vector of the default indicator processes of the clients at every pricing time i based on (47). Figure 6 illustrates the ensuing simulation scheme for the default indicator of a generic client of the bank, with sampled default times $\tau^{k,l}$.

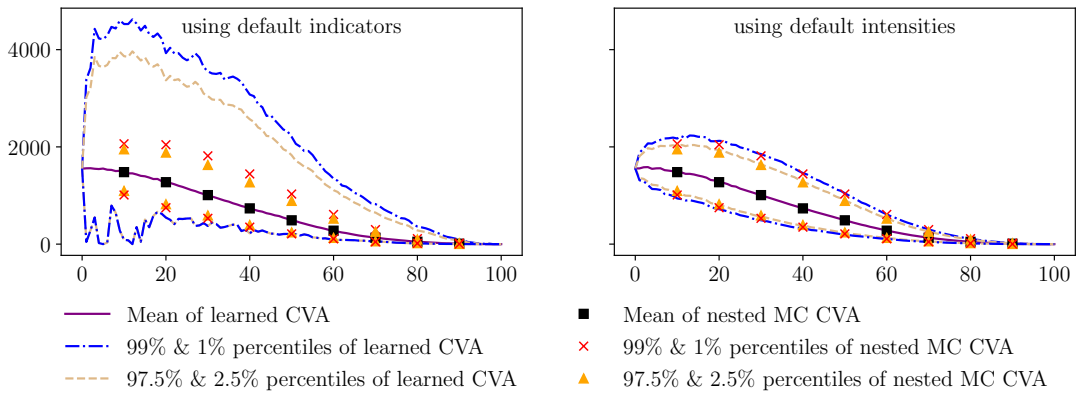

Figure 5: CVA learned using default indicators versus using default intensities (x axis pricing times, y axis CVA levels). Statistics computed using out-of-sample paths.

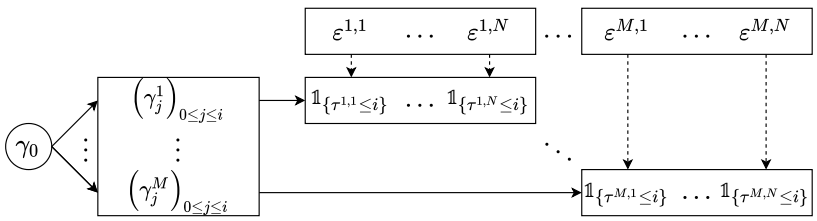

Figure 6: Default simulation scheme

We then learn the CVA process at different time steps for the whole portfolio at once based on (27), trying different combinations of the number of market paths M and of the hierarchical simulation factor N. Figures 7 and 8 show the relative RMSE of the trained neural network against the nested Monte Carlo benchmark[10], the simulation and training times on the GPU and the host RAM usage, as functions of the number of diffusion paths M and of the hierarchical simulation factor N. For the execution times in Figure 8, the runs were done on a server with an Intel Xeon Gold 6248 CPU and 4 Nvidia Tesla V100 GPUs (out of which we used only one). For performance comparison reasons, we use for all (M, N) configurations the same number of epochs $E = 8$ and number of batches $|\mathcal{B}| = 32$, which yields a total of 256 stochastic gradient descent steps during any training task. From Figures 7 and 8, we already see some configurations $(\frac{1}{2}M, N)$ being better than $(M, \frac{1}{2}N)$, as they achieve a similar accuracy with less memory footprint and computational time. For example, $(32768, 1024)$ is better than $(65536, 512)$, given that the former achieves a similar RMSE of 0.07 but is 30% faster to simulate and price, while also occupying 23% less CPU memory. In addition, to the credit of the twin Monte Carlo validation procedure of Section 2.4, the comparison between Figures 7 and 9 shows that the error estimates provided by the latter are very much in line with the ones[11] provided by a much heavier nested Monte Carlo procedure (which would become unfeasible on more complex problems).

The dominance of the impact of the variance of X on that of ξ has been demonstrated in Figure 4. Figure 10 shows the $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$ (cf. (9)) obtained in the base case $N = 1$. The values fluctuate quite significantly both in pricing time steps i and SGD iterations. For the purpose of applying Heuristic 1 (which can be applied for any N but the performance of which needs to be assessed carefully), we retain a rough average order of magnitude of the order of a few tens. To obtain from the $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$ the N_i^θ in (9), one needs to multiply them by $\sqrt{P}$ (e.g. if a market simulation is 100 times slower than an ensuing default simulation, then the factors displayed in Figure 10 must be multiplied by 10). Solving the equation (10) for P on the basis of the columns $M = 65536$ in Figures 7-8 yields $P \approx 497$. So the numbers in Figure 10 need to be multiplied by $\sqrt{497} \approx 22.3$ to get the optimal N as per Heuristic 1. In view of this, we expect an optimal hierarchical simulation factor N of the order of a few hundreds.

Path-wise CVA estimators learned for $N = 1, 32, 64, 128, 512$ are shown in Figure 11, which are to be compared to the right plot in Figure 5 obtained when learning the CVA relying on the intensity-based formula (29). In line with the above expectations, one needs $N = 512$ in order to have a close enough match between the 1, 2.5, 97.5 and 99-th percentiles of the CVA learned from defaults and those of the nested Monte Carlo estimator (or of the intensity-based CVA learner represented by the right panel in Figure 5).

[10] RMSE restricted to the realizations where the benchmark is non-zero.

[11] in spite of the slight variation in the way these errors are computed (cf. the captions of Figures 7 and 9).

