Neural Networks for Credit Risk and xVA in a Front Office Pricing Environment

Isabelle Frodé, Viktor Sambergs

Master's thesis
2022:E31

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Abstract

We present a data-driven proof of concept model capable of reproducing expected counterparty credit exposures from market and trade data. The model has its greatest advantages in quick single-contract exposure evaluations that could be used in front office xVA solutions. The data was generated using short rates from the Hull-White One-Factor model. The best performance was obtained from a GRU neural network with two recurrent layers, which with adequate accuracy could reproduce the exposure profile for an interest rate swap contract. Errors were comparable to those expected from a Monte Carlo simulation with 5K paths. With regards to computational efficiency, the proposed model showed great potential in outperforming traditional numerical methods. Further development and calibration to actual market data is required for the model to be applicable in the industry. The proposed architectures may then prove useful, especially for contracts with high-rated counterparties, traded in a normal and liquid market.

Keywords: OTC, xVA, CVA, Counterparty Credit Risk, Interest Rate Swaps, Hull-White Model, Machine Learning, Artificial Neural Networks, Gated Recurrent Units.

Acknowledgments

This project would not have been possible without Nordea and our industry supervisor Shengyao Zhu at Nordea Markets Copenhagen. We are very thankful for all guidance and valuable discussions throughout the project.

We would further like to extend our special regards to our academic supervisor, Associate Professor Magnus Wiktorsson at the Centre for Mathematical Sciences of Lund University, for his wholehearted commitment and for bringing new perspectives. With great enthusiasm, he provided us with important insights and knowledge even beyond the scope of this project.

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ATM	At the Money
BCBS	Basel Committee on Banking Supervision
BCP	Basel Core Principles
BIS	Bank of International Settlements
BPS	Basis Points
BPTT	Backpropagation Through Time
ColVA	Collateral Valuation Adjustment
CSA	Credit Support Annex
CVA	Credit Valuation Adjustment
DVA	Debit Valuation Adjustment,
EAD	Exposure at Default
ENE	Expected Negative Exposure
EPE	Expected Positive Exposure
FVA	Funding Valuation Adjustment
GFC	Global Financial Crisis
GRU	Gated Recurrent Unit
ICP	Insurance Core Principles
IRS	Interest Rate Swap
ISDA	International Swaps and Derivatives Association
KVA	Capital Valuation Adjustment
LIBOR	London Interbank Offered Rate
LSTM	Long Short-term Memory
MC	Monte Carlo
MLP	Multilayer Perceptron
MVA	Margin Valuation Adjustment

OECD	Organisation for Economic Cooperation and Development
OIS	Overnight Index Swap
OTC	Over-the-counter
OTM	Out-of-the-money
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
TVA	Tax Valuation Adjustment
xVA	Valuation Adjustments

Chapter 1

Introduction

1.1 Motivation

The global financial crisis (GFC) spanning over the years 2007-2009 exposed several critical flaws in the valuation methods used in the derivatives markets. The default of many large derivative counterparties during the financial crisis led to an increased demand for counterparty credit risk assessment in the valuation of derivatives contracts. Thus, the framework for Credit Valuation Adjustments (CVA) was rapidly developed. Increased awareness and regulation has since then led to the development of several Valuation Adjustments, such as Funding Valuation Adjustments and Capital Valuation Adjustments, all of which are included in the umbrella term Valuation Adjustments (xVA). Today, even the most simple derivatives contracts are priced at xVA desks in banks using complex methods and large scale Monte Carlo simulations [19].

Comparing contracts on the OTC derivatives market is a regular activity within most banks. A comparison may provide important insights into how different contract specifications, such as maturities and strike rates, affect risk profiles. Today, each evaluation relies on time-consuming and extensive Monte Carlo simulations. We will present a model based on artificial neural networks that maps yield data to the expected credit risk exposure for single interest rate swap contacts. A deep learning model, trained offline, could potentially be a shortcut if it generates an accurate enough estimate of a simulated exposure profile in an instant.

The potential benefits of machine learning approaches in financial applications, with regards to their increased efficiency, has been discussed and researched. Artificial neural networks (ANNs) is a deep learning approach which have grown largely in effectiveness and popularity in recent years. The networks have successively been applied to more complex tasks and performance has increased over time. It is primarily due to improvement in computer hardware, development of software infrastructure and an increasing amount of available data [16].

1.2 Literature Review

The bank for international settlements (BIS) points out machine learning as one of the main drivers of the ongoing transformation of the financial sector [33]. Machine learning and deep learning have been applied in various financial applications since the 1990s. Hutchinson et al. (1994) [25] show that neural networks could be useful in pricing of derivatives. Ruf and Wang (2020) [34] present an overview of additional 150 papers on ANNs for pricing and hedging derivatives. Hung et al. (2020) [23] furthermore show that ANNs have been extensively applied to stock market prediction, stock trading, portfolio management, exchange rate prediction, macroeconomic prediction and credit and default risk. Zhu et al. (2019) [42] highlight the potential of machine learning and neural networks, particularly in xVA calculations.

Current development of machine learning methods in finance challenges current standards of regulation and governance [33]. The frameworks for the governance of machine learning models are still regarded as being in an early stage of development. In April 2021, the European Commission issued a proposal for a regulation specifically targeting the use of machine learning across all sectors. Other attempts to form international principles related to machine learning include the Organisation for Economic Cooperation and Development (OECD) AI Principles and the G20 AI Principles. All publications aim to reduce the risks of machine learning by consolidating common principles of reliability, accountability, transparency, fairness and ethics [33].

There exist no international standards or regulations that apply specifically for the usage of machine learning in the financial industry today. Governance of machine learning should however be in line with the current requirements on traditional models, stated for instance in the Basel Core Principles

(BCP) and Insurance Core Principles (ICP). Among the principles in the BCP and ICP, the usage of machine learning in the financial industry is particularly challenged by requirements of transparency, reliability and accountability. Transparency may suffer as many machine learning models are regarded as 'black box' models, where it is difficult to derive how the input data is used in producing the results. Issues in accountability may arise from difficulties of determining who should be responsible for the results and performance of the model. The reliability of a machine learning model differs from a traditional model as the performance is highly dependent on the quality and relevance of the data. Furthermore, machine learning models may require regular updates and learn from new data to stay up to date, which may in current frameworks be regarded as changes that require supervisory re-approval. For these reasons, financial regulators are in the process of updating the current regulatory frameworks to consider the specific challenges of machine learning governance [33].

A great challenge of machine learning in finance is that of finding enough and representative data. Ferguson and Green (2018) [13] use an artificial neural net to value derivatives. They address the issue of insufficient data by proposing a method to generate realistic data using Monte Carlo simulations. Their results show that it is beneficial to have a larger training set with more Monte Carlo noise. Borchani et al. (2019) [6] use a similar approach to synthetically generate data.

Artificial neural networks have to some extent been applied to xVA. Crépey et al. (2019) [2] suggest a neural network based forward/backward stochastic differential equations (SDE) solver for efficient computation of xVA. Gnoatto et al. (2021) [15] present another deep backward SDE solver that enables high-dimensional computation of xVA. She and Grecu (2018) [36] moreover apply the backward SDE solver to calculate CVA/DVA. Welack (2019) [38] introduces an ANN approach to get market and trade data from expected exposure. Welack further presents an ANN that successfully reproduces a Monte Carlo generated expected exposure profile of an interest rate swap.

1.3 Objective

The aim of this project is to contribute to current research on machine learning applied to xVA. The paper will focus on the potential benefits of machine learning methods over traditional numerical methods with regards to their increased efficiency. The objective is made concrete in the development of a proof of concept model based on artificial neural networks that maps market data to the expected credit risk exposure for an interest rate swap. Such a model has a natural area of use in swift single contract exposure comparisons, where traditional numerical methods may be regarded as unnecessarily costly.

By contributing to the discussion on potential benefits and limitations of applied machine learning in the stated context, the project aims to contribute with knowledge that may lead to the development of new data-driven methods. The conclusions and proposed framework may also be extended to provide insight into other areas of finance and predictive time series modelling. The problem formulation can be summarized in the following research questions:

- What are the potential benefits and limitations of a machine learning approach in exposure calculations with regards to computational efficiency and accuracy?
- What are the requirements for such a method to be regarded as feasible in a practical context?
- What is the outlook on machine learning methods in the xVA-area?

1.4 Procedural Overview

The theoretical framework for the study is presented in the next section. To set the scene, the over-the-counter derivatives market is briefly described and various Valuation Adjustments are explained. It is demonstrated that the exposure profile of a derivative plays an important role in calculating several Valuation Adjustments. The following sections within the theoretical framework define the building blocks in the data generation approach. In order to calculate the expected exposure, the value of the contract is required. Interest rate swaps are therefore introduced and the valuation methodology defined. A section about the Hull-White One-Factor model describes how the short rates are simulated from

market data, and a section about the Monte Carlo method describes stochastic inference. A section about neural networks is included to provide a background and to explain the developed model.

The methodology is described in section 3. The contract is specified and the data generation procedure is properly explained, including all parameter settings. The model development approach and the measure to evaluate performance of the model is defined. The results are presented in section 4, which is followed by a section with a discussion and conclusions. An overview of the workflow is described in Figure 1.1 below.

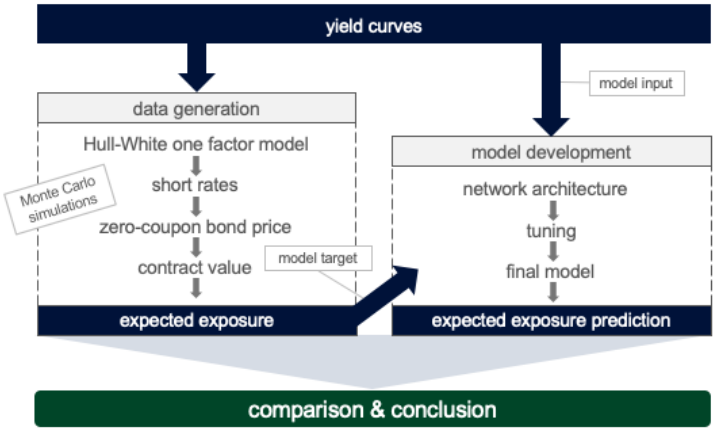


Figure 1.1: Schematic diagram of workflow. The data set consisted of 1,540 observed yield curves and 1,540 synthesized yield curves. Exposure data was generated with a traditional and computationally heavy method. The yield data, Hull-White parameters and deviation from at the money strike rate was used as model input and the generated exposure data was used as model target. The data-driven model was developed and the novel method was then compared to the traditional method.

Chapter 2

Theoretical Framework

2.1 The OTC Derivatives Market

Over-the-counter (OTC) derivatives are financial contracts that are arranged directly between counterparties instead of being traded on an exchange. A derivative is a contractual agreement in which involved parties are bound to deliver payments and obligations that are based on the value of an underlying asset. OTC derivatives offer solutions to arrange specialized contracts that may be used to suit specific hedging needs, to trade large amounts outside the public exchange, or to access liquidity in an alternative channel [22]. Some examples of common OTC derivatives are swaps, forwards, swaptions and exotic options.

The customers of OTC derivatives are commonly hedge funds, corporations or institutional investors. The trades are often executed through large institutions who act as dealers and in turn hedge their own risk, either in the interdealer market or the exchange traded market. The institutional dealers may also engage in their own trades or act as market makers on the OTC market [22].

Statistics of total outstanding OTC derivative positions are published each year by BIS. They reported in [4] that the notional outstanding totaled $610 trillion in June 2021, which is an increase from $607 trillion in June 2020. The size of the OTC market has been fairly stable since 2008, with its largest total notional outstanding of $707 trillion in June 2011. Interest rate derivatives constitute the largest part of the OTC

market, with a reported fraction of 80% of the total notional outstanding in June 2021. The corresponding fraction for foreign exchange derivatives was 16.7% [4].

The legal terms of OTC derivatives trading are commonly established in an International Swaps and Derivatives Association (ISDA) master agreement, which is a template published by the ISDA. The purpose of the ISDA master agreement is to provide legal and risk protection for the counterparties of the trade. If an institution is involved in several contracts with the same counterparty, the ISDA master agreement may allow netting of positions. Netting is a settlement method where the outstanding transactions are consolidated to a net value, which may reduce the exposure to the counterparty [22]. The terms for collateralization are clarified in the Credit Support Annex (CSA) which may voluntarily be included as one of the annexes in the ISDA master contract. The CSA may state liabilities between counterparties to post collateral in order to secure the derivative. Collateralization reduces the credit risk but does not remove it completely [18].

2.1.1 Counterparty Credit Risk

In contract agreements involving payments between two counterparties, counterparty credit risk refers to the risk that one of the parties will fail to fulfill their obligations. Counterparty credit risk is present in loans, bonds, derivatives contracts and other financial investment products where there is a risk of insolvency of the counterparty [18].

In trades that involve exposure of one counterparty to another, counterparty credit risk is always present. The Exposure at Default (EAD) is the value that would be the loss that one party would face in the event of the other's default. The exposure is only non-zero when the value V of any contract from the investors perspective is positive. The EAD is therefore computed in accordance with equation 2.1 [18].

$$EAD = \max(V, 0) \tag{2.1}$$

The Expected Positive Exposure (EPE) is obtained as the expectation of the exposure at default at a given time t in the future,

$$EPE(t) = \mathbb{E}[\max(V, 0)|\mathcal{F}_t]. \tag{2.2}$$

A corresponding Expected Negative Exposure (ENE) is obtained as the

expectation of the positive parts of the negative value,

$$ENE(t) = \mathbb{E}[\max(-V, 0)|\mathcal{F}_t]. \tag{2.3}$$

The exposure profile over some time interval can then defined as the expected exposures in equation 2.2 as a function of time t [18].

2.1.2 Wrong Way and Right Way Risk

One may note that counterparty credit risk is dependent both on market risk and credit risk. The market risk is dependent on the value of outstanding transactions and the credit risk is dependent on the probability of default of the counterparty. Wrong-way and right way risk are important concepts related to the interplay of market risk and credit risk. By a definition given by ISDA, wrong way risk is present when the credit quality of the counterparty is adversely correlated to the exposure. This implies that the probability of default increases with the credit exposure. Right way risk is present when the credit quality of the counterparty is favorably correlated to the credit exposure [18].

2.1.3 Basel III

The OTC derivatives market has been blamed for amplifying and triggering the GFC. The crisis led to many changes in regulatory requirements with a great impact on Valuation Adjustments [19]. The global standard for regulation and supervision of banks is set by the Basel Committee on Banking Supervision (BCBS) [3]. The committee was initialized by the G10 after the collapse of the Bretton Woods exchange rate system (1973), which in turn caused the collapse of two major banks. BCBS presented a set of standards in the *Basel Framework*, which has been adopted by many countries and regions, with the aim to decrease financial bubbles and make financial crises more rare. The framework has been updated several times. After the GFC, BCBS made significant changes to the framework, referred to as Basel III [18]. A major regulatory change for banks presented in Basel III is strengthening of capital bases in banks, including an introduction of a CVA capital charge and leverage ratio. Another major change presented in Basel III is strengthening of liquidity standards, including liquidity coverage ratio and net stable funding ratio. The changes made institutions obliged to put more emphasis on several Valuation Adjustments [19].

2.2 Valuation Adjustments (xVA)

Prior to the GFC, the general framework for valuation of derivative contracts was more simple than it is today. Valuation of derivatives was often performed by analyzing cash flows and applying a suitable discount factor. Derivatives were for a long time valued using single discounting curves based on three-month interbank offered rates, such as three-month LIBOR. The xIBOR curve was seen as a proxy for a unique, risk-free rate and was assumed to accurately represent the bank's cost of funding. The xIBOR based models were thus typically used both for secured and unsecured trades, regardless of their different funding requirements. Furthermore, even risky OTC derivatives were traded with very little consideration to the possible impact of credit risk [18].

At the beginning of the GFC in 2007, many larger banks had introduced Credit Valuation Adjustments to the prices of derivatives. Costs of credit risk, funding and capital were often priced by banks to some degree but did in most cases not have a definite impact on trading activity. The centralization of the OTC market, where a small number of large institutions trade with a large number of counterparties, was seldom regarded as a problem prior to the crisis. Large institutions were generally considered to be "too big to fail" and many banks naturally used unilateral CVA models which only considered the credit risk of the counterparty. The financial crisis exposed several weaknesses in the above assumptions and eventually led to an increased demand for methods for correctly assessing the costs associated with credit risk, funding and capital requirements [19].

The notion that large institutions were "too big to fail" was proven wrong as several banks collapsed, among them Lehman Brothers in 2008. Banks experienced larger CDS spreads and it was evident that their own credit risk should be considered in pricing. Institutions that used bilateral CVA models received large Debit Valuation Adjustment (DVA) benefits, which increased pressure on the development and usage of bilateral models. It was furthermore made evident that the idea of using the same discounting curves for collateralized and uncollateralized derivatives had several flaws. In collateralized trades, a CSA defines the agreed upon provision of collateral. The counterparty that receives collateral generally pays interest at an overnight funding rate which is typically approximated by the overnight index swap (OIS). The spread between OIS and three month

xIBOR was often small prior to the GFC, and was thus not regarded as significant in valuation. When the spreads increased greatly during the crisis, banks introduced OIS-based discounting for collateralized trades. The increased spread also showed that xIBOR should not be regarded as a good proxy for the risk-free rate. Furthermore, three-month xIBOR could no longer be regarded to capture banks' funding costs accurately. The costs of funding increased as many banks experienced larger CDS spreads, which exposed the importance of correctly assessing the price of funding.