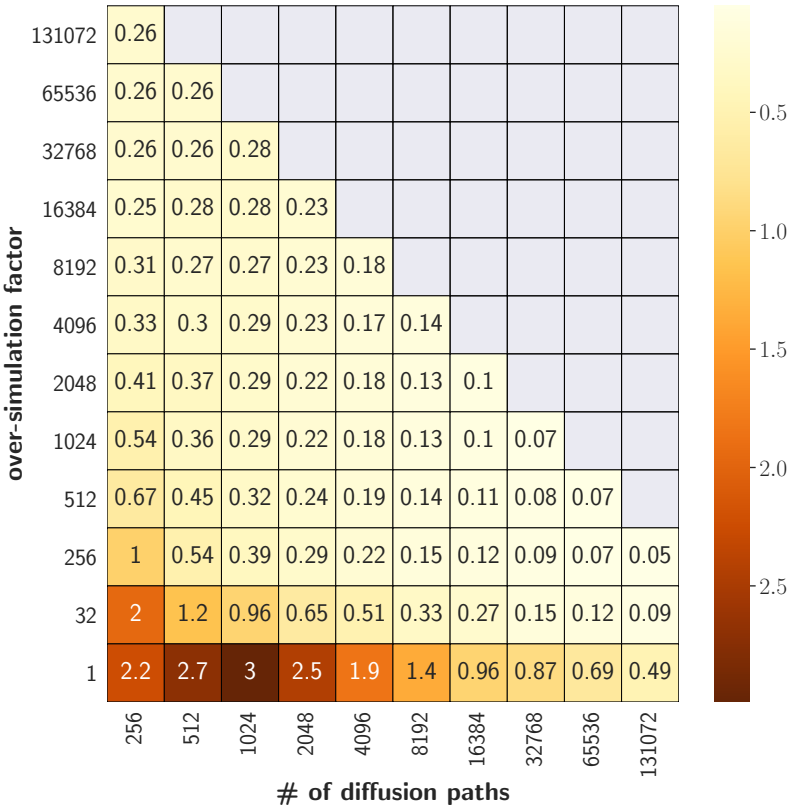

Figure 7: Relative RMSE of the prediction against a nested Monte Carlo benchmark at the pricing time $i = 5$ years, for different combinations of the number of market paths M and of the hierarchical simulation factor N, when the nested Monte Carlo benchmark is non-zero. The error here is a Monte Carlo estimate of $\sqrt{\mathbb{E}\left[\left(\frac{\mathrm{CVA}_{\mathrm{pred}}-\mathrm{CVA}_{\mathrm{nested}}}{\mathrm{CVA}_{\mathrm{nested}}}\right)^2\right]}$ where $\mathrm{CVA}_{\mathrm{pred}}$ is the CVA estimate predicted by the considered neural network given a state of market and default factors and $\mathrm{CVA}_{\mathrm{nested}}$ is a nested Monte Carlo estimator given the same state.

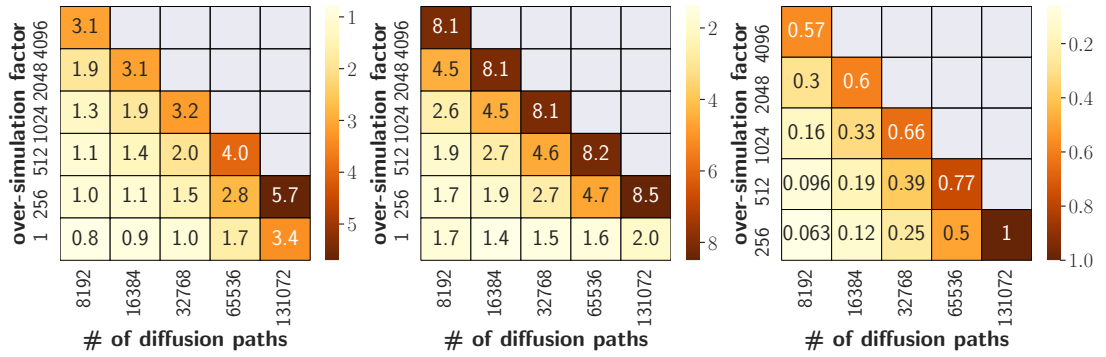

Figure 8: Simulation times in seconds (left) and training times in minutes (center) and RAM usage as a % of its maximum usage over all the displayed experiments (right), for different combinations of the number of market paths M and of the hierarchical simulation factor N.

The above results show that hierarchical simulation is essential to a defaults-based CVA learner. The efficiency of Heuristic 1 is attested numerically on the double basis of our nested Monte Carlo benchmark (Figure 7) and of the twin Monte Carlo validation procedure of Section 2.4 (Figure 9). The fact that Heuristic 1 already has merit in such case (with not so stable $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$) can be put to the credit of the method.

4.5 Conclusion

The bottom row of Figure 7 and the first plot ($N = 1$) of Figure 11 illustrate that a path-wise CVA cannot be learned based on the hybrid market and defaults formulation (27) without hierarchical simulation: For $M = 16384$ and 131072, the corresponding errors with respect to the benchmark nested Monte Carlo are 96% and 49%. However, increasing N from 1 (bottom row) to 256 brings these errors down to 11% and to 5%, while for $M = 1024$ and $N = 512$ the error is 1%. As visible from Figure 8, the simulation times are only marginally increased when increasing the hierarchical simulation factor N (while increasing the number of diffusion paths M increases the simulation time approximately proportionally). These results show that the hierarchical simulation technique is key to the success of a learning approach involving a combination of diffusive and default risk factors.

Even after writing an optimized GPU implementation for the nested Monte Carlo estimator, the latter takes at least 32 minutes on the same hardware as above to compute that estimator for $M = 16384$ and $\sqrt{M} = 128$ inner paths[12], compared to approximately 8 minutes in the case of the learning approach with a very high hierarchical simulation factor ($N = 2048$). Moreover, going to higher XVA layers such as the FVA and the

[12]However, when doing the error computations and in all plots, we used 1024 inner paths to get benchmark CVAs that are sufficiently accurate *point-wise* and be able to get accurate tail estimates, and nested Monte Carlo simulation thus takes 8 times more computation time.

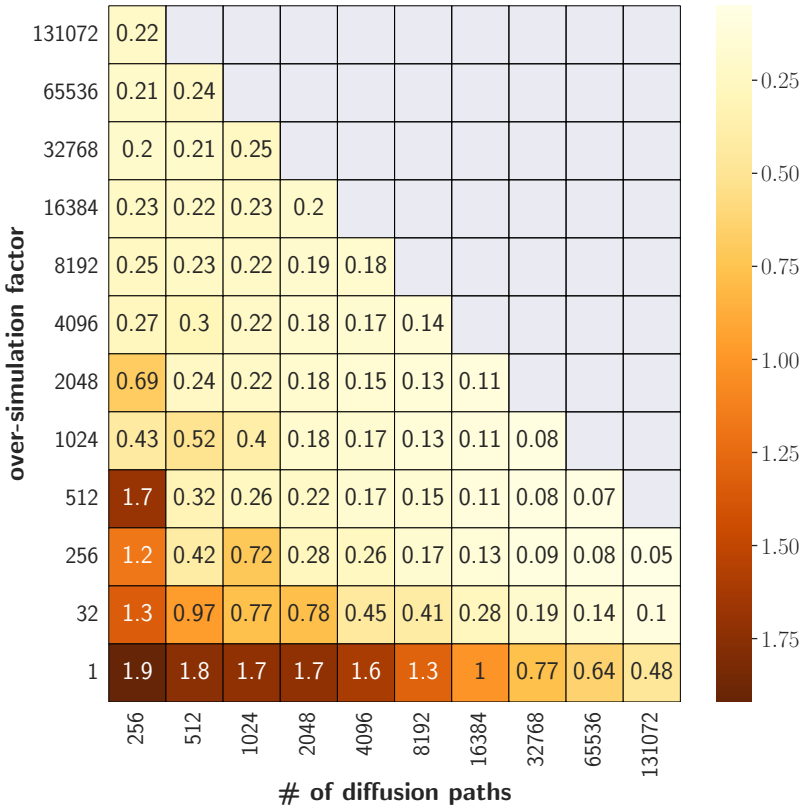

Figure 9: Relative RMSE of the prediction against the ground-truth CVA at the pricing time $i = 5$ years, computed by the twin simulation approach of Section 2.4 for different combinations of the number of market paths M and of the hierarchical simulation factor N. The error here is a Monte Carlo estimate of $\sqrt{\frac{\mathbb{E}\left[\left(\mathrm{CVA}_{\mathrm{pred}}-\mathrm{CVA}_{\mathrm{exact}}\right)^2\right]}{\mathbb{E}\left[\mathrm{CVA}^2_{\mathrm{exact}}\right]}}$, where $\mathrm{CVA}_{\mathrm{pred}}$ is the CVA estimate predicted by the considered neural network given a state of market and default factors and $\mathrm{CVA}_{\mathrm{exact}}$ is the exact CVA (the value of which does not need to be computed, by virtue of (7)-(8)), given the same state.