Since the crisis, increased awareness and regulation has driven the development of Valuation Adjustments to the base value of derivatives. Today, the valuation of derivative contracts is a rigorous task often performed by two desks at a bank. A risk-neutral base value is assessed by the trading desk specialized in the relevant asset, whereas the different Valuation Adjustment calculations are performed at the xVA desk. The Valuation Adjustment is then added on top of the base value of the contract [19].

An xVA is generally defined as an adjustment to the price of a derivative. A large number of different adjustments exist, each motivated by implied costs and benefits that arise on the derivatives market. Today, the adjustments play a significant role in pricing, accounting, hedging and profit and loss analysis. CVA and Funding Valuation Adjustments (FVA) could be considered as traditional Valuation Adjustments and account for costs of credit risk and funding [19]. Other common Valuation Adjustments include Capital Valuation Adjustments (KVA), Tax Valuation Adjustments (TVA), Margin Valuation Adjustments (MVA) and Collateral Valuation Adjustments (ColVA). The most commonly adopted adjustments are briefly described and motivated in subsequent sections.

2.2.1 CVA/DVA

Credit Valuation Adjustment reflects the market price of credit risk. CVA constitutes a cost for the bank which originates from the expected loss due to counterparty default. The Valuation Adjustment can thus be defined as the difference between the risk-free portfolio value and the credit risky value,

$$CVA = V(\text{default free}) - V(\text{credit risky}).$$

The CVA is computed on a portfolio-level for each counterparty, taking possible netting effects into account. Models to assess CVA may differ in their ways of treating the credit risk of both counterparties in a trade. Unilateral CVA models only consider the credit risk of one counterparty, whereas bilateral CVA models assume that both counterparties have a probability of default [18]. The definition generally holds for both unilateral and bilateral CVA.

Bilateral CVA models introduce a DVA term which reflects the dealer's own credit risk. The CVA is a cost, whereas DVA is a benefit which increases the accounting value. The CVA of one counterparty should thus be constructed to reflect the DVA of the other, and vice versa. It should be noted that this relationship only holds in theory due to differences in accounting practice and CVA models between institutions [18].

2.2.2 FVA

Funding costs and benefits may arise when an institution enters into uncollateralized and not fully collateralized contracts. Funding Valuation Adjustments represent those costs and benefits which are similarly to CVA and DVA sometimes regarded separately [18]. FVA includes net stable funding ratio and liquidity coverage ratio, both regulated within the Basel III framework [19].

Assume that a bank enters into an uncollateralized derivative contract with a corporate counterparty. Hedging the trade with a fully collateralized trade on the interbank market will bring costs and benefits to the bank that depend on how the value of the contract evolves over time. When the unsecured trade has a positive value to the bank, the bank will be required to post collateral on the secured contract which constitutes a negative exposure. Funding the collateral is associated with a cost which would not be present if the trade with the corporate counterparty was secured under a CSA. The CSA can allow rehypothecation, which allows parties to offset funding positions without having to enter into additional loans. In the case where the value of the unsecured contract is negative to the bank, the bank receives collateral from the offsetting position, thus a funding benefit [18]. From this argument it may be understood that any unsecured derivative should be priced with consideration to the associated funding costs and benefits [18].

2.2.3 KVA, MVA, TVA and ColVA

The GFC led to large increase in the capital requirements for banks and thus an increased demand for proper assessments of the implications of holding capital [18]. Capital Valuation Adjustments reflect the cost of holding regulatory capital during the lifetime of a trade [18]. KVA includes CVA capital charge and leverage ratio, both regulated in the Basel III framework [19].

Tax Valuation Adjustments reflect the costs that may arise due to taxation when a bank reports profits and losses. Profits and losses may be present when an institution warehouses counterparty credit risk. The profits are taxable whereas any losses gives the institution future tax benefits [18].

Margin Valuation Adjustments reflects the funding cost of the initial margin that is posted for a particular contract. MVA is sometimes included in valuation along with FVA, but it is not standard valuation practice [19].

Collateral Valuation Adjustment captures costs that arise from differences in valuation and remuneration rates for collateral transactions. These differences may for example be present when transactions in one currency are collateralized in another or when the margin agreement otherwise specifies a spread between valuation and margin remuneration rates. The spread may be handled either by changing the discount rate or including ColVA as an additional cost [19].

2.2.4 Implementation

Given the large scope of the concept, interpretation and implementation of xVA may vary slightly between institutions. It is common that the computation of Valuation Adjustments requires many risk measures and quantities that have to be regularly assessed and updated. Large amounts of market and portfolio data, complex mathematical models and advanced software is required in the procedure. Furthermore, each model is associated with their own assumptions and parameters which have to be tuned and calibrated thoroughly.

Counterparty credit exposure is an important risk measure that has to be evaluated for several Valuation Adjustments. Counterparty credit

exposures have to be computed on a daily level in order to control that they lie within certain specified thresholds set by the institution. The exposures are often computed by computationally intensive Monte Carlo simulations of portfolio scenarios. Each exposure is based on thousands of Monte Carlo simulations of the portfolio value on a time grid that is typically discretized in hundreds of time steps. Furthermore, banks typically have exposures to thousands of derivative counterparties. The xVA desk thus has to handle very large amounts of data and continuously analyze limits as well as changes in the obtained exposures.

Credit exposures are natural constituents of all CVA models. The CVA term is always obtained as an integral over the Expected Positive Exposure profile and, in bilateral models, the DVA term is an integral over the Expected Negative Exposure profile. Assume a contract with value V and maturity time T. The counterparty is assumed to default at time τ upon which the recovery rate is R. If $\mathbb{1}$ denotes the indicator function and $\mathcal{G}_t$ is the filtration up to time t, unilateral CVA can be defined as an expectation,

$$CVA = \mathbb{E}[\mathbb{1}(\tau \leq T)\max(V(t,\tau),0)(1-R)|\mathcal{G}_t].$$

We will now assume that there is no wrong way or right way risk, which implies an independence between credit risk and market risk factors. Expectations over credit and market state variables may thus treated separately and the unilateral CVA can be expressed as an integral [18],

$$CVA(t) = \int_{s=t}^{T} \lambda_C(s) e^{\int_{u=0}^{s} -\lambda_C(u)du} \mathbb{E}[e^{\int_{u=0}^{s} -r(u)du}(1-R)EAD(s)|\mathcal{F}_s]ds,$$

where λ_C(s) denotes an instantaneous default probability and $\mathcal{F}_s$ is the filtration that contains the information of the market state variables up to time s. We may define a risky discount factor, $\mathrm{D}_{r+\lambda_C} = e^{\int_{u=0}^{s} -(\lambda_C(u)+r(u))}du$ and simplify the expression,

$$CVA(t) = (1-R)\int_{s=t}^{T} \lambda_C(s) D_{r+\lambda_C} EPE(s)ds.$$

The integral may be approximated with a sum [18],

$$CVA(t) = (1-R)\sum_{i=0}^{n-1} PD(t_i, t_{i+1})EPE^*(t_i),$$

where EPE^* denotes the discounted Expected Positive Exposure and $PD(t_i, t_{i+1})$ is the default probability in the interval $[t_i, t_{i+1}]$. Thus, unilateral CVA can be represented by a weighted average of the EPE profile. The EPE is often obtained in a risk-neutral measure, and explicit discount factors are thus generally not needed. The interpretation of the above formula is that CVA roughly can be seen as the product of EPE and the probability of default. CVA can be calculated path-wise using the discrete sum. It is a straightforward but computationally heavy and time consuming approach [19].

Bilateral CVA is motivated by the need to consider the credit risk of both the counterparty and the institution. In a simple form, bilateral CVA can be represented as a cost term (CVA) and a benefit term (DVA),

$$BCVA = CVA + DVA.$$

The CVA and DVA are obtained as integrals over the positive and negative expected exposures respectively. The suffixes "C" and "I" indicate that recovery rates, instantaneous default probabilities and discount factors are now considered for both sides of the trade,

$$CVA(t) = (1 - R_C) \int_{s=t}^{T} \lambda_C(s) D_{r+\lambda_C+\lambda_I} EPE(s) ds,$$

$$DVA(t) = (1 - R_I) \int_{s=t}^{T} \lambda_P(s) D_{r+\lambda_C+\lambda_I} ENE(s) ds.$$

Similarly to unilateral CVA, bilateral CVA can be represented as weighted sums of credit exposures,

$$CVA(t) = (1 - R_C) \sum_{i=0}^{n-1} EPE^*(t_i) PD_C(t_i, t_{i+1})[1 - PD_I(t_i, t_{i+1})],$$

$$DVA(t) = (1 - R_I) \sum_{i=0}^{n-1} ENE^*(t_i) PD_I(t_i, t_{i+1})[1 - PD_C(t_i, t_{i+1})].$$

Note that the sums that represent bilateral CVA include own default probability. The CVA term in bilateral CVA thus differs from the unilateral CVA in which only considers the default probability of the counterparty [19].

FVA is similarly to CVA often calculated in exposure based frameworks. Funding costs arise from in-the-money contracts and Expected Positive

Exposures (EPE) whereas funding benefits arise from out-of-the-money (OTM) contracts and Expected Negative Exposures (ENE). Assessing FVA can be done in numerous ways depending on convention and the premises for funding. We may show a simple example of an exposure based framework for FVA by regarding a symmetric FVA formula which can be expressed as

$$FVA = \sum_{i=1}^{m} EFV^*(t_i)FS(t_{i-1}, t_i)(t_i - t_{i-1}).$$

Note the similarities to the CVA formula. EFV, which is the discounted expected value which has to be funded has replaced EPE, and FS which is the funding spread has replaced the counterparty default probability of the CVA formula [19]. The EFV may be seen as the sum of the expected positive and negative exposures, and the equation for FVA may therefore be seen as a combination of funding costs and funding benefits FCA and FBA

$$FVA = FCA + FBA,$$

where FCA and FBA is defined as

$$FCA = \sum_{i=1}^{m} EPE^*(t_i)FS(t_{i-1}, t_i)(t_i - t_{i-1}),$$

$$FBA = \sum_{i=1}^{m} ENE^*(t_i)FS(t_{i-1}, t_i)(t_i - t_{i-1}).$$

Note how, similarly to CVA, the FVA can be represented as a weighted sum over exposure profiles.

There is a generally agreed upon overlap between funding benefits (FBA) and DVA as defined in a bilateral CVA framework. Both the DVA and the FBA are defined as benefits that arise from negative exposures. Many models therefore assess CVA and FVA together, in order to properly evaluate the benefits of negative exposures without double counting [19]. In a study performed by Deloitte in 2013, it is reported that frameworks based on CVA and symmetric funding, which include CVA, FCA and FBA, were the most common among market practitioners [9]. Another common framework assumes bilateral CVA and symmetric funding and thus includes CVA, DVA and FCA.

It is clear that counterparty credit exposures play a significant role in xVA calculations, especially in CVA and FVA. These are also very common

adjustments among institutions, with well developed frameworks for their implementation. Other adjustments may differ more between institutions, partly because of differences in capital requirements, taxation laws, and accounting practice. KVA is the Valuation Adjustment which differs the most between institutions. Mainly because banks of different size in different regions need to follow different capital requirements, different models are used in different banks and finally banks have different view on return on capital [19].

2.3 Interest Rate Swaps

An interest rate swap (IRS) is an OTC derivative where a fixed interest rate typically is exchanged for a floating interest rate on the same notional principal. The notional principal is the amount on which the derivative is contracted on. The parties of the contract do not usually exchange notional principal, instead only streams of future interest payments are exchanged. The payment stream corresponding to the fixed rate is referred to as the fixed leg of the swap and the payment stream corresponding to the floating rate is referred to as the floating leg [24]. In this section, the valuation of a plain vanilla IRS where the interest rate is set in advance and pays in arrears (later) is derived.

2.3.1 Assumptions

In order to value the contract, some assumptions about the bond market are required. Let the price of a zero-coupon bond with maturity T at time t be $p(t, T)$. As a means to avoid arbitrage, it is assumed that $p(t, t) = 1$ for all t. It is further assumed that there is a market for zero-coupon bonds for all maturities $T > 0$ and that there are no transaction costs involved. The zero-coupon bond price $p(t, T)$ is furthermore assumed to be stochastic with variables t and T [5].

A normal bid-ask spread for an IRS is assumed to be between 0.1 and 0.2 basis points (BPS) under the condition that there is a liquid currency and the market is calm.

2.3.2 Valuation

There are two basic models for valuing interest rate swaps. One model interprets the IRS as the difference between two bonds, whilst the

other model interprets the IRS as a replicating portfolio of forward rate agreements [24]. Here, the former approach is applied.

For the *pay fixed receive floating* party, or the *receiver* party of a swap agreement, the swap is equivalent to having a long position in a floating-rate bond and a short position in a fixed-rate bond. The value of the IRS at time t can then be described as the difference between the value of the two bonds, $V(t) = V_{fl}(t) - V_{fix}(t)$ for all remaining payments after t [24]. The IRS is initialized at T_0 and the dates $T_0, T_1, ..., T_n$ are equally spaced on an interval $I \in [T_0, T_n]$. The fixed leg of the swap with payments at T_i, $i = 1, ..., n$ can be replicated by a portfolio of zero-coupon bonds with maturities T_i. The time between any two payments is given by $\delta_i^{fix} = T_i - T_{i-1}$. The notional principal which the IRS is based on is denoted N and the fixed strike rate R. At every T_i the amount paid is $N\delta_i^{fix}R$.

To get the present value of the swap, all future cash flows need to be discounted. The discounting factor at $t < T_i$ of the cash flow at T_i is denoted D_i and it is equivalent to the present value of receiving \$1 at the future date T_i. It can thus be obtained from the zero-coupon bond price at t with maturity corresponding to the time of the payment T_i, that is $D_i = p(t, T_i)$. The present value at t of the fixed leg is given by the sum of all future discounted cash flows in accordance with equation 2.4.

$$V_{fix}(t) = RN \sum_{i=1}^{n} \delta_i^{fix} D_i \theta_i, \qquad \theta_i = \begin{cases} 1 & \text{if} \quad t \leq T_i \\ 0 & \text{if} \quad t > T_i \end{cases} \tag{2.4}$$

The floating leg of the swap with payments received at T_j, $j = 1, ..., m$ can be replicated by a self-financing portfolio [5]. Let the dates $T_0, T_1, ..., T_m$ be equally spaced, the time between any two payments $\delta_j^{fl} = T_j - T_{j-1}$ and the notional principal the same N as above. The floating-rate is the spot LIBOR rate $L(T_{j-1}, T_j)$. The rate is set at T_{j-1} and paid at T_j. It is by definition [5] given by

$$L(T_{j-1}, T_j) = -\frac{p(T_{j-1}, T_j) - 1}{\delta_j^{fl} p(T_{j-1}, T_j)}.$$

The amount received at T_j is $N\delta_j^{fl} L(T_{j-1}, T_j)$. The simple forward rate, or the LIBOR forward rate, contracted at t and received at T_j is used to value the contract at time t. The definition [5] of the LIBOR forward

rate $F_j(t)$ is [5],

$$F_j(t) = L(t; T_{j-1}, T_j) = -\frac{p(t,T_j) - p(t,T_{j-1})}{\delta_j^{fl} p(t,T_j)}. \tag{2.5}$$

The present value of the floating leg at t is given by equation 2.6, summing up all future discounted cash flows.

$$V_{fl}(t) = N \sum_{j=1}^{m} F_j \delta_j^{fl} D_j \theta_j, \qquad \theta_j = \begin{cases} 1 & \text{if} \quad t \leq T_j \\ 0 & \text{if} \quad t > T_j \end{cases} \tag{2.6}$$

When valuing the floating leg at time t^* between any two payments T_{j-1} and T_j, the LIBOR forward rate corresponding to payment j can be approximated by the LIBOR spot rate,

$$F_j^*(t^*) \approx L(T_{j-1}, T_j).$$

The value V_{fl} between two floating payments at $t^* \in]T_{j-1}, T_j[$ can then be simplified as follows

$$\begin{aligned} V_{fl}(t^*) &= N\left(L(T_{j-1},T_j)\delta_j^{fl} D_j + \sum_{k=j+1}^{n} L(t^*; T_{k-1}, T_k)\delta_k D_k \right) \\ &= N\left(p(t^*,T_j)\Big(\frac{1}{p(T_{j-1},T_j)} - 1\Big) + \sum_{k=j+1}^{n} \Big(p(t^*,T_{k-1}) - p(t^*,T_k)\Big) \right) \\ &= N\left(\frac{p(t^*,T_j)}{p(T_{j-1},T_j)} - p(t^*,T_n) \right). \end{aligned}$$

The value of the floating leg at $t = 0$ can further be obtained from equation 2.6:

$$V_{fl}(0) = N \sum_{j=1}^{m} F_j(0)\delta_j^{fl} p(0,T_j) = N \sum_{j=1}^{m} \big(p(0,T_{j-1}) - p(0,T_j)\big) = N\big(1 - p(0,T_n)\big).$$

If the swap is considered at the money (ATM), the strike rate R is determined such that it is fair at initialization. It means that at $t = 0$, the value of the fixed interest rate cash flows and the value of the floating interest rate cash flows are equal [5]. Inserting the expression for the floating leg value at $t = 0$ above and the fixed leg value at $t = 0$ gives the strike rate,

$$V_{fl}(0) = V_{fix}(0) \implies R = \frac{1 - p(0,T_n)}{\sum_{i=1}^{n} \delta_i^{fix} D_i}. \tag{2.7}$$

Once the zero-coupon bond prices $p(t,T)$ are known for every point in time $t \in \{[0,T_n] \cup [0,T_m]\}$ and maturities T corresponding to the payments, the risk neutral value of the IRS at time t can be obtained. The IRS is valued by computing the difference between the value of the floating and fixed leg for all remaining payments after t. Using equation 2.4-2.6, the contract value V can be obtained. The expression for the contract value is properly demonstrated in equation 2.8.

$$V(t) = N\left(\sum_{j=1}^{m} F_j \delta_j^{fl} D_j \theta_j - R\sum_{i=1}^{n} \delta_i^{fix} D_i \theta_i\right) \tag{2.8}$$

The LIBOR forward rate F_i obtained from equation 2.5 and the strike rate R is obtained by applying equation 2.7. θ_i and θ_j represent the step functions defined in equation 2.4-2.6. The discount factors are $D_i = p(t,T_i)$ and $D_j = p(t,T_j)$. The notional principal N is solely a scaling factor.

2.4 Short Rate Models

A short rate model describes how interest rates evolve through time and can be used to calculate bond prices $p(t,T)$. The interest rate depends on how the short rate evolves through time, thus it can be obtained once the process of the short rate is determined [24].

The short rate r_t, or the *instantaneous short rate*, is the interest rate for an infinitesimally short period specifically at time t. The *instantaneous forward rate* $f(t,T)$ is the corresponding forward rate for an infinitesimally short period. Given that $p(t,T)$ is the observed price at time $t \in [0,T]$ of a zero-coupon bond maturing at T, then according to Björk [5], the instantaneous forward and short rate can be defined as

$$f(t,T) = -\frac{\partial \ln p(t,T)}{\partial T},$$

$$r_t = f(t,t).$$

An Itô process is used in most short rate models to model the risk-neutral process for the instantaneous short rate. The drift of the short rate is denoted m and the instantaneous standard deviation by s. The process for the short rate under the risk-neutral measure $\mathbb{Q}$ can then be described by the following equation [24]

$$dr = m(r)dt + s(r)dW(t). \tag{2.9}$$

Equilibrium models and no-arbitrage models are two different types of short rate models. The former include models such as the Vasicek Model (1977) and the Cox, Ingersoll and Ross Model (1985). These models assume that the instantaneous drift m of the short rate and the standard deviation s are time independent as in equation 2.9, hence providing an approximate fit to today's term structures of interest rates [24].

The latter no-arbitrage models can provide an exact fit to today's term structures. The difference is that the drift m generally is time dependent for these no-arbitrage models. Examples of such models are the Hull-White One-Factor Model (1990) which extends the Vasicek Model, the Hull-White Two-Factor Model (1994) which gives a richer pattern of the term structure and volatility than the One-Factor version, and the Hoo-Lee Model (1986) [24]. Hull (2012) emphasizes that one should use a model with more than one factor when hedging whilst the standard procedure when pricing interest rate derivatives is to use a one-factor model [17].

2.4.1 Affine Term Structure

In an arbitrage free setting, the family of zero-coupon bonds $\{p(.,T); T > 0\}$ is assumed to be dependent on the short rate r_t through the relation,

$$p(t,T) = F(t, r_t; T).$$

It is assumed that F is a smooth function of the evaluation time t, the corresponding short rate r_t and the time to maturity T. In risk-neutral valuation, bond prices are obtained through

$$F(t, r_t; T) = E^Q_{t,r}[e^{\int_t^T r_s ds}].$$

The $\mathbb{Q}$-dynamics of the short rate are given by equation 2.9. In the case where $F(t, r_t; T)$ has the form

$$F(t, r_t; T) = A(t,T)e^{-B(t,T)r_t}, \tag{2.10}$$

the model is said to have an affine term structure. In a model that possesses an affine term structure, bond prices can be obtained by evaluating equation 2.10 for its associated functions A(t,T) and B(t,T) [5].

2.4.2 The Hull-White One-Factor Model

In the Hull-White One-Factor model, the following $\mathbb{Q}$-dynamics of the short rate are assumed,

$$dr_t = \{\theta(t) - ar_t\}dt + \sigma dW_t^{\mathbb{Q}}, \tag{2.11}$$

where $W^{\mathbb{Q}}$ is a Brownian motion under the risk-neutral measure $\mathbb{Q}$, in which prices are adjusted for risk aversion in the market. The model introduces two constants, the mean reversion a and the volatility σ. The time dependent and deterministic function $\theta(t)$ is defined as,

$$\theta(t) = \frac{d}{dt}f(0,t) + af(0,t) + \frac{\sigma^2}{2a}(1 - e^{-2at}),$$

where $f(0,t)$ denotes the observed instantaneous forward rate which allows for calibration to the current term structure of interest rates. The volatility structure of the model is commonly set by calibrating a and σ to observed market data of caps and swap options [24].

The Hull-White One-Factor model assumes a mean reverting behavior, where the short rates will revert back to their long term mean $\frac{\theta(t)}{a}$. This is a common property among short rate models since rates are expected to stay within certain limits [24].

Using Itôs lemma, the solution to the stochastic differential equation in 2.11 can be obtained for $0 \leq s \leq t \leq T$ as,

$$r(t) = e^{-a(t-s)}r(s) + \int_s^t e^{-a(u-t)}\theta(u)du + \sigma\int_s^t e^{-a(t-u)}dWu.$$

Conditionally on $\mathcal{F}_s$, the short rates are Gaussian distributed with expectation and variance given by,

$$\mathbb{E}[r_t|\mathcal{F}_s] = e^{-a(t-s)}r(s) + \int_s^t e^{-a(u-t)}\theta(u)du,$$

$$Var[r_t|\mathcal{F}_s] = \frac{\sigma^2}{2a}\left(1 - e^{-2a(t-s)}\right).$$

Due to the conditional distribution of the short rates, simulated paths can be obtained by using a step-wise stochastic simulation method from a defined starting point.

Zero-coupon bond prices in the Hull-White One-Factor model can be obtained by exploiting the affine term structure. The bond prices $p(t,T)$ at time t are then given by equation 2.10, where A(t,T) and B(t,T) are defined as,

$$B(t,T) = \frac{1}{a}\{1 - e^{-a(T-t)}\} \tag{2.12}$$

$$\ln A(t,T) = ln\frac{p(0,T)}{p(0,t)} + B(t,T)f(0,t) - \frac{1}{4a^3}\sigma^2(e^{-aT} - e^{-at})^2(e^{2at} - 1). \tag{2.13}$$

2.4.3 Change of Numeraire in the Hull-White Framework

So far we have defined the short rate in the risk-neutral measure $\mathbb{Q}$, in which the bank account B is the numeraire. The forward T-measure $\mathbb{Q}^T$ is defined as the martingale measure where the numeraire process is a zero-coupon bond $p(t,T)$. A change of numeraire from the risk-neutral measure $\mathbb{Q}$ to the T-forward measure $\mathbb{Q}^T$ may facilitate computations as it allows for simple discounting with zero-coupon bond price processes $p(t,T)$. The price Π_t is for any contingent claim X with maturity T, or T-claim, given by

$$\Pi_t[X] = p(t,T)E^T[X|\mathcal{F}_t],$$

where E^T denotes integration with respect to $\mathbb{Q}^T$ [5]. In the following derivation a Hull-White One-Factor model framework as defined in the previous section is assumed. The $\mathbb{Q}^T$-Brownian motion W^T is defined [7] by,

$$dW^T = dW^Q + \sigma B(t,T) \tag{2.14}$$

where $W^{\mathbb{Q}}$ is the $\mathbb{Q}$-Brownian motion and $B(t,T)$ is defined by equation 2.12. The short rate dynamics in the T-forward measure are obtained by substituting the dynamics in 2.14 into the short rate dynamics in equation 2.11. Thus,

$$dr_t = \{\theta(t) - ar_t - \sigma^2 B(t,T)\}dt + \sigma dW_t^{\mathbb{Q}^T}. \tag{2.15}$$

The solution to 2.15 implies that short rates obtained in the $\mathbb{Q}$-measure can be transformed to the $\mathbb{Q}^T$-measure by adding a deterministic term $M(s,t)$,

$$r_t^{\mathbb{Q}^T} = r_t^{\mathbb{Q}} + M(s,t),$$

where the correctional term $M(s,t)$ is given by

$$M(s,t) = -\frac{\sigma^2}{a^2}\left(1 - e^{-a(t-s)}\right) + \frac{\sigma^2}{2a^2}\left(e^{-a(T-t)} - e^{-a(T+t-2s)}\right).$$

Short rates are after a change of measure from $\mathbb{Q}$ to $\mathbb{Q}^T$ still, conditionally on $\mathcal{F}_s$, Gaussian distributed with expectation and variance given by,

$$\mathbb{E}[r_t|\mathcal{F}_s] = e^{-a(t-s)}r(s) + \int_s^t e^{-a(u-t)}\theta(u)du + M(s,t), \tag{2.16}$$

$$Var[r_t|\mathcal{F}_s] = \frac{\sigma^2}{2a}\left(1 - e^{-2a(t-s)}\right). \tag{2.17}$$

2.5 The Monte Carlo Method

Monte Carlo simulation is a method for statistical estimation of integrals which is performed by repeated evaluations of the integrand at points determined by random draws from a suitable distribution. Let μ denote the expectation of a function h($\mathbf{X}$) of a random variable $\mathbf{X}$. By drawing identically distributed and independent random samples $\mathbf{X}_1, ..., \mathbf{X}_n$ from the probability density f of $\mathbf{X}$ we may estimate the expectation of $h(\mathbf{X})$, $\mathbb{E}[h(\boldsymbol{X})]$ [14]. As $n \to \infty$, the law of large numbers implies for the Monte carlo estimate μ_{MC},

$$\mu_{MC} = \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{X}_i) \to \int h(\mathbf{x})f(\mathbf{x})d\mathbf{x} = \mu.$$

Under the assumption that the variance of the objective function, $\mathbb{V}[h(\mathbf{X})]$ is finite, the central limit theorem implies a normal distribution of estimation error,

$$\sqrt{n}(\mu_{MC} - \mu) \xrightarrow{d.} \mathcal{N}(0, Var[h(\mathbf{X})]).$$

This implies an $\mathcal{O}(n^{-1/2})$ convergence of the integration as,

$$D[\mu_{MC} - \mu] = \sqrt{Var[\mu_{MC} - \mu]} = \sqrt{\frac{Var[h(\mathbf{X})]}{n}} = \frac{D[h(\mathbf{X})]}{\sqrt{n}}.$$

Note that due to the slow convergence, the Monte Carlo method may, depending on chosen accuracy requirements, require a large number of samples n. Monte Carlo integration is thus often a computationally

demanding task where the complexity of the implementation is also highly dependent on the dimensionality of the problem.

An unbiased estimate of the sampling variance is obtained as,

$$Var[\mu_{MC}] = \frac{1}{n-1}\sum_{i=1}^{n}[h(\mathbf{X}_i) - \mu_{MC}]^2$$

The two-sided confidence interval that covers μ with a 1-α probability can be obtained,

$$\mathcal{I}_\alpha = (\mu_n - \lambda_{\alpha/2}\frac{D(\mu_{MC})}{\sqrt{n}},\ \mu_n + \lambda_{\alpha/2}\frac{D(\mu_{MC})}{\sqrt{n}}),$$

where λ_p denotes the p-quantile of the normal distribution.

2.6 Neural Networks

Deep learning, or ANNs, is a machine learning technique and an approach to AI partly inspired by the biological brain. The idea is to learn the mapping from input to output for a particular task. The network uses a series of simple mappings, each mapping known as a layer, to solve a complicated mapping problem. In a regular feedforward neural network, the input data is fed to a visible first layer of the network and then propagated forward through a series of hidden layers and finally to an output layer. Each layer has multiple neurons, or units, that perform tasks simultaneously. The concept is visualized in figure 2.1 below. The layers are called hidden because the model itself determines which concepts are essential for explaining the patterns in the observed data. Similarly to any other data-driven machine learning method, neural networks learn from experience and data [16].

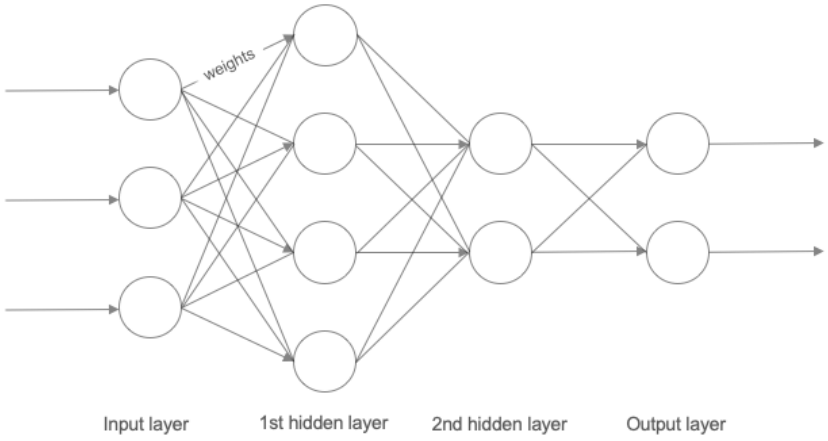


Figure 2.1: Schematic diagram of a feed-forward neural network with two hidden layers. Each node represents a unit and each directed edge represents a connection between the output of one unit to the input of another unit. Each connection has weights, which determines the influence of a node on another [16].

One of the most common acyclic feedforward neural networks is the Multilayer Perceptron (MLP). A forward pass of a MLP with one layer is the process of passing an input vector $\mathbf{x}$ to the input layer and propagating it forward through the hidden layer to the output layer. The hidden layer calculates the weighted sum of the input units and thereafter the activation function is applied to the sum. Nonlinear activation functions are appropriate for complex tasks since they make it possible for a neural network to model nonlinear equations. Two of the most common nonlinear activation functions are the hyperbolic tangent,

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1},$$

and the logistic sigmoid,

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

The hyperbolic tangent take an input x and give a value in the interval $[-1, 1]$ and the logistic sigmoid gives a value in the interval $[0, 1]$. The

output of the hidden layer, the activations, is then the input to the output layer. The output layer operates similar to the hidden layer. It calculates the weighted sum of the activations from the hidden layer and then applies an activation function to get the output vector $\mathbf{y}$ [17].

The idea when training a neural network for a regression task is to minimize the loss. When training a model on noisy data, it is important that the loss function is robust towards outliers. A robust loss function means that outliers have a minor impact on the model [30]. A commonly used loss function $\mathcal{L}$ is the mean squared error (MSE) of targets y and predictions $\hat{y}$,

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (y_i - \hat{y}_i)^2 \tag{2.18}$$

where N is the number of samples. In the training phase, the neural network learns the optimal weights $\mathbf{w}$, or parameters, such that the loss function is minimized for the training data. Training is an extremely important and expensive part of machine learning and ANNs, hence optimization methods have been developed specially for training machine learning and deep learning models [16].

Most deep learning models use some gradient-based optimization method to minimize the loss function $\mathcal{L}$. The weights of the neural network are commonly updated in iterative algorithms that utilize the gradients to minimize the loss function. These algorithms are called gradient descent algorithms [16]. The gradients with respect to the weights therefore need to be calculated. An efficient way to calculate the gradients is to repeatedly apply the chain rule, as suggested by the renowned backpropagation algorithm [17]. After each forward pass of the neural network, the backpropagation performs a backward pass through the network to tune the model's parameters.

Stochastic gradient descent is an extension of gradient descent algorithms that is used in most neural networks. Regular gradient descent may be computationally expensive for large data sets as every sample has to be propagated through the network in order to update the weights. In SGD, the training data set is randomly divided in batches that contain a subset of the whole set. All the weights of the network are then updated once every time a batch has been propagated through the network. Note that the particular case where batch-size equals the size of the whole data set is exactly regular gradient descent. An epoch has been performed

when all the samples have been used for weight updates once. Stochastic gradient descent is typically more efficient than regular gradient descent as the weights are updated several times in every epoch. Furthermore, SGD may be beneficial for avoiding local minima of the loss function [16].