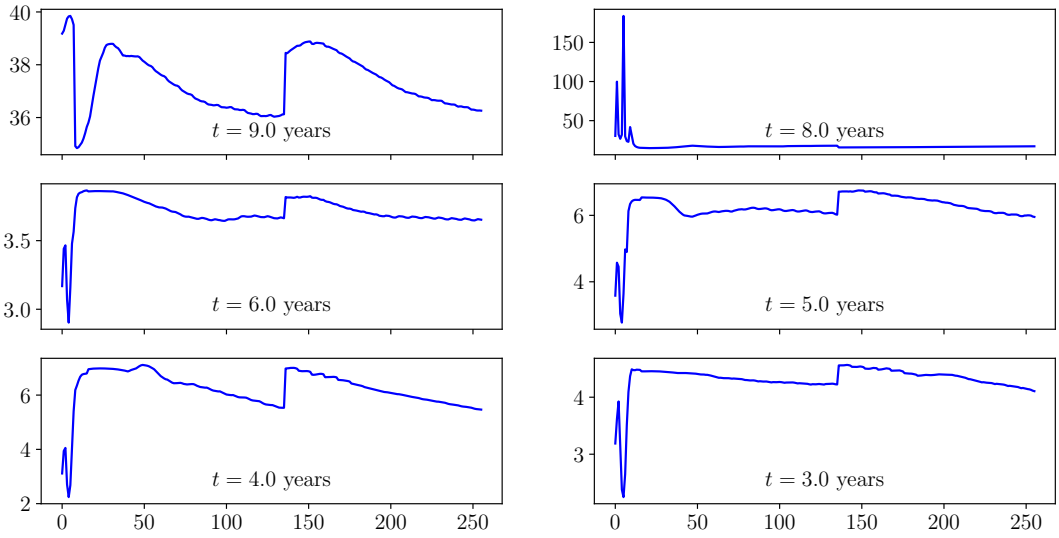

Figure 10: $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$ at different pricing time steps i (panels) and SGD iterations (x axes).

KVA, a nested Monte Carlo approach would become $\sqrt{M}$ times slower per each new layer (Abbas-Turki et al., 2018, Section 3.3), whereas a regression approach would just become slower by a constant each time a new XVA layer is added. In addition, learned XVA metrics can be used in prediction at a very low cost (inference is very fast as it involves no automatic differentiation or stochastic gradient descent), whereas nested Monte Carlo numbers must be recomputed from scratch every time.

In a follow-up paper, the learning-based hierarchical simulation scheme of this paper will be deployed systematically on the whole suite of the XVA metrics and of the embedded risk measures, resorting to the twin Monte Carlo procedure of Section 2.4 for the related validation task (fault of a feasible nested Monte Carlo benchmark beyond the CVA case of this paper).

A Technical Proofs

The following proofs use arguments from Shapiro, Dentcheva, and Ruszczyński (2014) and extend similar results to the conditionally independent, hierarchical simulation case. Theorem 1 extends the finite case in Shapiro, Dentcheva, and Ruszczyński (2014, Section 5.3.1). The major modifications in the proof are the use of a conditional moment generating function, the establishment of a large deviation upper-bound based on it, and the strict convexity of $\log(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N])$ with respect to t that becomes more technical in the conditional case. Then, similar to Shapiro, Dentcheva, and Ruszczyński (2014, Section 5.3.2), Theorem 2 extends these results to the infinite and bounded case, by Lipschitz continuity arguments. In both cases we rely on the following:

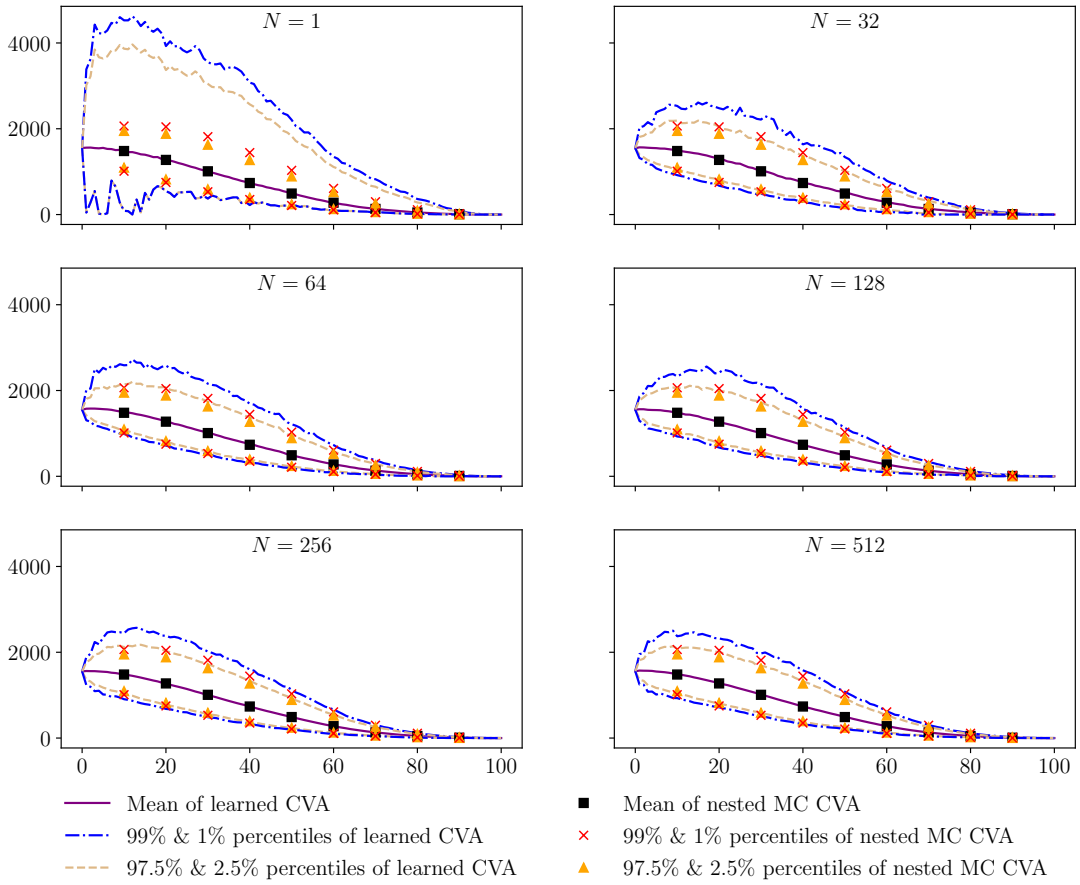

Figure 11: Learned and nested Monte Carlo CVA processes for various hierarchical simulation factors N (x-axis: pricing times, y-axis: CVA levels; $M = 16384$). Statistics computed using out-of-sample paths.