Supervised learning is a type of machine learning and ANN problem where input-target pairs are used for training. Sequence data, in terms of supervised learning, consists of input-target pairs $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ belongs to the input space $\mathcal{X} = \mathbb{R}^{\mathbb{M}}$ and $\mathbf{y}$ belongs to the target space $\mathcal{Y} = \mathbb{R}^{\mathbb{N}}$. The input and target spaces are sets of real-valued vectors of size M and N respectively [17]. The input-target pairs are commonly divided into a *training* set, a *validation* set and a *test* set with samples assumed to be independently drawn from a fixed distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ [17].

The loss is minimized for the training set during training, but the loss is also computed on the validation set to validate the performance and to determine a suitable number of epochs. The trained model is tested on previously unseen data in the test set. The goal when training an ANN is to carry over the performance from the training set to the test set, known as generalization [17]. Discrepancy between performance on training and test data is referred to as overfitting. It means that bias is low at the cost of increased variance, giving a complex and precise model but with considerable variation. Decreasing the variance will increase the bias and the model may be prone to underfitting instead, which means that the model is too simple. The bias-variance tradeoff refers to this challenge of finding the appropriate model complexity [32].

2.6.1 Regularization

Goodfellow et. al (2016) defines regularization as any modification to a learning algorithm that is intended to reduce its generalization error but not its training error. Common regularization strategies include parameter norm penalties, which constrain the parameter values of the model by modifying the objective function. Another common class of regularization methods is ensemble methods, which combine the outputs from several trained models. Choosing a small batch size is another way to achieve a regularizing effect. Batch normalization may also have a regularizing effect.

Parameter norm penalty methods can generally be described by,

$$\tilde{E}(\boldsymbol{\omega}) = E(\boldsymbol{\omega}) + \alpha\Omega(\boldsymbol{\omega}),$$

where Ω is the regularization term and the real number α is the regularization strength. A common regularization term Ω is the L2 norm regularization, defined as,

$$\Omega = \frac{1}{2}\sum_i \omega_i^2,$$

where the sum is obtained over all the weights. Another regularization term is the L1 norm regularization term, also known as lasso,

$$\Omega = \sum_i |\omega_i|.$$

The gradient of the L1-regularization term does not go to zero for small weights, which may force unnecessary weights to zero. Therefore L1-regularization can be used as a method for feature selection [16].

The dropout method introduces regularization by temporarily and randomly omitting nodes in the network. During training, the weights that correspond to a certain node are at random removed along with the node itself. The probability of removing a node is governed by a specified parameter p. The dropout procedure is applied for each sample and only the weights that have not been dropped are updated. The dropout method can thus be seen as a means to generate an ensemble of thinned networks. When the network is used for new data, the output of the ensemble can be approximated by the output of the unthinned network where the weights are multiplied with their probabilities of being included in the network. This result is known as the weight scaling inference rule [16].

Goodfellow et al. (2017) [16] recommend using some form of regularization as long as the data set does not include $>$ 10M samples, it is however highly problem specific. According to Goodfellow et al., the best models are typically large and well regularized.

2.6.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are designed specially for sequence modelling [16]. The connections of the network form cycles that allow the network to store information about the surrounding context of each

value in the sequence. Figure 2 shows a simplified view of an RNN. In addition to the connections that propagate data forward as in an MLP, there are connections between each hidden layer. Content from the previous timestep is thus available to the network, making it possible for an RNN to learn sequential patterns [17]. The universal approximation theory applied to RNNs implies that an RNN with an adequate number of hidden units can approximate any sequence-to-sequence mapping with an arbitrary high accuracy [21].

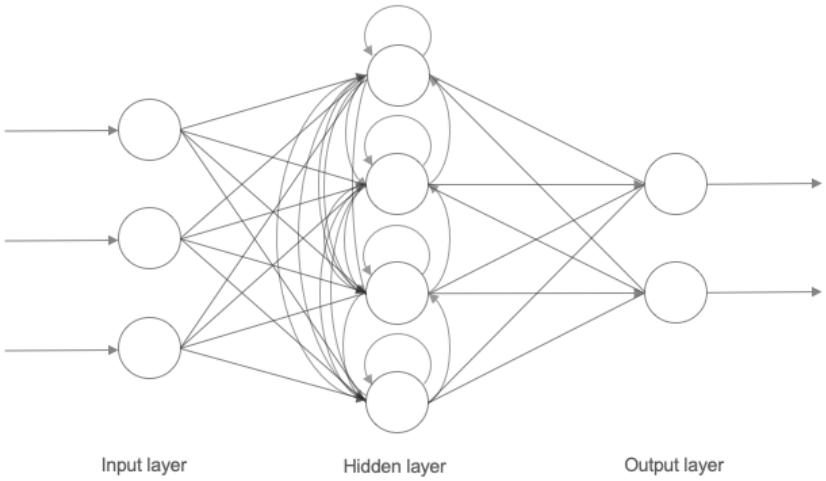


Figure 2.2: Schematic diagram of an RNN with one hidden layer. Each node represents a unit and each directed edge represents a connection between units. Connections from any hidden layer to another hidden layer are recurrent connections [17].

An RNN could be regarded as a feed-forward network with shared weights. The hidden nodes h_t can be seen to constitute a new hidden layer at each timestep t $= 0, ..., T$ and their feedback weights are shared between the layers. This way of regarding a recurrent neural network is called *unfolding in time* and it gives an intuitive representation of how the node outputs are affected by their previous timesteps [16]. The forward pass is the same in RNNs as in feed-forward neural networks, except that the input to the hidden layer also includes the hidden layer activations from the previous timestep. Assume the input vector has T timesteps and x_i^t is the input i at time t. Further assume that the RNN has I input units,

H hidden units and K output units. The weight w_{ih} is the connection from unit i to h. The network input $a_h^{(t)}$ for a particular timestep t and hidden unit h is given by

$$a_h^{(t)} = \sum_{i=1}^{I} w_{ih} x_i^{(t)} + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{(t-1)},$$

where the first term corresponds to the network input to unit h at timestep t and the second term includes the information from all hidden units for the previous timestep [17]. The activation function θ_h for the hidden layer h is then applied to $a_h^{(t)}$, which gives the activation $b_h^{(t)}$ of the hidden unit h at timestep t,

$$b_h^{(t)} = \theta_h(a_h^{(t)}).$$

All hidden activations are obtained by recursively computing $a_h^{(t)}$ and $b_h^{(t)}$, starting from $t = 1$ and iterating through all timesteps. The input $a_k^{(t)}$ to the output units for each timestep is given by

$$a_k^{(t)} = \sum_{h=1}^{H} w_{hk} b_h^{(t)}$$

where w_{hk} is the connection from unit h to k [17].

The derivatives with respect to the weights in RNNs are typically obtained by applying a variant of the classical backpropagation algorithm, namely the Backpropagation Through Time (BPTT) algorithm [16]. The difference between the BPTT and the traditional backpropagation algorithm is that the loss $\mathcal{L}$ on the activation depends on the impact on the hidden layer at the next timestep in addition to the impact on the output [17]. Differentiating $\mathcal{L}$ with respect to the activation $a_j^{(t)}$ at unit j gives the following expression [17],

$$\frac{\partial \mathcal{L}}{\partial a_j^{(t)}} = \theta'(a_j^{(t)}) \Big(\sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial a_k^{(t)}} w_{jk} + \sum_{h=1}^{H} \frac{\partial \mathcal{L}}{\partial a_h^{(t+1)}} w_{jh} \Big).$$

The full sequence of $\mathcal{L}$ differentiated with respect to all timesteps of the activation is obtained by starting at the final timestep and moving through all steps backwards [17]. The weights are shared over all timesteps, hence the derivatives with respect to the weights are given by summing over the weights of all timesteps,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial a_j^{(t)}} \frac{\partial a_j^{(t)}}{\partial w_{ij}}.$$

The BPTT algorithm described above can be used with any gradient-based optimization method [16]. One of the most popular optimization algorithms is the adaptive movements algorithm *Adam* [16]. Adam is especially designed for optimization of large scale problems with large data sets and large amounts of parameters, making it suitable for training neural networks [28].

Traditional RNNs process the input sequence in order which makes the output depend on past context solely. There are however many circumstances in sequence modelling that could be improved by letting the output depend on future context as well. Bidirectional recurrent neural networks (BRNN) targets this issue by reading the input sequence forwards in one hidden layer and backwards in another hidden layer. Both hidden layers are connected to the output layer such that the output layer knows the full, past and future, context [17].

A major issue for conventional RNNs is the vanishing gradient problem and more rarely the exploding gradient problem [16]. It refers to the effect of a given input on the hidden layer, which either decays or blows up exponentially as it passes through the network in cycles. There have been many attempts to solve this problem [17]. One of the most successful architectures targeting the issue is the long short-term memory (LSTM) architecture developed in 1997. In 2014, Cho et al. presented the Gated Recurrent Unit (GRU) architecture, similar to LSTM but with fewer parameters [16].

For tasks where input and output are sequences, Goodfellow et al. (2017) [16] suggest using a gated recurrent net such as LSTM or GRU. Chung et al. (2014) [10] conducted an empirical study of RNNs comparing different recurrent units. The study shows that the LSTM units and GRUs outperformed traditional recurrent units without gates. The study further suggests that there are situations where the LSTM units perform better than GRUs, and situations where GRUs perform better. Chung et al. (2014) therefore emphasizes that the choice of unit highly depends on the data set and task in question. Graves (2012) [17] suggests starting the model development phase by conducting an experiment where different network architectures are tested. Graves states that it is important to have simple models with approximately the same number of parameters for comparability reasons.

2.6.3 Long Short-term Memory

The major difference between an LSTM and a traditional RNN is the concept of memory blocks and memory cells introduced in the LSTM architecture [17]. The memory blocks replace the summation units in the hidden layer of a traditional RNN. Each memory block consists of a cell input activation function, at least one memory cell, three multiplicative units (referred to as the input, output and forget gate) and an output activation function [17]. Figure 2.3 shows the schematic of a LSTM block, or memory block, with one single memory cell. A common choice for the input and output activations is the hyperbolic tangent *tanh* [17].

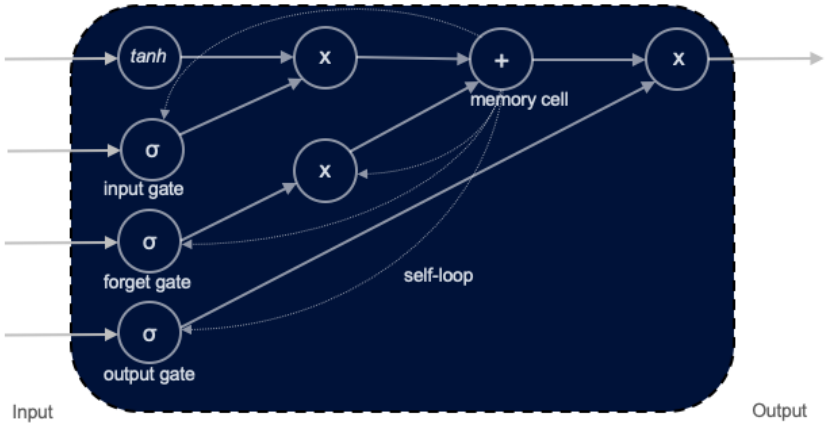


Figure 2.3: Schematic of a LSTM unit with one memory cell.

The gates control the activation of the memory cell whilst no activation function is applied directly to the memory cell itself. Instead, the cell has an internal state with an update rule that incorporates the information from the input gate and forget gate. Each gate has an activation function, here and typically, it is the logistic sigmoid σ. When the activation is close to 0, the gate is closed and when the activation is close to 1, the gate is open. If the input gate is open, the activation of the memory cell is overwritten by new inputs. If the output gate is open, previous information is used by the neural network. Closing the forget gate allows the memory cells to reset itself and forget previous input. The memory cell is connected to the gates through weights, referred to as *peephole*

weights [17].

The forward pass in an LSTM is similar to the one in an RNN. The difference, in simple terms, is that the hidden units are replaced by memory blocks. The information from the gates thus need to be added in the forward and backward pass of the neural net [17]. Let the weight from i to j be w_{ij} and the input i at time t be $x_i^{(t)}$, consistent with the previous RNN section. Let the gates have indexes ι for input, ϕ for forget and ω for output. Let the peephole weights for the corresponding gates be $w_{c\iota}$, $w_{c\phi}$ and $w_{c\omega}$. The index c refers to the memory cell in question. The state of memory cell c at time t is further denoted s_c^t. Moreover, let the network input and activation of j at time t once again be $a_j^{(t)}$ and $b_j^{(t)}$. The number of inputs, outputs and cells in the hidden layer and the number of memory cells are denoted I, K, H and C respectively.

In a LSTM forward pass, all activations are computed for a sequence $\mathbf{x}$ of length T. At $t = 0$, all activations and states are set to zero. Thereafter, the sequence is passed, one t at a time, starting at $t = 1$. The forward pass of a hidden layer in a LSTM network, for a single memory block, is obtained by first computing the network input and activation of the input gate [17],

$$a_\iota^{(t)} = \sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} b_h^{(t-1)} + \sum_{c=1}^{C} w_{c\iota} s_c^{(t-1)},$$

$$b_\iota^t = \sigma(a_\iota^{(t)}).$$

The next step is to calculate the network input and activation to the forget gate [17],

$$a_\phi^{(t)} = \sum_{i=1}^{I} w_{i\phi} x_i^{(t)} + \sum_{h=1}^{H} w_{h\phi} b_h^{(t-1)} + \sum_{c=1}^{C} w_{c\phi} s_c^{(t-1)},$$

$$b_\phi^t = \sigma(a_\phi^{(t)}).$$

The network input to the memory cell is the same as the network input to a traditional RNN. Instead of calculating the activation of the network input $a_c^{(t)}$ as in the simple RNN case, the LSTM updates the state using the activations from the input gate and forget gate. The network input to the memory cell and the update of the state is given by [17],

$$a_c^{(t)} = \sum_{i=1}^{I} w_{ic} x_i^{(t)} + \sum_{h=1}^{H} w_{hc} b_h^{(t-1)},$$

$$s_c^{(t)} = b_\phi^{(t)} s_c^{(t-1)} + b_\iota^{(t)} g(a_c^{(t)}),$$

where g is either the logistic sigmoid σ or the hyperbolic tangent $tanh$. The network input and activation of the output gate can then be calculated [17],

$$a_\omega^{(t)} = \sum_{i=1}^{I} w_{i\omega} x_i^{(t)} + \sum_{h=1}^{H} w_{h\omega} b_h^{(t-1)} + \sum_{c=1}^{C} w_{c\omega} s_c^{(t)},$$

$$b_\omega^{(t)} = \sigma(a_\omega^{(t)}).$$

The cell output is then obtained [17],

$$b_c^{(t)} = b_\omega^{(t)} h(s_c^t),$$

where h is either the logistic sigmoid σ or the hyperbolic tangent $tanh$. The cell output is connected to the other blocks in the network layer and the rest of the information is kept hidden from the other blocks [17].

The BPTT backward pass in LSTM is similar to the BPTT backward pass in a regular RNN. The expressions for differentiating $\mathcal{L}$ with respect to the network input, activation, and state can be defined as

$$\delta_j^{(t)} := \frac{\partial \mathcal{L}}{\partial a_j^{(t)}}, \qquad \epsilon_c^{(t)} := \frac{\partial \mathcal{L}}{\partial b_c^{(t)}}, \qquad \epsilon_s^{(t)} := \frac{\partial \mathcal{L}}{\partial s_c^{(t)}}.$$

By repeatedly applying the chain rule, the complete BPTT backward pass of the network can be obtained. Gradients are equal to zero at the first step at $t = T + 1$ and t is then decremented. The number of inputs to the hidden layer, regardless of type (both cell inputs and gate inputs), is denoted G. The expression for the cell output is obtained from

$$\epsilon_c^{(t)} = \sum_{k=1}^{K} w_{ck} \delta_k^{(t)} + \sum_{g=1}^{G} w_{cg} \delta_g^{(t+1)}.$$

Let h be either the hyperbolic tangent $tanh$ or the logistic sigmoid σ. The backward pass equation for the output gate is then

$$\delta_\omega^{(t)} = \sigma'(a_\omega^{(t)}) \sum_{c=1}^{C} h(s_c^{(t)}) \epsilon_c^{(t)}.$$

Let g also be either the hyperbolic tangent $tanh$ or the logistic sigmoid σ. The states and cells backward pass equations are

$$\epsilon_s^{(t)} = b_\omega^{(t)} h'(s_c^{(t)}) \epsilon_c^{(t)} + b_\phi^{(t+1)} \epsilon_s^{(t+1)} + \omega_{cl} \delta_l^{(t+1)} + \omega_{c\phi} \delta_\phi^{(t+1)} + \omega_{c\omega} \delta_\omega^{(t)},$$

$$\delta_c^{(t)} = b_l^{(t)} g'(a_c^{(t)}) \epsilon_s^{(t)},$$

respectively. The forget and input gate backward pass equations are thereafter given by

$$\delta_\phi^{(t)} = f'(a_\phi^{(t)}) \sum_{c=1}^{C} s_c^{(t-1)} \epsilon_s^{(t)},$$

$$\delta_l^{(t)} = f'(a_l^{(t)}) \sum_{c=1}^{C} g(a_c^{(t)}) \epsilon_s^{(t)}.$$

2.6.4 Gated Recurrent Unit

The GRU is similar to the LSTM unit in the sense that it has gating units controlling the information flow inside the units, or blocks. The major difference between GRU and LSTM is that GRU networks lack memory cells, or other internal states [9]. GRU further has two gates, an update gate and a reset gate, instead of three. As a result, GRU networks have less parameters than LSTM networks. The GRU and LSTM unit have in common that they keep the information from the previous timestep at $t-1$ and add the new information at t on top of it. This is not the case in traditional RNNs, which completely replace the information at each timestep. The addition makes the LSTM and GRU networks remember important features for long periods and reduces the vanishing gradient problem that traditional RNNs are commonly facing [9]. The schematic of a GRU is illustrated in figure 2.4.

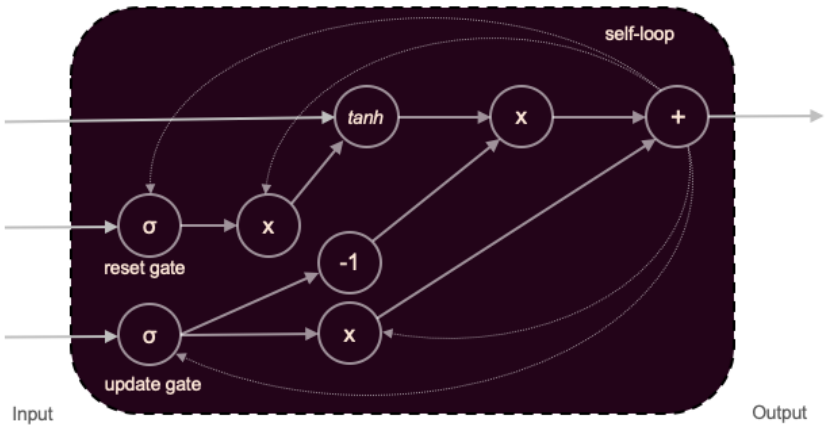


Figure 2.4: Schematic of a GRU.

The forward pass in a GRU network is given by calculating the output $h^{(t)}$ for each cell [16],

$$h^{(t)} = b_u^{(t-1)} h^{(t-1)} + (1 - b_u^{(t-1)}) tanh\left(\sum_{i=1}^{I} w_{ih} x_i^{(t-1)} + \sum_{h'=1}^{H} w_{h'h} b_r^{(t)} h^{(t-1)} \right),$$

where b_u and b_r is the activation from the update and reset gate respectively. The connection from i to j is again denoted w_{ij} and the input i at time t is denoted $x_i^{(t)}$. The network input and activation of the update gate, indexed u, is given by

$$a_u^{(t)} = \sum_{i=1}^{I} w_{iu} x_i^{(t)} + \sum_{h=1}^{H} w_{hu} h_h^{(t)},$$

$$b_u^{(t)} = \sigma(a_u^{(t)}).$$

The network input and activation of the reset gate, indexed r, is similarly given by [16]

$$a_r^{(t)} = \sum_{i=1}^{I} w_{ir} x_i^{(t)} + \sum_{h=1}^{H} w_{hr} h_h^{(t)},$$

$$b_r^{(t)} = \sigma(a_u^{(t)}).$$

The BPTT backward pass in GRU is obtained by applying the same algorithm as for the BPTT backward pass in LSTM derived in the previous section.

2.6.5 Robustness

Robustness of a model refers to the model's ability to consistently generate accurate predictions under various conditions. Adversarial robustness is a similar concept which refers to the models ability to not be fooled when input is manipulated in order to disturb the model or network [27]. Sengputa and Friston (2018) [35] show that highly accurate RNNs, including LSTMs and GRUs, can be unstable and lack robustness. Goodfellow et al. [16] suggest using *data augmentation*, which means creating fake data and adding it to the training set, in order for the model to generalize better and increase robustness.

Machine learning and deep learning models have generated great results in various fields and applications. There is still however a reliability issue when it comes to machine learning due to the lack of proven robustness of models [26]. Robustness and adversarial robustness is clearly a major issue and focus area especially in security-critical applications [8]. It has gained more attention recently as models have improved and a lot of research has been published on the area [26], [27], [41], [31], [40], [39].

Most research on robustness and neural networks concerns feed-forward neural networks and convolutional neural networks. Some work has been published on robustness in RNNs including LSTMs and GRUs. The Propagated-output Quantified Robustness Algorithm for RNNs (POPQORN) [29] (2019) has been considered state-of-the-art on the topic. In November 2021 CERT-RNN, another framework for certifying robustness of RNNs, was published [11]. The authors state that CERT-RNN outperforms POPQORN in terms of effectiveness and efficiency. Amini et al. proposed yet another algorithm to improve robustness of RNNs in 2021 [1].

Chapter 3

Method

In this section, we describe our method for data augmentation, sample generation, model development and performance evaluation. A machine with Intel Core i5-4690K CPU and 8.0 GB DDR3 RAM was used to generate data, train and evaluate the model. The data generation methods and models were implemented using QuantLib-Python [12] and the neural network was implemented by using the machine learning platform TensorFlow [37]. The software environment was Python 3.8. The code is available on GitHub[1].

3.1 Contract Specification

The considered contract was specified as a ten-year *pay fixed receive floating* plain vanilla IRS where the interest rate is set in advance and pays in arrears. The floating-rate payments are expected every 3 months and fixed-rate payments every 6 months. The notional principal amount was set to 100M USD. All payments were assumed to be made in USD and the floating rate to be based on the LIBOR spot rate. The contract was defined as initialized at $t = 0$, 3 months before the first payment.

[1] https://github.com/frodiie/Credit-Exposure-Prediction-GRU

3.2 Data

The objective of the model was to predict the Expected Positive Exposure of the contract specified in section 3.1. The input to the model was market data, Hull-White parameters and deviation from ATM strike rate as described in table 3.1. The Hull-White parameters capture the market dynamics and the deviation from the ATM strike rate expressed in BPS relates to the contact specification.

Table 3.1: Features description.

yield	10 year yield curve on USD bonds.
σ	Hull-White volatility parameter.
a	Hull-White mean rate reversion parameter.
strike	Positive or negative deviation from the ATM strike rate.

A data set consisting of 1,540 yield curves of observed USD bonds between dates 04-01-2016 and 26-01-2022 was gathered and used. All yields, at maturities less or equal to 10 years, were included in the data set. The data set was properly explored to detect any abnormalities. No samples were excluded.

3.2.1 Augmentation

To increase the amount and diversity of the data and thus improve the robustness of the model, an additional set of augmented yield curves were synthesized. The full data set was expanded to consist of 3,080 different yield curves, 50% observed and 50% augmented. The augmented curves were created through manipulation of the observed data. The manipulation techniques included parallel shifting, tilting, merging and multiplication of different yield curves. Each augmented sample was created by performing 5-10 manipulation techniques in sequence on one observed yield curve. Examples of augmented yield curves are presented in figure 3.1.

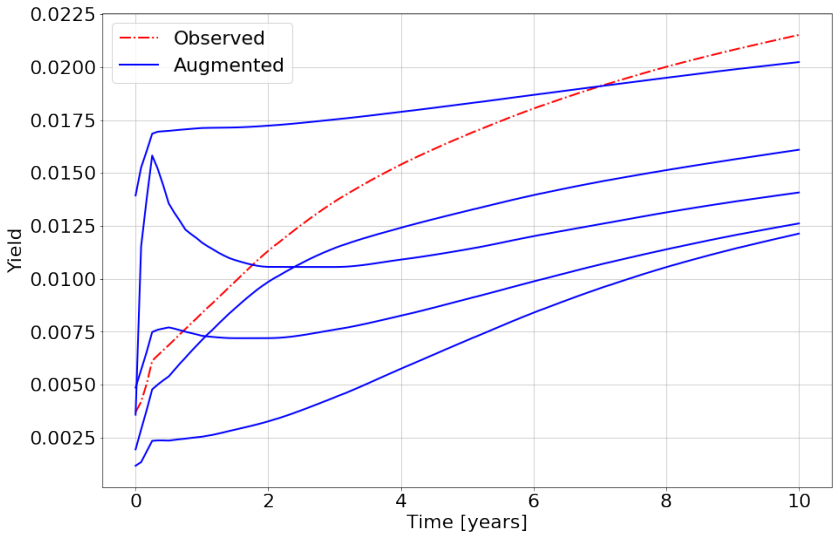
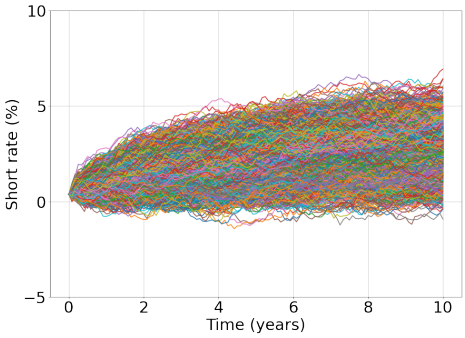


Figure 3.1: Five augmented yield curves. All curves are based on the same observed curve marked in the figure.

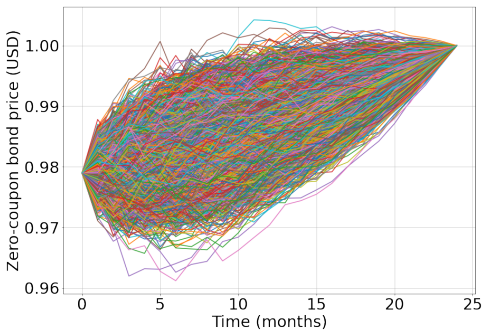
3.2.2 Sample Generation

The next step in generating data was to calculate the exposure profile for each yield curve using different a-, σ- and *strike*-values. The yield curves were interpolated to a monthly time grid using cubic spline interpolation.

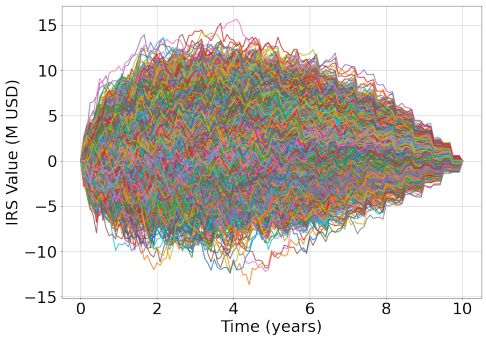
The exposure calculation for each yield curve and parameter combination consisted of four steps shown in figure 3.2. The first step was to simulate the *short rate* paths using the Hull-White One-Factor model in accordance with the theory in section 2.4. The simulated paths are obtained from equation 2.16 and 2.17 by using a step-wise stochastic simulation method from a defined starting point. An example of a short rate simulation is shown in figure 3.2(a). The *zero-coupon bond price* paths could then be obtained from equation 2.10, 2.12 and 2.13. Figure 3.2(b) show simulated bond price paths. The risk-neutral *IRS value* paths in figure 3.2(c) were calculated from the bond prices using equation 2.8. Once the value paths were obtained, the Expected Positive Exposure was computed in accordance with equation 2.2 as explained in section 2.1.1. All obtained exposures follow a pattern of jumps which correspond to the specified transaction frequency of the contract. An example of an expected exposure over the lifetime of the trade is shown in figure 3.2(d).



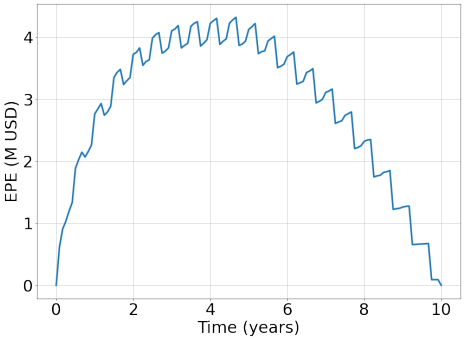
(a) Simulated paths of the short rate r_t.



(b) Simulated zero-coupon bond price paths $p(t, T)$. All price paths reach one at maturity.



(c) Interest rate swap values $V(t)$. The values are both positive and negative over the lifetime of the contract.



(d) Expected Positive Exposure $EPE(t)$. The obtained exposures follow a pattern of jumps that correspond to the specified transaction frequency of the contract.

Figure 3.2: Steps in the exposure calculation.

Two data sets were generated in order to develop a first model and then expanding it with an additional feature. Intervals for the parameters a and σ in the Hull-White One-Factor model and the number of paths for each Monte Carlo simulation were defined to mimic realistic bank settings for the first data set. Five a-values and six σ-values, evenly spaced in the intervals, were used in simulation. The exposures for each yield curve were thus computed using 30 different parameter combinations. The parameters settings, number of Monte Carlo paths and number of samples are shown in table 3.2. The deviation from the ATM strike rate was added in the second data set. The interval was defined to mimic realistic bank settings. In order to keep the amount of data within reasonable limits, the intervals for a and σ were made smaller. Three a-, three σ-, and seven

strike-values were evenly spaced in the intervals. The exposure profiles for each yield curve were thus computed using 63 different parameter combinations.

Table 3.2: Data set specification.

Parameter	**Set/Value**	
	Data set 1	*Data set 2*
a	$\in [0.001, 0.23]$	$\in [0.05825, 0.17275]$
σ	$\in [0.001, 0.01]$	$\in [0.00280, 0.00640]$
strike	0	$\in [-3, 3]$ BPS
#observed	1,540	1,540
#augmented	1,540	1,540
#samples	92,400	194,040
#MC paths	5,000	5,000

Both data sets were separately divided into a training and a test set, where 90% of the data was used to train the model and the last 10% to test the model. The test set consisted of augmented curves. During the training phase, 20% of the training data was used for validation. The training data was used for testing network architectures and tuning hyperparameters, and the rest of the data was used for testing the final model. An additional test set was generated to accompany each set in table 3.2. The purpose was to test the generalization performance of the network by introducing exposure profiles generated with new Hull-White parameters. The additional test set's parameters are specified in table 3.3. The parameters a and σ were chosen in the same interval as in *Data set 1* and *Data set 2* respectively, but with an as large as possible deviation from the parameter values in those data sets.

Table 3.3: Extra test set specification.

Parameter	**Set/Value**	
	Test set 1	*Test set 2*
a	$\in [0.0296, 0.201]$	$\{0.0087, 0.144\}$
σ	$\in [0.0019, 0.0091]$	$\{0.0037, 0.0055\}$
strike	0	$\{\pm 0.5, \pm 1.5, \pm 2.5\}$ BPS
#observed	0	0
#augmented	462	809
#samples	9,240	19,400
#MC paths	5,000	5,000

To get a better understanding of how much data that could be useful and how well the data augmentation works, an additional data set was created. This data set consisted of 64K augmented yield curves and the 1,540 observed real life yield curves. The exposures were generated with one combination of parameters, instead of 30 or 63 as in the other sets. The specification of parameter values and number of samples are specified in table 3.4.

Table 3.4: Data set for evaluating data, specification.

Parameter	**Value**
a	0.1155
σ	0.0064
strike	0
#observed	1,540
#augmented	64,000
#samples	65,540
#MC paths	5,000

3.3 Model Development

The neural network was developed in two steps where the initial attempt was based on a contract specified with a fair strike rate at inception. This baseline network was trained and validated on data in *Data set 1* with settings defined in table 3.2. Experiments described below were carried out to find the most suitable network architecture. An additional feature

specified as deviation from at the money (ATM) strike was added in the next stage. The same network architecture was used in an expanded version of the model using data in *Data set 2*.

Each input to the model was a time series consisting of a yield curve sampled with a resolution of one point per month, and the Hull-White parameters a and σ. In the expanded model, the deviation from ATM strike rate was added. The Hull-White parameters a and σ, as well as the *strike* were regarded as constant time series. The output of the model was a time series consisting of the EPE sampled on the same dates as the input yield curve. Adam was used as the optimization algorithm. The hyperbolic tangent *tanh* was used in all architectures, except for the gates which always use the logistic sigmoid as the activation function. A dense layer was added as a final layer to all architectures and all units were biased.

3.3.1 Network Architecture

In order to find a suitable network architecture, a few different reasonable architectures were tested. The architectures had roughly the same number of parameters for the performance to be comparable and the considered network architectures were simple to avoid overfitting.

The considered nets were trained for 30 epochs and with a batch size of 32. The experiment included four RNN architectures with either LSTM units or GRUs. The architectures are presented in table 3.5. All recurrent layers returned the full sequence of hidden states except for the last recurrent layer which only returned the last value. Bidirectional units are denoted BiLSTM and BiGRU. The performance of each model was assessed by evaluating the mean squared error (MSE) between observed and predicted values in the validation set.

Table 3.5: Network architectures to compare LSTM units to a GRUs.

Architecture	**Number of units**		**Parameters**
	Layer 1	*Layer 2*	
LSTM	18	18	6,547
BiLSTM	11	10	6,501
GRU	20	20	6,561
BiGRU	12	11	6,449

Another experiment was carried out to compare the bidirectional unit to the unidirectional unit. The networks were trained for 50 epochs and with a batch size of 32. Each hidden layer had approximately the same number of parameters. The networks are presented in table 3.6.