Lemma 1. *Let $\varphi : \mathcal{X} \times \mathcal{Y} \to R$ be such that $\varphi(X, Y)$ is integrable, does not degenerate to a constant and that, for all $z \in \mathbb{R}$ and $y \in V$, $\mathcal{M}(z, y) = \mathbb{E}[\exp(z\varphi(X, Y))|Y = y]$ is well-defined. Then the Fenchel conjugate $I_N : a \mapsto \sup_{t\in\mathbb{R}}\{ta - \log(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N])\}$ of $t \mapsto \log(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N])$ is well-defined and*

$$\frac{1}{M}\log\{\mathbb{Q}(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l}, Y^k) \geq a)\} \leq -I_N(a) \quad \textit{for all } a > \mathbb{E}[\varphi(X, Y)] \tag{31}$$

$$I_N(a) = \frac{(a - \mathbb{E}[\varphi(X, Y)])^2}{\frac{1}{N}\mathbb{E}[\mathrm{Var}(\varphi(X, Y)|Y)] + \mathrm{Var}(\mathbb{E}[\varphi(X, Y)|Y])} + o(|a - \mathbb{E}[\varphi(X, Y)]|^2). \tag{32}$$

Proof. Let $t > 0$. Applying the Markov inequality, we have:

$$\begin{aligned}\mathbb{Q}(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\geq a) &= \mathbb{Q}(\exp(\frac{t}{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k))\geq \exp(Mta))\\ &\leq \exp(-Mta)\mathbb{E}[\exp(\frac{t}{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k))]. \quad (33)\end{aligned}$$

For every $i \in \{1,\dots,M\}$, denote $Z_N^i = \exp(\frac{t}{N}\sum_{j=1}^{N}\varphi(X^{i,j},Y^i))$. By using the tower property repeatedly, one can show recursively that for all $i \in \{1,\dots,M\}$, denoting $\mathcal{Z}_i = \sigma(Z_{M,N}^1,\dots,Z_{M,N}^{M-i},Y^{M-i+1},\dots Y^M)$:

$$\begin{aligned}\mathbb{E}[\exp(\frac{t}{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k))] &= \mathbb{E}[\mathbb{E}[(\prod_{k=1}^{M-i+1}Z_N^k)(\prod_{k=1}^{i-1}\mathcal{M}(\frac{t}{N},Y^{M-k+1})^N)|\mathcal{Z}_i]]\\ &= \mathbb{E}[(\prod_{k=1}^{M-i}Z_N^k)(\prod_{k=1}^{i-1}\mathcal{M}(\frac{t}{N},Y^{M-k+1})^N)\mathbb{E}[Z_N^{M-i+1}|Y^{M-i+1}]]\\ &= \mathbb{E}[(\prod_{k=1}^{M-i}Z_N^k)(\prod_{k=1}^{i}\mathcal{M}(\frac{t}{N},Y^{M-k+1})^N)].\end{aligned} \quad (34)$$

In particular, this identity for $i = M$ yields (recalling Y and the Y^k are i.i.d.)

$$\mathbb{E}[\exp(\frac{t}{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k))] = (\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N])^M.$$

Hence, by (33),

$$\frac{1}{M}\log\{\mathbb{Q}(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\geq a)\}\leq -ta+\log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]).$$

The inequality being true for arbitrary $t > 0$, taking the infimum over $t > 0$ on the RHS yields

$$\frac{1}{M}\log\big(\mathbb{Q}(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\geq a)\big)\leq -\sup_{t>0}\big(ta-\log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N])\big).$$

In order to establish (31), it remains to show that $t \mapsto \log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N])$ is convex and that $I_N(a) = \sup_{t>0}\{ta - \log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N])\}$.

Define $\Lambda(t) = \log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N])$. As a moment generating function is infinitely differentiable on its domain of definition, Λ is infinitely differentiable. After computations we get $\Lambda'(0) = \mathbb{E}[\varphi(X,Y)]$ and $\Lambda''(t) = \frac{\det(\mathbb{E}[A])}{\mathbb{E}[\mathbb{E}[V|Y]^N]^2}$ for the 2×2 random matrix

$$A = \left[\begin{array}{c|c} \frac{1}{N}\mathbb{E}[U^2V|Y]\mathbb{E}[V|Y]^{N-1}+(1-\frac{1}{N})\mathbb{E}[UV|Y]^2\mathbb{E}[V|Y]^{N-2} & \mathbb{E}[UV|Y]\mathbb{E}[V|Y]^{N-1} \\ \hline \mathbb{E}[UV|Y]\mathbb{E}[V|Y]^{N-1} & \mathbb{E}[V|Y]^N \end{array}\right],$$

where $U = \varphi(X,Y)$ and $V = \exp(\frac{t}{N}\varphi(X,Y))$. We have:

$$A = \mathbb{E}[V|Y]^{N-2}\left[\begin{array}{c|c} \frac{1}{N}\mathbb{E}[U^2V|Y]\mathbb{E}[V|Y] + (1-\frac{1}{N})\mathbb{E}[UV|Y]^2 & \mathbb{E}[UV|Y]\mathbb{E}[V|Y] \\ \hline \mathbb{E}[UV|Y]\mathbb{E}[V|Y] & \mathbb{E}[V|Y]^2 \end{array}\right]$$

Let $\alpha, \beta \in \mathbb{R}$ and

$$\begin{aligned}\Delta_{\alpha,\beta} &= \frac{1}{\mathbb{E}[V|Y]^{N-2}}[\alpha,\beta]A\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \\ &= \alpha^2(\frac{1}{N}\mathbb{E}[U^2V|Y]\mathbb{E}[V|Y] + (1-\frac{1}{N})\mathbb{E}[UV|Y]^2) + \beta^2\mathbb{E}[V|Y]^2 + 2\alpha\beta\mathbb{E}[UV|Y]\mathbb{E}[V|Y].\end{aligned}$$

From the Cauchy-Schwarz inequality, we have:

$$\mathbb{E}[U^2V|Y]\mathbb{E}[V|Y] \geq \mathbb{E}[UV|Y]^2. \tag{35}$$

We then have:

$$\begin{aligned}\Delta_{\alpha,\beta} &\geq \alpha^2\mathbb{E}[UV|Y]^2 + \beta^2\mathbb{E}[V|Y]^2 + 2\alpha\beta\mathbb{E}[UV|Y]\mathbb{E}[V|Y] \\ &= (\alpha\mathbb{E}[UV|Y] + \beta\mathbb{E}[V|Y])^2 \geq 0.\end{aligned}$$

Furthermore, we have $\Delta_{\alpha,\beta} > 0$ because $\Delta_{\alpha,\beta} = 0$ would imply equality in (35), which in turn is only attained when $U = 1$ *a.s.*, contradicting the non-degeneracy assumption made on φ. Therefore A is a.s. positive definite. Hence $\mathbb{E}[A]$ is positive definite, i.e. $\det(\mathbb{E}[A]) > 0$. In conclusion, Λ is strictly convex.