Table 3.6: Network architectures to compare a unidirectional unit to a bidirectional.

Architecture	**Number of units**		**Parameters**
	Layer 1	*Layer 2*	
GRU	46	32	18,711
BiGRU	32	16	18,969

3.3.2 Layers and Units

A GRU network was expanded by adding recurrent layers and output units. The number of hidden layers was fixed to three. The networks had two recurrent layers with the same number of units and one dense layer. The full sequence of hidden states was returned in the first recurrent layer and only the last value in the sequence was returned in the second recurrent layer. The number of output units per recurrent layer was varied between 8-128 units and the validation losses compared. The networks were trained for 30 epochs.

The number of output units, or nodes, per recurrent layer was then fixed to 16. The number of recurrent layers was varied between 2-10 layers. The validation losses were compared. All recurrent layers except the last returned the full sequence of hidden states. The last recurrent layer only

returned the last value in the sequence. The networks were trained for 30 epochs.

3.3.3 Regularization

The learning curve was inspected and considered throughout all experiments to get an indication of whether the net was suffering from overfitting at any point. The learning curve for the GRU network with two recurrent layers, 32 and 64 number of units per recurrent layer was carefully examined.

3.3.4 Number of Samples

To gain insight into how well the data augmentation worked and how many augmented samples that would be required for the model to learn the mapping, another experiment was carried out. The data set for evaluating data, specified in table 3.4 was used. One combination of parameters was considered instead of 30 or 63. The input to the neural network was solely the yield curve. The model was trained for 50 epochs on 1-64K number of augmented samples which is assumed to roughly represent 30K-1.92M samples in a data set with 30 parameter combinations. The model was tested on the observed data consisting of 1,540 samples and an additional augmented data set consisting of 1,540 randomly selected augmented samples.

3.4 Performance

The proposed network was evaluated on data in the original test set. The yield curves and Expected Positive Exposures were thus unseen by the model but generated with parameter combinations that had been part of the training set. Further generalization performance was then evaluated by testing the performance on data generated with previously unseen parameters specified in table 3.3. The new parameter values were the midpoints of the intervals between any two seen subsequent parameter values

The main metric for evaluating the network was the percentage of predicted points with *absolute errors less than 10 BPS of the notional amount*. For liquid currency and normal market conditions the bid-ask spread for the used interest rate swap is usually in the range between 0.1

and 0.2 BPS of the notional amount. We regard CVA errors as acceptable if they are within the bid-ask spread of the IRS. We assume that our model is used for contracts with high rated counterparties with default probabilities less than 2%. Any absolute exposure error less than 10 BPS of the notional amount will thus not lead to CVA errors that are above the threshold for acceptable errors.

The model was further evaluated by analyzing the *error distributions* and *worst predictions* with regards to i) largest absolute error and ii) largest mean error over the whole curve.

The proposed machine learning model was benchmarked against a traditional Monte Carlo approach by comparing the error distributions with maximum deviations within a 95% confidence interval for the simulated exposures. The confidence interval for the simulated Expected Positive Exposure is given by equation 2.5. The errors as well as the maximum deviations implied by the confidence interval were visualized in histograms in order to compare the distributions. The percentage of samples predicted within a confidence interval for the Monte Carlo simulation could be used as an alternative benchmark. This could however be considered as a rather unforgiving metric since the standard deviation is small or even zero in some data points, often in the beginning and end of the curve.

The efficiency of the method was evaluated by considering the *offline training time* and the *prediction time* compared to the corresponding Monte Carlo simulation time.

Chapter 4

Results

4.1 Sample Generation

The sample generation computation time were dependent on the number of paths in the Monte Carlo simulation. Computation times for varying numbers of simulation paths are shown in table 4.1.

Table 4.1: Simulation time for varying number of Monte Carlo paths.

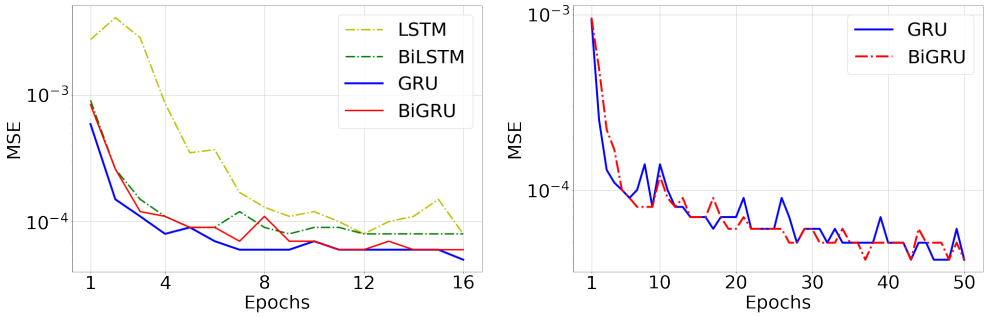
MC paths	Time per sample (s)
5,000	1.75
10,000	2.18
20,000	3.98
50,000	10.45

4.2 Model Development

The validation loss (MSE) for the network architectures specified in table 3.5 comparing an LSTM unit to a GRU are shown in figure 4.1(a) below. The outcome of the experiment indicates that GRU networks are more suitable than LSTM networks for this particular task and data, since the GRU networks give similar or lower validation loss while having a simpler structure.

Figure 4.1(b) shows that GRU and bidirectional GRU networks specified

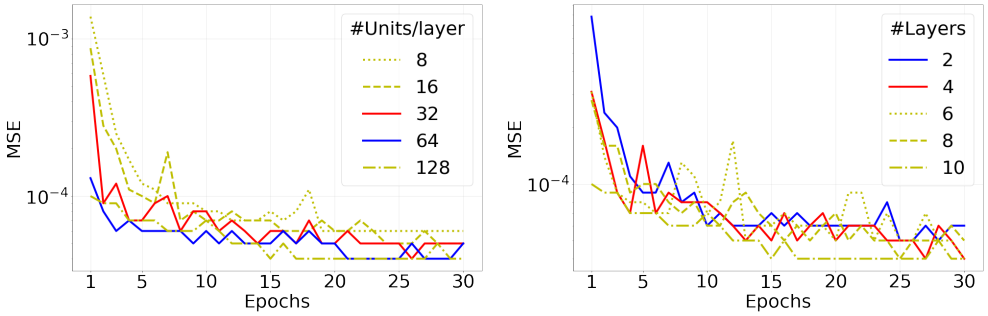
in table 3.6, performed similarly in terms of validation loss. The networks compared in figure 4.1(b) approximately had 19K parameters instead of 6.5K parameters as the networks in figure 3.5. Each training epoch was 1.13 times faster for the GRU network than for the bidirectional GRU network. There were no signs of overfitting. Given that GRU networks have a more simple structure than bidirectional GRU networks, figure 4.2 imply that a GRU network with a normal unidirectional unit is most suitable for the task.



(a) 6.5K parameters. **(b)** 19K parameters.

Figure 4.1: Validation loss for different network architectures in logarithmic scale.

The results obtained from adding more layers and output units are presented in figure 4.2. The figure shows that in this setting, adding more layers may not increase performance. The validation loss of the network with two recurrent layers is somewhat in the middle of the validation losses of networks with more layers. The figure further indicates that 8-16 units per recurrent layer may be too few and that 32-64 is enough.



(a) Varying number of units per recurrent layer. Number of recurrent layers is 2.

(b) Varying number of recurrent layers. Number of nodes per recurrent layer is 16.

Figure 4.2: Validation loss for GRU networks of different sizes in logarithmic scale.

The training times varied as the number of layers and units per recurrent layer varied. The approximate training times per epoch are presented in table 4.2. The training time greatly increased when the number of units increased from 64 to 128, without gaining significant performance. Figure 4.2(a) and table 4.2 thus motivates a choice of 32-64 units per recurrent layer.

Table 4.2: Training time per epoch for varying number of units per recurrent layer.

Units	Training time/epoch (s)
8	130
16	140
32	150
64	170
128	380

The training time increased as the number of layers increased. The approximate training times per epoch is presented in table 4.3. The performance visualized in figure 4.2(b) and the training times in table 4.3 indicates that two recurrent layers is an appropriate choice. The extended training times when adding more layers can not be motivated by an increase in performance.

Table 4.3: Training time per epoch for varying number of recurrent layers.

Layers	Training time/epoch (s)
2	140
4	270
6	410
8	540
10	680

The learning curve for a GRU neural network with 64 units in a first recurrent layer and 32 units in a second recurrent layer is presented in figure 4.3. The figure shows that there are no signs of overfitting. Regularization was therefore not further explored.

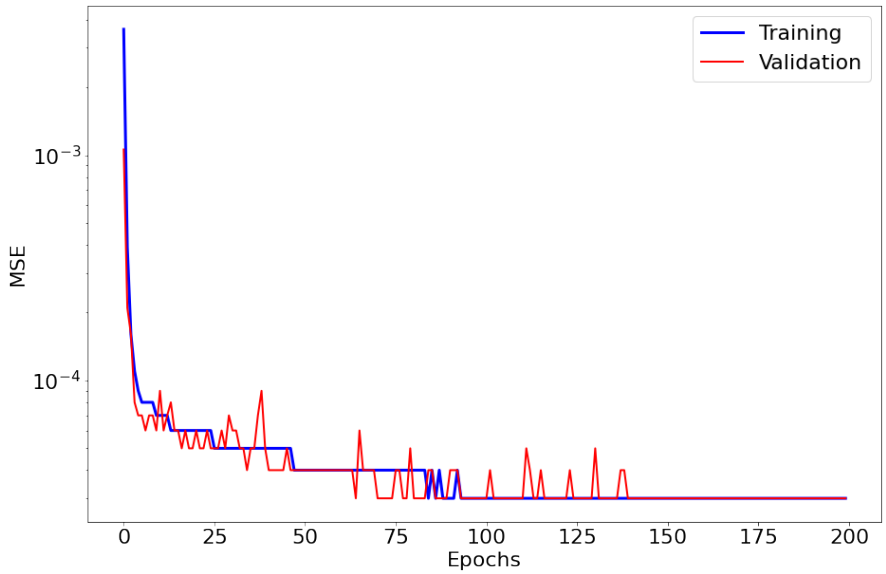


Figure 4.3: Learning curve in logarithmic scale for a GRU network with two recurrent layers of 64 and 32 units.

Figure 4.4 illustrates the outcome of adding more data. The experiment was carried out on a separate data set where the Hull-White parameters and features a and σ were removed. All exposures were instead generated using the same Hull-White parameters. The results show that adding more augmented data with our augmentation technique could increase

performance to a certain degree. The figure shows that it was a significant difference between the test loss when the model was trained on 1K samples compared to when the model was trained on 2K samples. The improvement did however stagnate and there was only a minor difference between the test loss on the model trained on 16K samples and the model trained on 64K samples. The figure further shows similar behavior of the loss on the data generated from the actual observed yield curves and the data generated from augmented yield curves. There were no signs of overfitting.

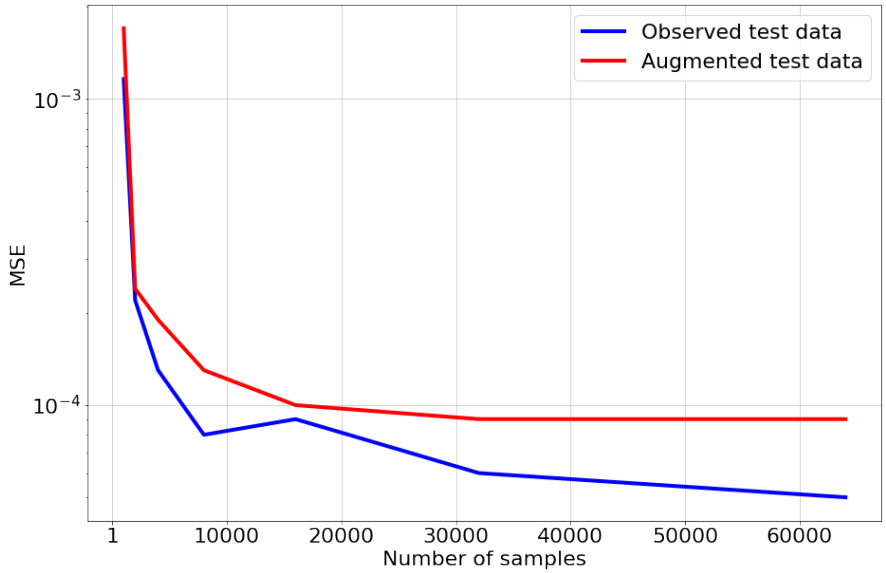


Figure 4.4: Test loss on separate data set in which all training and test data consisted of exposures generated with Hull-White parameters $a = 0.1155$ and $\sigma = 0.0064$. All training data targets (exposures) were generated from augmented yield data. The test data was generated from the actual observed yield curves, marked as observed test data, and from augmented yield curves, marked as augmented test data.

4.3 Model Specification

Our suggested model is a GRU neural network with three hidden layers motivated by the previous section. The first layer is a recurrent GRU layer with 64 units where the full output sequence of hidden states is returned. The second layer is a recurrent GRU layer with 32 units, returning the

last hidden state in the output sequence. The last hidden layer is a dense layer with 121 output units. The optimization algorithm is Adam and the learning rate initially set to 0.0005 and thereafter continuously decreased. A schematic of the network is presented in figure 4.5.

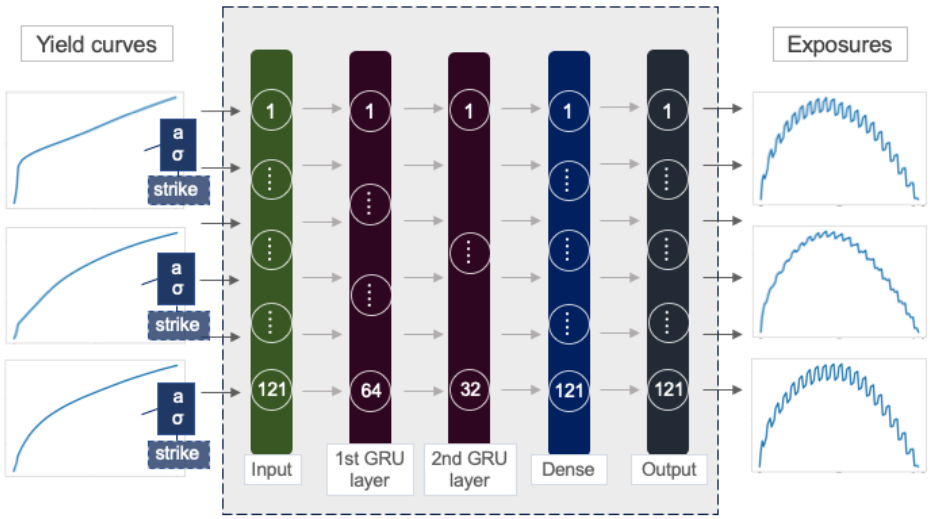


Figure 4.5: Schematic of our proposed neural network.

4.4 Base Model

This section shows the performance of the neural network trained on three features, *yield*, a and σ, with the deviation from the ATM strike rate fixed at 0. *Data set 1* was used to train the model which was then tested on the test data in *Data set 1* and the accompanied extra test set with unseen parameters. The model was trained offline within a workday.

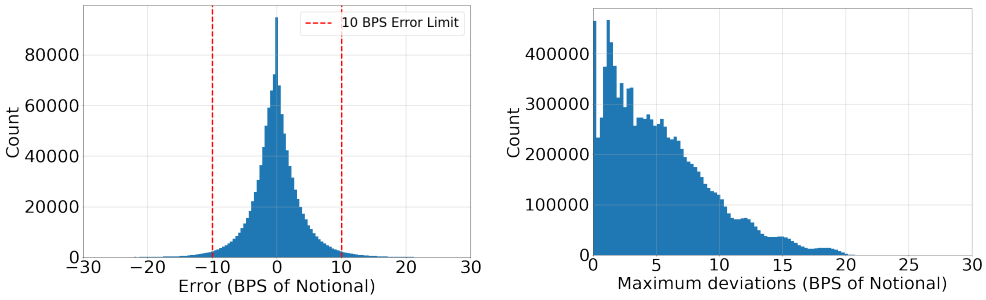
4.4.1 Performance Summary

Table 4.4 summarizes results for the evaluation metrics described and motivated in section 3.5. Only a small proportion of the obtained errors were greater than 10 BPS of the notional. An exposure profile prediction with the model takes 0.4 ms which is approximately 4,000 times faster than the exposure estimation using standard Monte Carlo simulation with 5K paths and 24,000 times faster than Monte Carlo simulation with 50K paths. The distribution of the pointwise errors expressed in BPS of notional is shown in figure 4.6(a). The pointwise errors can be compared

with figure 4.6(b), in which the distribution of the maximum deviations within a 95% confidence interval of the Monte Carlo simulation using 5K paths is expressed in BPS of notional.

Table 4.4: Results statistics on test data with seen Hull-White parameters.