Let $\psi(t) = ta - \Lambda(t)$. For $a > \mathbb{E}[\varphi(X,Y)]$, we have

$$\psi'(0) = a - \Lambda'(0) = a - \mathbb{E}[\varphi(X,Y)] > 0.$$

Therefore there exists some $\varepsilon > 0$ such that $\psi'(t) > 0$ for all $t \in (0,\varepsilon)$. Hence, for all $t \in (0,\varepsilon)$, we have $ta - \Lambda(t) > 0$.

On the other hand, we have:

$$\sup_{t<0}\{ta - \Lambda(t)\} = \sup_{t<0}\{\underbrace{t(a - \mathbb{E}[\varphi(X,Y)])}_{<0} + t\mathbb{E}[\varphi(X,Y)] - \Lambda(t)\}.$$

Using convexity and concavity inequalities, we have

$$\begin{aligned}\Lambda(t) &= \log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]) \\ &\geq N\log(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)]) \\ &\geq t\mathbb{E}[\varphi(X,Y)].\end{aligned}$$

We then obtain that

$$\sup_{t<0}\{ta - \Lambda(t)\} \leq 0.$$

Thus

$$\sup_{t>0}\{ta - \log(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N])\} = \sup_{t\in\mathbb{R}}\{ta - \log(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N])\} = I_N(a),$$

which finishes to prove (31). As the Fenchel conjugate of the twice differentiable and strictly convex function Λ, I_N is twice differentiable and we have

$$\begin{aligned} I_N(\mathbb{E}[\varphi(X,Y)]) &= -\Lambda(0) = 0 \\ I'_N(\mathbb{E}[\varphi(X,Y)]) &= 0 \\ I''_N(\mathbb{E}[\varphi(X,Y)]) &= \frac{1}{\Lambda''(0)} = \frac{1}{\frac{1}{N}\mathbb{E}[\mathrm{Var}(\varphi(X,Y)|Y)] + \mathrm{Var}(\mathbb{E}[\varphi(X,Y)|Y])}. \end{aligned}$$

Hence a Taylor expansion around $\mathbb{E}[\varphi(X,Y)]$ gives (32). ∎

Remark 6. *The Taylor approximation for the Fenchel conjugate in Lemma 1 would become exact if we added the hypothesis that $\varphi(X,Y)$ is Gaussian conditionally on Y and that $\mathbb{E}[\varphi(X,Y)|Y]$ is Gaussian. We can still obtain a similar expression as an exact lower-bound if we just assume sub-Gaussianity as in Theorems 1-2.*

A.1 Proof of Theorem 1

Under the assumptions of Theorem 1, we have in view of (13):

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)) \leq \sum_{\theta\in E\setminus S^{\epsilon}(E)} \mathbb{Q}(\bigcap_{\theta'\in E}\{\hat{G}_{M,N}(\theta) \leq \hat{G}_{M,N}(\theta') + \delta\}) \tag{36}$$

Let us consider a minimizer $\theta^\star$ of G over E, hence

$$\forall\theta \in E \setminus S^{\epsilon}, G(\theta^\star) < G(\theta) - \epsilon. \tag{37}$$

We then have:

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)) \leq \sum_{\theta\in E\setminus S^{\epsilon}(E)} \mathbb{Q}(\hat{G}_{M,N}(\theta) \leq \hat{G}_{M,N}(\theta^\star) + \delta) \tag{38}$$

Define:

$$\hat{\Gamma}_{M,N}(\theta) = \frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\Gamma(\theta^\star, \theta, X^{k,l}, Y^k) = \hat{G}_{M,N}(\theta^\star) - \hat{G}_{M,N}(\theta)$$

Inequality (38) can then be rewritten:

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}(E)) \leq \sum_{\theta\in E\setminus S^{\epsilon}(E)} \mathbb{Q}(\hat{\Gamma}_{M,N}(\theta) \geq -\delta). \tag{39}$$

Now observe that

$$\mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)] = G(\theta^\star) - G(\theta) < -\epsilon < -\delta \tag{40}$$

for all $\theta \in E$. Hence, from inequality (39) and Lemma 1, we obtain

$$\begin{aligned} \mathbb{Q}(\hat{S}^\delta_{M,N}(E) \not\subset S^\epsilon(E)) &\leq |E| \max_{\theta \in E \setminus S^\epsilon(E)} \{\mathbb{Q}(\hat{\Gamma}_{M,N}(\theta) \geq -\delta)\} \\ &\leq |E| \exp(-M \min_{\theta \in E \setminus S^\epsilon(E)} \{I^\theta_N(-\delta)\}), \end{aligned} \tag{41}$$

where I^θ_N is the Fenchel conjugate of $\log(\mathbb{E}[\mathcal{M}^\theta(\frac{t}{N}, Y)^N])$ and $\mathcal{M}^\theta(z, y) = \mathbb{E}[\exp(z\Gamma(\theta^\star, \theta, X, Y))|Y = y]$ for every $\theta \in E$. From inequalities (14) and (15), we get for every $\theta \in E \setminus S^\epsilon(E)$ and $t \in \mathbb{R}$:

$$\log(\mathbb{E}[\mathcal{M}^\theta(\frac{t}{N}, Y)^N]) \leq \mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)]t + \frac{1}{2}(\frac{b_1^2}{N} + b_2^2)t^2$$

Thus, for every $\theta \in E \setminus S^\epsilon(E)$, we have:

$$\begin{aligned} I^\theta_N(-\delta) &= \sup_{t \in \mathbb{R}} \{t(-\delta - \mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)]) + \mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)]t - \log(\mathbb{E}[\mathcal{M}^\theta(\frac{t}{N}, Y)^N])\} \\ &\geq \sup_{t \in \mathbb{R}} \{t(-\delta - \mathbb{E}[\Gamma(\theta^\star, \theta, \xi, \Gamma)]) - \frac{1}{2}(\frac{b_1^2}{N} + b_2^2)t^2\} \\ &= \frac{(-\delta - \mathbb{E}[\Gamma(\theta^\star, \theta, \xi, \Gamma)])^2}{2(\frac{b_1^2}{N} + b_2^2)} \\ &> \frac{(\epsilon - \delta)^2}{2(\frac{b_1^2}{N} + b_2^2)} \end{aligned} \tag{42}$$

where the last inequality comes from (40). This yields the result.

A.2 Proof of Theorem 2

From the Lipschitz assumption (17), it follows that for all $\theta, \theta' \in \Theta$:

$$|\hat{G}_{M,N}(\theta) - \hat{G}_{M,N}(\theta')| \leq \hat{L}_{M,N}\|\theta - \theta'\| \quad \text{a.s.}$$

and

$$|G(\theta) - G(\theta')| \leq \mathbb{E}[L(X, Y)]\|\theta - \theta'\|$$

where

$$\hat{L}_{M,N} = \frac{1}{MN} \sum_{k=1}^{M} \sum_{l=1}^{N} L(X^{k,l}, Y^k).$$

Let $0 < \delta' < \epsilon'$ and let $\Theta' = \{\theta_1, \ldots, \theta_C\}$ be a minimal ϱ-covering of Θ, for a given $\varrho > 0$. We then have (cf. Vershynin (2018, Corollary 4.2.13 p.85)):

$$C \leq (\frac{2D}{\varrho} + 1)^d.$$

Let $\tilde{\Theta} = \Theta' \cup \{\theta^\star\}$, where $\theta^\star \in \mathrm{Argmin}_{\theta\in\Theta} G(\theta)$. We have $|\tilde{\Theta}| \leq (\frac{2D}{\varrho}+1)^d + 1$. Theorem 1 yields:

$$\mathbb{Q}(\hat{S}^{\delta'}_{M,N}(\tilde{\Theta}) \not\subset S^{\epsilon'}(\tilde{\Theta})) < \big((\frac{2D}{\varrho}+1)^d + 1\big)\exp(\frac{-M(\epsilon'-\delta')^2}{2(\frac{b_1^2}{N}+b_2^2)}).$$

Our next goal is to show the following assertion for suitable choices of δ' and ϵ' and for any $L' > \bar{L}$ (note that $\bar{L} < \infty$):

$$\{\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}\} \cap \{L' \geq \hat{L}_{M,N}\} \subset \{\hat{S}^{\delta'}_{M,N}(\tilde{\Theta}) \not\subset S^{\epsilon'}(\tilde{\Theta})\}. \tag{43}$$

Let $L' > \bar{L}$ and assume that $\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}$ and $L' \geq \hat{L}_{M,N}$, and that $\hat{S}^{\delta'}_{M,N}(\tilde{\Theta}) \subset S^{\epsilon'}(\tilde{\Theta})$. In particular, there exists $\theta \in \Theta$ such that $\hat{G}_{M,N}(\theta) \leq \min_\Theta \hat{G}_{M,N} + \delta$ and $G(\theta) > \min_\Theta G + \epsilon$. Let then $\theta' \subset \tilde{\Theta}$ such that $\|\theta - \theta'\| < \varrho$. We have:

$$\hat{G}_{M,N}(\theta') \leq \hat{G}_{M,N}(\theta) + L'\varrho \leq \min_{\Theta} \hat{G}_{M,N} + \delta + L'\varrho.$$

Thus, if we choose $\delta' = \delta + L'\varrho$, then we have $\hat{G}_{M,N}(\theta') \leq \min_\Theta \hat{G}_{M,N} + \delta'$, and consequently $\theta' \in \hat{S}^{\delta'}_{M,N}(\tilde{\Theta})$ (as $\tilde{\Theta} \subset \Theta$). Hence, by our assumption $\hat{S}^{\delta'}_{M,N}(\tilde{\Theta}) \subset S^{\epsilon'}(\tilde{\Theta})$, we get that $\theta' \in S^{\epsilon'}(\tilde{\Theta})$. Thus:

$$G(\theta) \leq G(\theta') + L'\varrho \leq \min_{\tilde{\Theta}} G + \epsilon' + L'\varrho = \min_{\Theta} G + \epsilon' + L'\varrho,$$

as $\min_{\tilde{\Theta}} G = \min_\Theta G$ (since $\theta^\star \in \tilde{\Theta}$). Hence, if we also choose $\epsilon' = \epsilon - L'\varrho$ with ϱ such that $\varrho < \frac{\epsilon-\delta}{2L'}$ in order to ensure that $0 < \delta' < \epsilon'$, then $G(\theta) \leq \min_\Theta G + \epsilon$, which contradicts our assumption and proves (43). Thus

$$\mathbb{Q}(\{\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}\} \cap \{L' \geq \hat{L}_{M,N}\}) \leq \mathbb{Q}(\hat{S}^{\delta'}_{M,N}(\tilde{\Theta}) \not\subset S^{\epsilon'}(\tilde{\Theta})).$$

As a consequence,

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}) \leq \mathbb{Q}(\hat{S}^{\delta'}_{M,N}(\tilde{\Theta}) \not\subset S^{\epsilon'}(\tilde{\Theta})) + \mathbb{Q}(\hat{L}_{M,N} > L').$$

But applying Hoeffding's lemma on the inequalities (18) and (19), Lemma 1, and proceeding similarly as in (42) to establish a lower bound for the Legendre transform, yields

$$\mathbb{Q}(\hat{L}_{M,N} > L') \leq \exp(\frac{-M(L'-\bar{L})^2}{2(\frac{\ell_1^2}{N}+\ell_2^2)}). \tag{44}$$

Thus,

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}) \leq \big((\frac{2D}{\varrho}+1)^d + 1\big)\exp(\frac{-M(\epsilon'-\delta')^2}{2(\frac{b_1^2}{N}+b_2^2)}) + \exp(\frac{-M(L'-\bar{L})^2}{2(\frac{\ell_1^2}{N}+\ell_2^2)}).$$

We have $\epsilon' - \delta' = \epsilon - \delta - 2L'\varrho$. Finally, if we choose $\varrho = \frac{\epsilon-\delta}{4L'}$, then we get $\epsilon' - \delta' = \frac{\epsilon-\delta}{2}$ and

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}) \leq \big((\frac{8L'D}{\epsilon-\delta}+1)^d + 1\big)\exp(\frac{-M(\epsilon-\delta)^2}{8(\frac{b_1^2}{N}+b_2^2)}) + \exp(\frac{-M(L'-\bar{L})^2}{2(\frac{\ell_1^2}{N}+\ell_2^2)}).$$

This concludes our proof.

B Market and Credit Model in Continuous Time

For every economy e, the short-rate $r^{\langle e\rangle}$ and the exchange rate $\chi^{\langle e\rangle}$ against the reference currency respectively follow Vasicek and log-normal dynamics

$$\begin{aligned} dr_t^{\langle e\rangle} &= a^{\langle e\rangle}(b^{\langle e\rangle} - r_t^{\langle e\rangle})dt + \sigma^{r,\langle e\rangle} d\tilde{B}_t^{r,\langle e\rangle} \\ d\log \chi_t^{\langle e\rangle} &= (r_t^{\langle 0\rangle} - r_t^{\langle e\rangle} - \frac{1}{2}|\sigma^{\chi,\langle e\rangle}|^2)dt + \sigma^{\chi,\langle e\rangle} dB_t^{\chi,\langle e\rangle}. \end{aligned} \tag{45}$$

For both the bank ("$c = 0$") and every counterparty c ($\neq 0$), the process $\gamma^{\langle c\rangle}$ (funding spread for $c = 0$ and default intensity for $c \geq 1$) follows CIR dynamics

$$d\gamma_t^{\langle c\rangle} = \alpha^{\langle c\rangle}(\delta^{\langle c\rangle} - \gamma_t^{\langle c\rangle})dt + \nu^{\langle c\rangle}\sqrt{\gamma_t^{\langle c\rangle}} dB_t^{\gamma,\langle c\rangle}. \tag{46}$$