Metric	Value
Absolute errors within 10 BPS of notional (%)	96.90
Prediction time per curve (ms)	0.44
MSE	0.0016
Max. observed error (M USD)	0.449
Max. mean error (M USD)	0.185

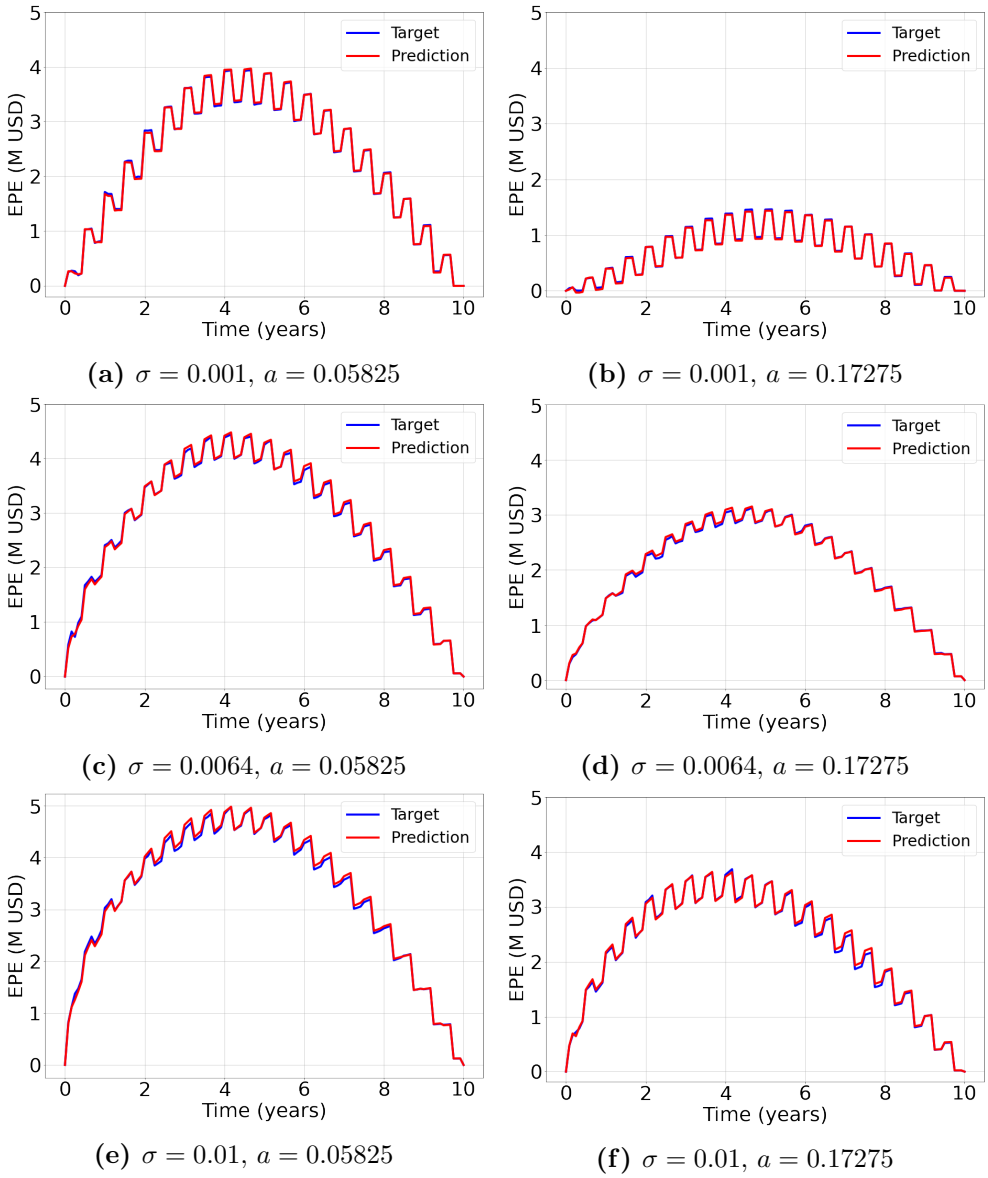


(a) **(b)**

Figure 4.6: Error distribution (a) and distribution of the maximum deviations within a 95% confidence interval for the MC simulation (b).

4.4.2 Representative Predictions

Figure 4.7 shows the network's predictions for eight different parameter combinations and corresponding targets. The predictions are chosen to be representative of the network's performance for the chosen parameter combinations. The selected predictions have a loss equal to, or approximately equal to the median loss for each parameter combination. Note that the exposures originate from different yield curves.



(a) $\sigma = 0.001$, $a = 0.05825$ **(b)** $\sigma = 0.001$, $a = 0.17275$

(c) $\sigma = 0.0064$, $a = 0.05825$ **(d)** $\sigma = 0.0064$, $a = 0.17275$

(e) $\sigma = 0.01$, $a = 0.05825$ **(f)** $\sigma = 0.01$, $a = 0.17275$

Figure 4.7: Median loss predictions for different Hull-White parameter combinations.

Figure 4.8 shows how the median loss depends on the parameters a and σ. The loss is the largest for large σ and small a and the smallest for small σ and large a.

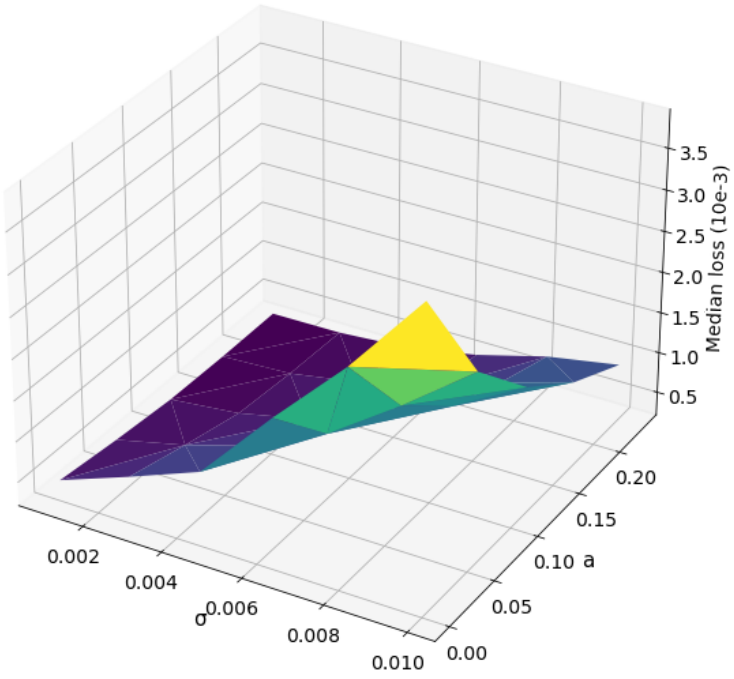
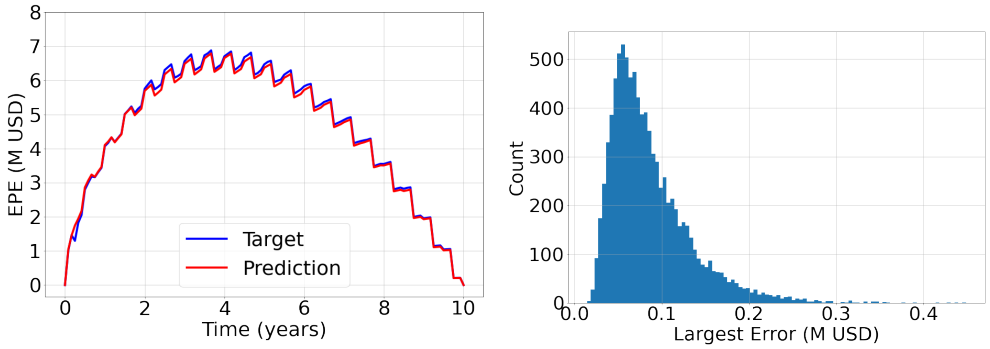


Figure 4.8: Median loss against Hull-White parameters

4.4.3 Error Analysis

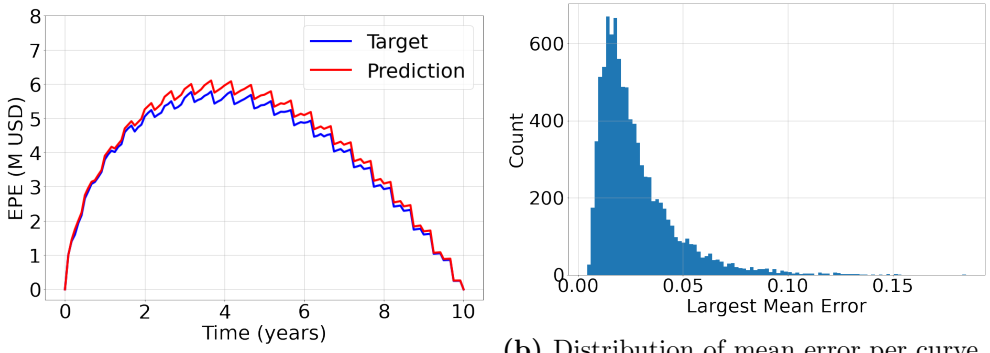
The largest observed test error, presented in table 4.4, was found in the prediction of the expected exposure profile presented in figure 4.9(a) below. The largest absolute errors for each predicted exposure profile are visualized in a histogram in figure 4.9(b). The distribution is right-skewed and centered around 0.06 M USD. The maximum error, 0.449 M USD, in figure 4.9(a) could presumably be considered as an outlier. The largest mean error in an exposure profile was observed for the exposure profile in figure 4.10(a). Figure 4.10(b) displays a histogram of absolute mean errors for each predicted exposure profile. The distribution is right-skewed and centered around 0.02 M USD. The maximum mean error 0.1848 M USD in figure 4.10(a) could reasonably also be considered as an outlier.



(a) Max. error 0.449 M USD at Time=0.2 years.

(b) Distribution of max. error per curve.

Figure 4.9: Worst prediction in terms of largest error and corresponding distribution.



(a) Max. mean error 0.185 M USD.

(b) Distribution of mean error per curve.

Figure 4.10: Worst prediction in terms of mean error and corresponding distribution.

4.4.4 Generalization Performance

The model was tested on the additional test data with unseen Hull-White parameters. The parameters a and σ of the test data were chosen in the range of the parameters of the training data with an as large as possible deviation from the seen values. The result statistics are presented in table 4.5. All statistics are comparable to those presented for the seen parameters. The results indicate that the network can handle various feature values in the range it was trained on and not only on the previously seen parameters.

Table 4.5: Results statistics on test data with unseen Hull-White parameters.

Metric	Value
Absolute errors within 10 BPS of notional (%)	96.97
Prediction time per curve (ms)	0.45
MSE	0.0016
Max. observed error (M USD)	0.462
Max. mean error (M USD)	0.181

4.5 Extended Model

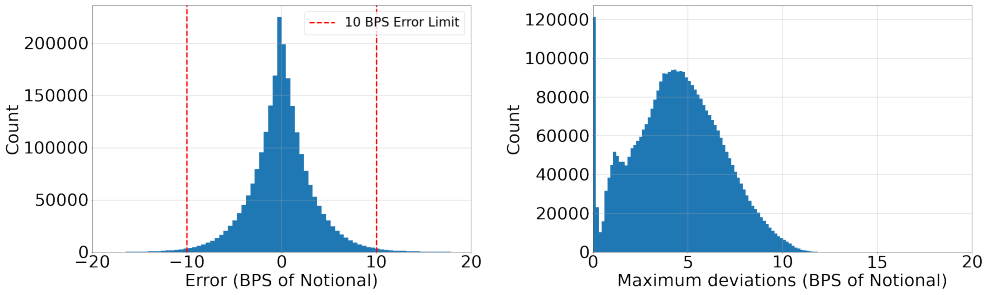
This section shows the performance of the neural network for the model where deviation from the ATM strike rate was included as a feature, allowing contracts with variable strike rates. The model is thus trained on all four features described in table 3.1, *yield*, a, σ, and *strike*. The model was trained offline within a workday.

4.5.1 Performance Summary

Table 4.6 summarizes the results on the evaluation metrics. Only a small proportion of the obtained errors were greater than 10 BPS of the notional. An exposure profile prediction with the model takes 0.6 ms which is approximately 2,900 times faster than the exposure estimation using standard Monte Carlo simulation with 5K paths and 17,400 times faster than Monte Carlo simulation with 50K paths. The distribution of the pointwise errors expressed in BPS of the notional is shown in figure 4.11(a). The pointwise errors of the model can be compared with figure 4.11(b), in which the distribution of the maximum deviations within a 95% confidence interval of the Monte Carlo simulation using 5K paths is expressed in BPS of notional.

Table 4.6: Results statistics on test data with seen Hull-White parameters and deviation from ATM strike.

Metric	Value
Absolute errors within 10 BPS of notional (%)	98.56
Prediction time per curve (ms)	0.59
MSE	0.0011
Max. observed error (M USD)	0.753
Max. mean error (M USD)	0.288

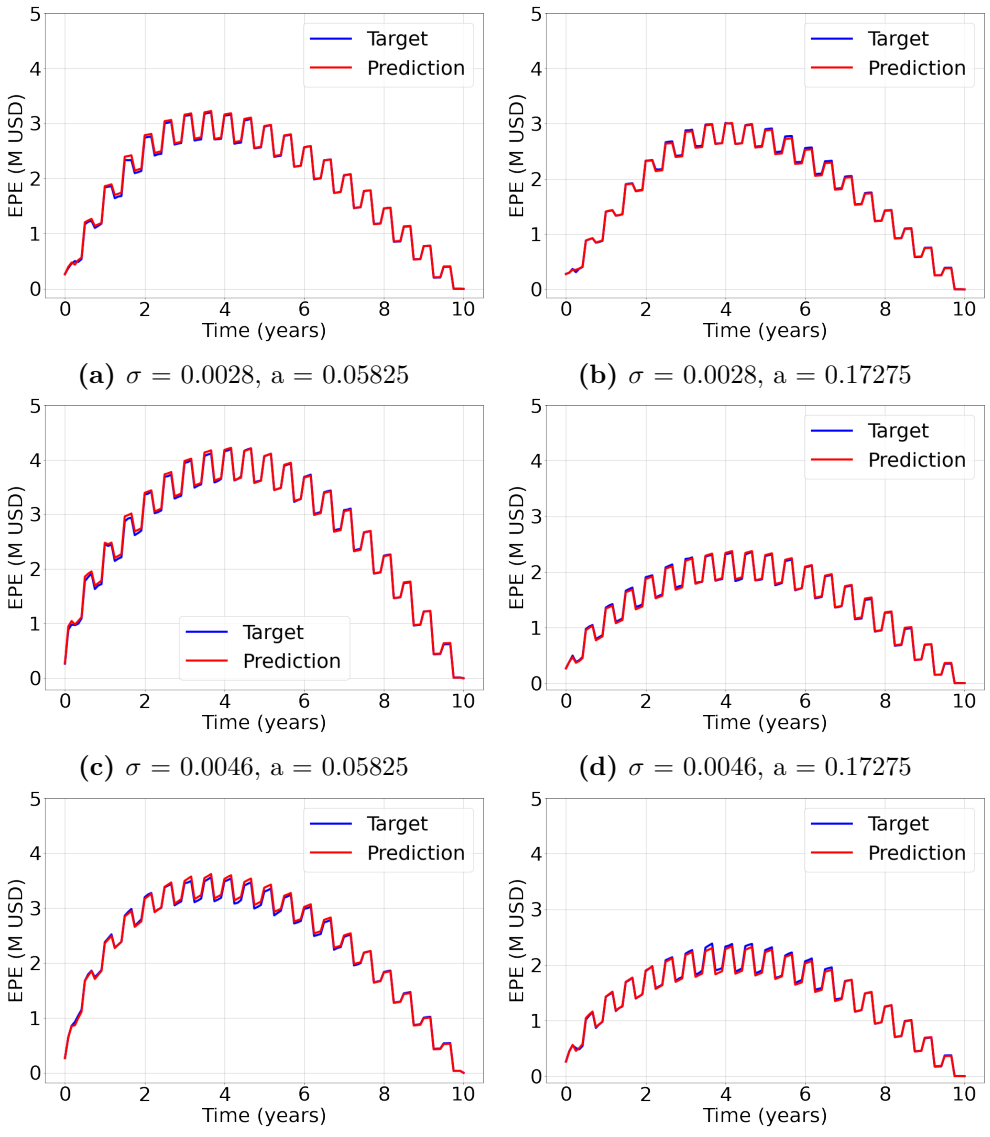


(a) **(b)**

Figure 4.11: Error distribution (a) and distribution of the maximum deviations within a 95% confidence interval for the MC simulation (b).

4.5.2 Representative Predictions

Figures 4.12 and 4.13 show the network's predictions for different parameter combinations and corresponding targets. The strike rates in figures 4.12 and 4.13 are set to 3 BPS below and above the ATM strike rate respectively. The presented predictions are chosen to be representative of the network's performance for the chosen parameter combinations. The mean error of the presented predictions is equal or approximately equal to the median of all mean errors for its parameter combination. Note that the exposures are based on different yield curves.