In the above, for every e, $\tilde{B}^{r,\langle e\rangle}$ is a $\mathbb{Q}^{\langle e\rangle}$ Brownian motion and, for every client c and economy e, $B^{\chi,\langle e\rangle}$ and $B^{\gamma,\langle c\rangle}$ are $\mathbb{Q}^{\langle 0\rangle}$ Brownian motions. Here $\mathbb{Q}^{\langle e\rangle}$ is the risk-neutral measure corresponding to the numeraire $\exp(\int_0^\cdot r_s^{\langle e\rangle} ds)$, and $a^{\langle\cdot\rangle}$, $b^{\langle\cdot\rangle}$, $\sigma^{\cdot,\langle\cdot\rangle}$, $\alpha^{\langle\cdot\rangle}$, $\delta^{\langle\cdot\rangle}$, $\nu^{\langle\cdot\rangle}$ are model parameters calibrated using liquid market instruments.

In line with the fundamental theorem of asset pricing, for any asset Z priced in a foreign currency $e \geq 1$, $\exp(-\int_0^\cdot r_s^{\langle e\rangle} ds)Z$ and $\exp(-\int_0^\cdot r_s^{\langle 0\rangle} ds)\chi^{\langle e\rangle} Z$ are martingales with respect to $\mathbb{Q}^{\langle e\rangle}$ and $\mathbb{Q}^{\langle 0\rangle}$ respectively. In particular,

$$\mathbb{E}^{\mathbb{Q}^{\langle e\rangle}}[e^{-\int_0^t r_s^{\langle e\rangle} ds} Z_t] = Z_0 = \frac{1}{\chi_0^{\langle e\rangle}} \mathbb{E}^{\mathbb{Q}^{\langle 0\rangle}}[e^{-\int_0^t r_s^{\langle 0\rangle} ds} \chi_t^{\langle e\rangle} Z_t].$$

Thus,

$$\begin{aligned} \frac{d\mathbb{Q}^{\langle e\rangle}}{d\mathbb{Q}^{\langle 0\rangle}}_{|t} &= \exp(\int_0^t (r_s^{\langle e\rangle} - r_s^{\langle 0\rangle})ds) \frac{\chi_t^{\langle e\rangle}}{\chi_0^{\langle e\rangle}} \\ &= \exp(-\frac{1}{2}\big(\sigma^{\chi,\langle e\rangle}\big)^2 t + \sigma^{\chi,\langle e\rangle} B_t^{\chi,\langle e\rangle}). \end{aligned}$$

Hence, by Girsanov's theorem, if we define $B_t^{r,\langle e\rangle}$ such that:

$$d\tilde{B}_t^{r,\langle e\rangle} = dB_t^{r,\langle e\rangle} - \sigma^{\chi,\langle e\rangle} d\langle B^{r,\langle e\rangle}, B^{\chi,\langle e\rangle}\rangle_t,$$

then $B_t^{r,\langle e\rangle}$ is a $\mathbb{Q}^{\langle 0\rangle}$ Brownian motion. In particular, assuming $d\langle B^{r,\langle e\rangle}, B^{\chi,\langle e\rangle}\rangle_t = \rho^{\langle e\rangle} dt$, we get the following $\mathbb{Q}^{\langle 0\rangle}$ dynamics for the short-rate of economy e:

$$dr_t^{\langle e\rangle} = (a^{\langle e\rangle}(b^{\langle e\rangle} - r_t^{\langle e\rangle}) - \rho^{\langle e\rangle}\sigma^{\chi,\langle e\rangle})dt + \sigma^{r,\langle e\rangle} dB_t^{r,\langle e\rangle}.$$

For every counterparty c, the default time $\tau^{\langle c\rangle}$ can be modeled as a the stopping time $\inf\{t > 0; \int_0^t \gamma_s^{\langle c\rangle} ds \geq \epsilon^{\langle c\rangle}\}$, where $\epsilon^{\langle c\rangle}$ is a standard exponential. That is, for every $t \geq 0$,

$$\mathbb{1}_{\{\tau^{\langle c\rangle} \leq t\}} = 1 \Leftrightarrow \tau^{\langle c\rangle} \leq t \Leftrightarrow \int_0^t \gamma_s^{\langle c\rangle} ds \geq \epsilon^{\langle c\rangle}. \tag{47}$$

For the instruments, we assume a book comprised of interest rate swaps at par at inception. For each swap, we denote the set of its reset dates by $\mathcal{R}$ and by t_- and t_+ the reset dates respectively immediately preceding and following t. We assume that successive reset dates are regularly spaced by δ, that the swap is spot starting, i.e. $0 \in \mathcal{R}$, and that the swap is paying fixed $\delta\Sigma$, where Σ is the swap rate, and receiving floating $\frac{1}{\mathrm{ZC}_{t_-}(t)} - 1$, where $\mathrm{ZC}_t(t')$ is the price of a zero-coupon bond[13] at time t with maturity t', at each reset date $t \in \mathcal{R} \setminus \{0\}$. Denoting by P_t^{sw} the price of the swap at time t in units of the underlying currency[14], we have for all $t \leq \bar{t} := \max \mathcal{R}$:

$$P_t^{sw} = \begin{cases} \dfrac{\mathrm{ZC}_t(t_+)}{\mathrm{ZC}_{t_-}(t_+)} - \mathrm{ZC}_t(\bar{t}) - \delta\Sigma \displaystyle\sum_{t' \in \mathcal{R}, t' > t} \mathrm{ZC}_t(t') & \text{if } t \notin \mathcal{R} \setminus \{0\} \\ \dfrac{1}{\mathrm{ZC}_{t_-}(t)} - \mathrm{ZC}_t(\bar{t}) - \delta\Sigma(1 + \displaystyle\sum_{t' \in \mathcal{R}, t' > t} \mathrm{ZC}_t(t')) & \text{if } t \in \mathcal{R} \setminus \{0\} \\ 1 - \mathrm{ZC}_0(\bar{t}) - \delta\Sigma \displaystyle\sum_{t' \in \mathcal{R} \setminus \{0\}} \mathrm{ZC}_0(t') & \text{if } t = 0. \end{cases}$$

Remark 7. *The path-dependence induced by the previous reset date can be resorbed by including the short rates of that date among the risk factors Y.*

C Python/CUDA Optimized Implementation Using GPU

Contrary to most use-cases of machine learning where the final product is the trained model and thus execution time is only critical during inference, in the case of learning from simulated data in pricing applications, the training process itself is part of the final product. Hence particular care is needed when writing the training procedures.

We implemented Algorithm 1 using Python programming with the CUDA API (application programming interface). Because the considered problem involves high variances (see Section 3) and thus requires a sufficiently large sample size, both training and inference are not easy to achieve in a reasonable execution time. First, we need to leverage the manycore parallel architecture of GPUs that involves streaming multiprocessors, which are used for the simulation, learning and inference phases. All phases are intertwined and performed inline. Hence we need to carefully optimize each part of the algorithm.

On the simulation side, due to their intrinsically parallel nature, Monte Carlo simulations easily lend themselves to parallelization on GPUs. Nevertheless, various optimizations are needed to achieve a reasonable solution executed within a few seconds (cf. Figure 8 in Section 4.4). We chose to use Python and the CUDA kernels are compiled *just-in-time* using the module numba, which allows to dynamically generate CUDA kernels at run-time.

[13] Note that the price of a zero-coupon bond has a closed-form under our affine short-rate model.

[14] The swap prices are then to be multiplied by the cross-currency exchange rate processes to have all prices in the same reference currency.

Regarding learning, we opted for PyTorch for its proximity to the CUDA programming model and its *just-in-time* compiler allowing for static computation graphs and automatic fusion, whenever appropriate, of the kernels associated with the PyTorch operations used by the model.

We used most of the optimization techniques introduced in Abbas-Turki, Diallo, and Crépey (2018), except those related to regressions since these are replaced here by neural networks. We also introduced several additional optimizations, the most important one being to judiciously manage the CPU and GPU memories. A naive solution would involve the CPU/GPU virtual unified memory (NVIDIA Corporation, 2020) and let the compiler choose. However, this usually results in sub-optimal memory accesses. Our choice rather targets an efficient use of the GPU memory space, a reduction of CPU/GPU transfer and an optimized transfer when needed. These optimizations and implementation choices are developed in the accompanying Github repository[15].

Having in mind a portfolio of the order of one million trades spread over maturities ranging over 50 years and involving a few thousands of clients, the computational CPU ressources typically available in banks hardly allow computing (even overnight) a mark-to-market cube with more than 10^4 paths. Switching to GPU ressources (as required anyway if training path-wise XVA metrics is envisioned) could allow computing a mark-to-market cube with 10^5 to 10^6 paths in about one hour of computations spread over a few GPUs.

In fact, while we performed our computations using only one GPU, we expect a bank to have access to more than just a single GPU. Monte Carlo simulations and stochastic gradient descent can easily be adapted to multi-GPU setups: see for instance (Abbas-Turki et al., 2014) for a study of the parallelization of a Monte Carlo pricing procedure over multiple GPUs and nodes. As for training, the main parallelization issue is the ability of the optimization algorithm to scale over multiple GPUs or nodes: see in particular (Recht et al., 2011) for an asynchronous SGD algorithm which does not require synchronization between the different workers involved. For a discussion on synchronous vs. asynchronous SGD in the context of distributed optimization or training, we refer the reader to (Chen et al., 2016). Combining these approaches would allow for an implementation that can readily scale to multiple GPUs and nodes, reducing the computation times proportionally to the total number of GPU nodes that are available.

Bibliography

Abbas-Turki, L., B. Diallo, and S. Crépey (2018). XVA principles, nested Monte Carlo strategies, and GPU optimizations. *International Journal of Theoretical and Applied Finance 21*, 1850030.

Abbas-Turki, L. A., S. Vialle, B. Lapeyre, and P. Mercier (2014). Pricing derivatives on graphics processing units using monte carlo simulation. *Concurrency and Computation: Practice and Experience 26*(9), 1679–1697.

[15] `https://github.com/BouazzaSE/NeuralXVA`, see the coverpage of the paper.

Albanese, C., S. Crépey, R. Hoskinson, and B. Saadeddine (2021). XVA analysis from the balance sheet. *Quantitative Finance 21*(1), 99–123.

Bengio, Y., A. Courville, and I. Goodfellow (2016). *Deep learning*. MIT press Cambridge.

Bergstra, J. and Y. Bengio (2012). Random search for hyper-parameter optimization. *Journal of machine learning research 13*(Feb), 281–305.

Bozinovski, S. (2020). Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica 44*(3).

Carmona, R. and S. Crépey (2010). Particle methods for the estimation of credit portfolio loss distributions. *International Journal of Theoretical and Applied Finance 13*(04), 577–602.

Cesari, J., J. Aquilina, and N. Charpillon (2010). *Modelling, Pricing, and Hedging Counterparty Credit Exposure*. Springer.

Chen, J., X. Pan, R. Monga, S. Bengio, and R. Jozefowicz (2016). Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981*.

Chizat, L. and F. Bach (2018). On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems 31*.

Choromanska, A., M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pp. 192–204. PMLR.

Crépey, S. (2022). Positive XVAs. *Frontiers of Mathematical Finance 1*(3), 425–465. doi: 10.3934/fmf.2022003.

Crépey, S. and S. Song (2015). BSDEs of counterparty risk. *Stochastic Processes and their Applications 125*(8), 3023–3052.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems 2*(4), 303–314.

Du, S., J. Lee, H. Li, L. Wang, and X. Zhai (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pp. 1675–1685. Proceedings of Machine Learning Research.

E, W., J. Han, and A. Jentzen (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics 5*(4), 370–398.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Applications of mathematics: stochastic modelling and applied probability. Springer.

Gnoatto, A., C. Reisinger, and A. Picarelli (2020). Deep xVA solver–a neural network based counterparty credit risk management framework. *Available at SSRN 3594076*.

Gordy, M. B. and S. Juneja (2010). Nested simulation in portfolio risk measurement. *Management Science 56*(10), 1833–1848.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks 4*(2), 251–257.

Huge, B. and A. Savine (2020). Differential machine learning: the shape of things to come. *Risk Magazine*, September.

Huré, C., H. Pham, and C. Warin (2020). Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation 89*(324), 1547–1579.

Kidger, P. and T. Lyons (2020). Universal approximation with deep narrow networks. In *Conference on learning theory*, pp. 2306–2327. Proceedings of Machine Learning Research.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Lei, Y., T. Hu, G. Li, and K. Tang (2019). Stochastic gradient descent for nonconvex learning without bounded gradient assumptions. *IEEE transactions on neural networks and learning systems 31*(10), 4394–4400.

Longstaff, F. A. and E. S. Schwartz (2001). Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies 14*(1), 113–147.

Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

NVIDIA Corporation (2020). Programming guide: Cuda toolkit documentation. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`. Accessed: 2020-04-28.

Pan, S. J. and Q. Yang (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*(10), 1345–1359.

Rasmussen, C. and C. Williams (2006). *Gaussian Processes for Machine Learning*. Adaptive computation and machine learning. MIT Press.

Recht, B., C. Re, S. Wright, and F. Niu (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems 24*.

Rockafellar, R. and S. Uryasev (2000). Optimization of conditional value-at-risk. *Journal of risk 2*, 21–42.

Shapiro, A., D. Dentcheva, and A. Ruszczyński (2014). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shorten, C. and T. Khoshgoftaar (2019). A survey on image data augmentation for deep learning. *Journal of Big Data 6*(1), 60.

Tsitsiklis, J. N. and B. Van Roy (2001). Regression methods for pricing complex american-style options. *IEEE Transactions on Neural Networks 12*(4), 694–703.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, Volume 47. Cambridge university press.