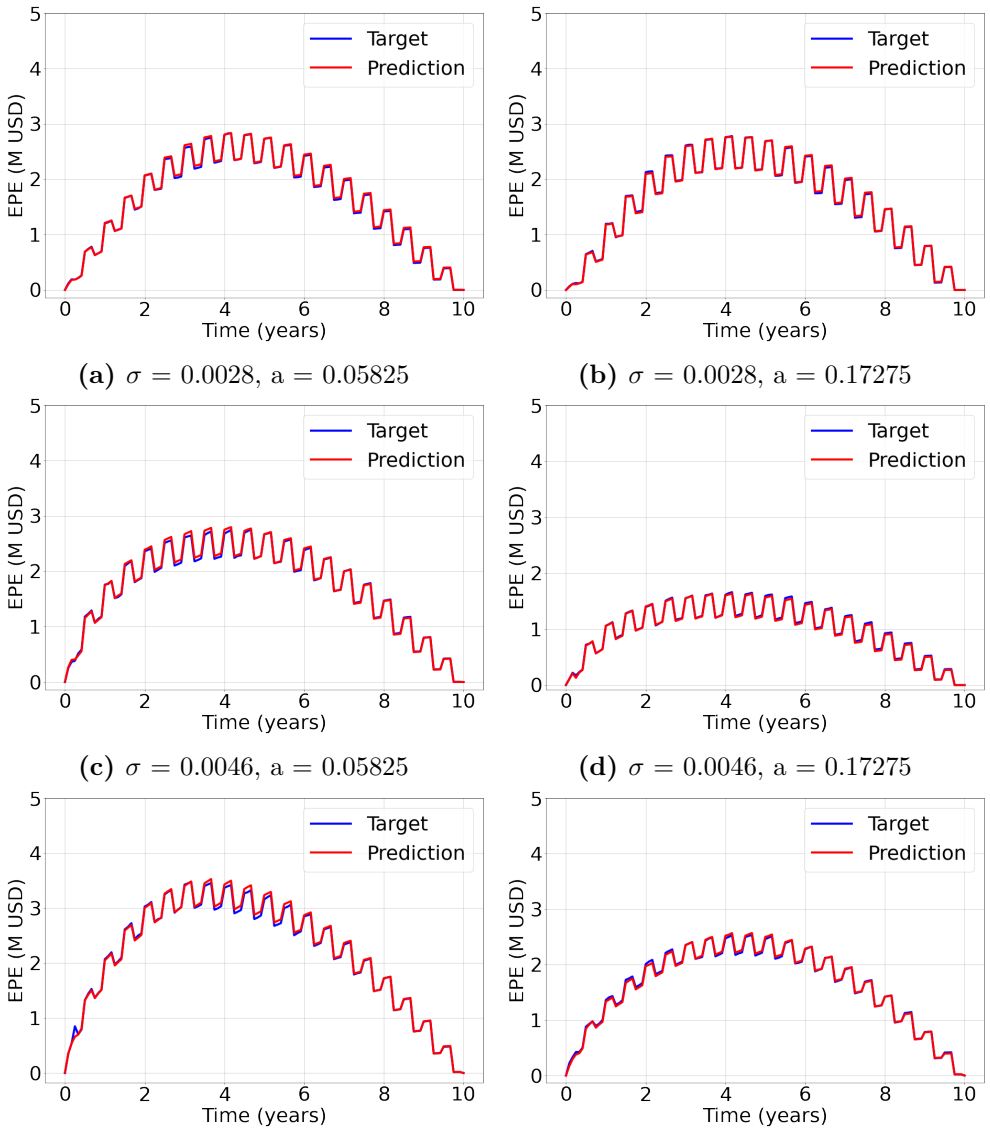


(a) $\sigma = 0.0028$, a $= 0.05825$ **(b)** $\sigma = 0.0028$, a $= 0.17275$

(c) $\sigma = 0.0046$, a $= 0.05825$ **(d)** $\sigma = 0.0046$, a $= 0.17275$

(e) $\sigma = 0.0064$, a $= 0.05825$ **(f)** $\sigma = 0.0064$, a $= 0.17275$

Figure 4.12: Median loss predictions for different Hull-White parameter combinations and strike rate set to ATM $-$ 3 BPS.



(a) $\sigma = 0.0028$, a $= 0.05825$ **(b)** $\sigma = 0.0028$, a $= 0.17275$

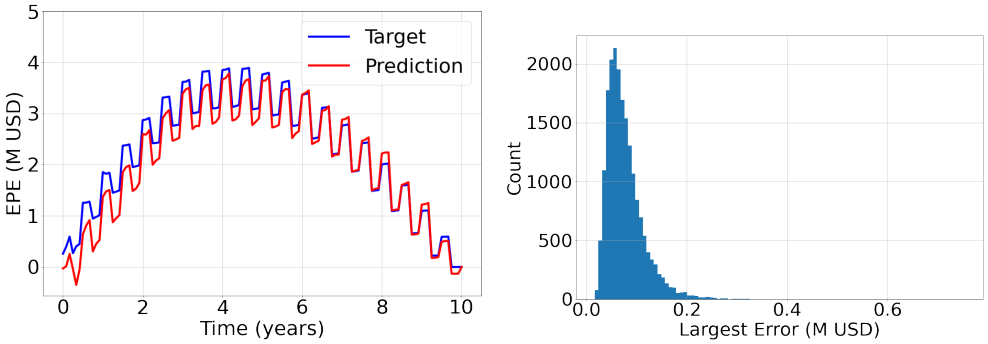
(c) $\sigma = 0.0046$, a $= 0.05825$ **(d)** $\sigma = 0.0046$, a $= 0.17275$

(e) $\sigma = 0.0064$, a $= 0.05825$ **(f)** $\sigma = 0.0064$, a $= 0.17275$

Figure 4.13: Median loss predictions for different Hull-White parameter combinations and strike rate set to ATM + 3 BPS.

4.5.3 Error Analysis

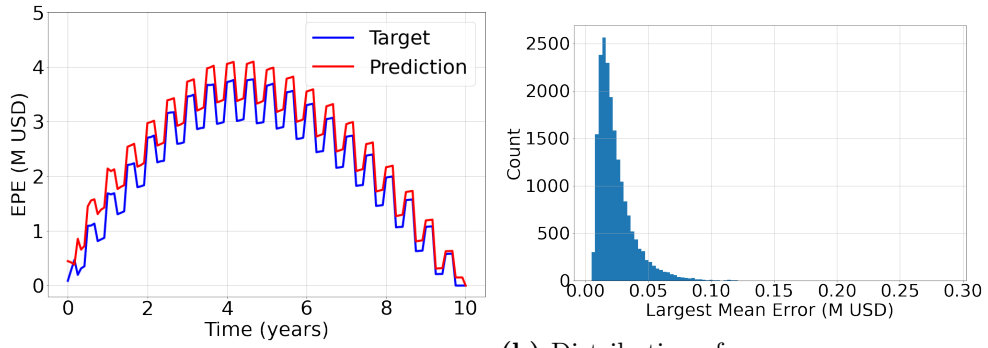
The largest observed test error, which was reported in table 4.6, was found in the prediction of the expected exposure profile shown in figure 4.14(a) below. The largest mean error over the entire curve was observed for the exposure profile in figure 4.15(a). The histograms in figure 4.14(b) and 4.15(b) show the corresponding error distributions. The maximum observed total and mean error are both far separated from the majority of the observations in their corresponding histograms.



(a) Max. error 0.753 M USD at Time=0.4 years.

(b) Distribution of largest error per curve.

Figure 4.14: Worst prediction in terms of largest error and the corresponding error distribution.



(a) Max. mean error 0.288 M USD.

(b) Distribution of mean error per curve.

Figure 4.15: Worst prediction in terms of largest mean error and the corresponding error distribution.

4.5.4 Generalization Performance

The model was tested on the additional test data with unseen Hull-White parameters and deviations from the ATM strike rate. The parameters were chosen in the same interval as the seen parameters but with as large as possible deviations from the seen parameters. The result statistics are presented in table 4.7 below. All results are comparable to those presented for the initial test set with seen parameters which implies a good generalization performance to new parameters within the interval of seen data.

Table 4.7: Results statistics on test data with unseen Hull-White parameters and deviation from ATM strike.

Metric	**Value**
Absolute errors within 10 BPS of notional (%)	98.19
MSE	0.0013
Max. observed error (M USD)	0.561
Max. mean error (M USD)	0.255

Chapter 5

Discussion

5.1 Performance

The proposed models were capable of reproducing expected counterparty credit exposures for a wide variety of market conditions. The extended model introduces additional challenges as the variable strike rate leads to more variation in the exposure profiles and added complexity in the mapping. The errors over the entire test set were generally smaller for the extended model, which indicates that the network responds well to additional features and increased complexity of task.

Our machine learning model was benchmarked against a traditional Monte Carlo approach by comparing the error distributions with maximum deviations within a 95% confidence interval for the simulated exposures. The errors of the proposed model are comparable in magnitude to the accepted deviations implied by the confidence intervals. For this particular task and data, neural networks could thus be regarded to generate equivalent results to those from a Monte Carlo simulation.

Determining the confidence intervals furthermore brought insight into how simulation variance may bring errors. The samples generated with high volatility parameters are associated with a higher uncertainty in the Monte Carlo simulation. The better overall accuracy of the extended model may to some degree be attributed to the chosen parameter intervals, where the smallest and largest volatility parameters were omitted from the data set in order to keep the amount of data within reasonable limits.

It is reasonable to assume that the model accuracy can be increased by reducing the variance, either by variance reduction techniques, or by using a larger number of Monte Carlo simulation paths.

The distributions over the largest errors were similar for both models which we present in the histograms in figure 4.14(b) and 4.15(b). It is however noted that the largest observed absolute error and mean error for a single curve was considerably larger for the extended model. The presented extremes are clearly separated from the majority of observations which implies that the largest observed absolute error and mean error may be regarded as outliers for both models. It should be noted that the largest errors are less pronounced on a longer tenor, especially for the extended model. This property is preferable since longer tenors are associated with more counterparty risk, and thus a larger sensitivity to errors when pricing xVA.

Comparable performance was demonstrated when the model was tested on data with a set of parameters that were different from, but within the same interval as those included in the training set. This result, which was observed both for the base model and for the extended model, suggests that the neural network can interpolate to new data in the specified interval. This could be a useful insight if additional features were to be added in a future extension of the model.

With regards to computational efficiency, neural networks demonstrate significant advantages over Monte Carlo methods. Our model generates predictions in less than 1 ms, which is significantly more efficient than the corresponding Monte Carlo simulation used in this example. Furthermore, the computational efficiency of the Monte Carlo method is greatly dependent on the number of performed simulations. The number of simulations could be tens or hundreds of thousands in an industry context for portfolios with more risk factors, which is significantly larger than used in this example. The possible benefits of a machine learning approach may thus be even greater than demonstrated in this example.

Another advantage of the model was the rather short offline training time. Both models were trained within a workday. It is fair to assume that more data and longer training would further increase the performance of the network.

5.2 Future Development

This report aims to provide insight into a class of methods that have the potential to improve the efficiency of banks' internal processes. In its current state, the model proposed in this project has its greatest advantages in quick single-contract exposure evaluations that could be used in front office xVA solutions. The model could be used as an efficient pricing tool and complement to existing pricing engines. If a client requests xVA pricing for a single trade, the model could give a quick indication of the price level. The area of application could be expanded by extending the model, for example by training also on quantiles of the exposure. With further development, the proposed architectures may prove useful especially for contracts with high-rated counterparties, traded in a normal and liquid market.

To be useful in an industry context, the machine learning model that is developed here may be applied to observed market data and exposure data generated within the bank. We chose to generate all data using the Hull-White One-Factor model. Even though the Hull-White One-Factor model is researched and established in financial theory, there exist more complex models that more accurately represent the dynamics of the short rate. Reliance on a specific short rate model could be omitted in a future attempt.

If deployed in an industrial context, reasonable standards for data gathering and model retraining should be established. The computational costs of offline training increase with the amount of data and retraining frequency. The performance of the model would presumably also increase. The trade-off between performance and costs needs further assessment. As proposed in this project, data augmentation may bring increased robustness and a better generalization to unexpected market conditions. Further evaluation of robustness of the model was intentionally omitted due to the recent rapid development in the field. It would be valuable to investigate robustness of the network if this method were to be adopted in industry. It is left for future work to apply a robustness framework to the proposed model and neural network approach.

The time resolution of the generated predictions could be different in a future model. In the model proposed here, a monthly resolution was used in all steps of the data generation. The choice of data resolution was made with consideration taken to simulation accuracy and computational

efficiency. The differences in the obtained exposure profiles were small when comparing simulations on a monthly and a daily level. Priority could therefore be placed on computational efficiency, for which a more sparse data set showed great advantages.

Regarding the network architecture, we suggest a GRU neural network with two recurrent hidden layers for future attempts with added features. In our proof of concept model, the simple structure is enabled by the complexity of the GRU. The results show that a GRU is to prefer over an LSTM unit or bidirectional GRU. In more complex applications, adding units and layers may prove beneficial.

Models developed for the financial industry must often comply with current standards and regulations. The usage of machine learning is particularly challenged by requirements of transparency, reliability and accountability. Many banks have nevertheless started to utilize machine learning to obtain data-driven insights in some parts of the organization. The proposed area of use is an example of a setting where machine learning methods may prove valuable as the models are currently not subject to extensive regulation.

It is unlikely that machine learning will provide an alternative to traditional MC pricing engines used for end of day pricing and P&L any time soon. The daily work of an xVA desk includes several xVA evaluations on a portfolio level. The xVA desks in financial institutions therefore require flexible models to handle large and frequent variations in their portfolios. This is an immense challenge for machine learning methods, which are generally trained for a static context. It is however possible to continuously update a neural network by feeding it with new data. The model proposed in this project is incapable of handling diversification effects. We do however acknowledge the possibility of extending a machine learning model to portfolio-level calculations. A model trained on standardized cash flows that represent the portfolio transactions could possibly achieve feasible results. Such a model would however be immensely more complex than the single-contract models proposed here, and would thus be associated with greater challenges in accurate tuning and calibration.

Bibliography

[1] Amini, A., Liu, G. & Motee, N. Robust Learning of Recurrent Neural Networks in Presence of Exogenous Noise. *2021 60th IEEE Conference on Decision and Control (CDC)*, (3 May, 2021): 783-788. Available at: https://doi.org/10.48550/arXiv.2105.00996.

[2] Albanese, C., Crépey, S., Hoskinson, R. & Saadeddine, B. XVA Analysis from the Balance Sheet. *Quantitative Finance*, (2021): 21:1, 99-123. Available at: https://doi.org/10.1080/14697688.2020.1817533.

[3] The Basel Committee on Banking Supervision. *Basel Committee Charter*. Bank for International Settlements, 5 Jun, 2019. Available at: https://www.bis.org/bcbs/charter.htm.

[4] BIS Statistics Explorer. *Global OTC Derivatives Market*. Bank of International Settlements, 21 Apr, 2022. Available at: https://stats.bis.org/statx/srs/table/d5.1?f=pdf.

[5] Björk, T. 2019. *Arbitrage Theory in Continuous Time*, 4th ed. Oxford: Oxford University Press.

[6] Borchani, H., Pawlak, W. Holmslykke, S. & Engsig-Karup, A. Data-driven American Option Pricing using Artificial Neural Networks. *ICDM*, (July, 2019). Available at: https://www.researchgate.net/publication/336987698_Data-driven_American_Op tion_Pricing_using_Artificial_Neural_Networks.

[7] Brigo, D. & Mercurio, F. 2007. *Interest Rate Models*. Berlin Heidelberg: Springer Finance.

[8] Carlini, N. & Wagner, D. Towards Evaluating the Robustness of Neural Networks. *2017 IEEE Symposium on Security and Privacy (SP)*, (22 Mar, 2017). Available at: https://doi.org/10.48550/arXiv.1608.04644.

[9] Cho, K., Chung, J., Gulcehre, C. & Bengio, Y.. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop*, (11 Dec, 2014). Available at: https://doi.org/10.48550/arXiv.1412.3555.

[10] Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop*, (11 Dec, 2014). Available at: https://doi.org/10.48550/arXiv.1412.3555.

[11] Du, T., Ji, S., Shen, L., Zhang, Y., Li, J., Shi, J., Fang, C., Yin, J., Beyah, R. & Wang, T. Cert-RNN: Towards Certifying the Robustness of Recurrent Neural Networks. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, (13 Nov, 2021). Available at: https://doi.org/10.1145/3460120.3484538.

[12] Durarte, D. *QuantLib-Python Object Building Documentation*, 2020. Available at: https://quantlib-python-docs.readthedocs.io/en/latest/.

[13] Ferguson, R, & Green, A.D. Deeply Learning Derivatives. *CompSciRN: Artificial Intelligence*, (Oct 14, 2018). Available at: https://doi.org/10.2139/ssrn.3244821.

[14] Givens, G, & Hoeting, J. 2013. *Computational Statistics*. Hoboken, New Jersey: John Wiley Sons, Inc.

[15] Gnoatto, A., Reisinger, C. & Picarelli, A. Deep xVA Solver – A Neural Network Based Counterparty Credit Risk Management Framework. *CompSciRN: Computational* (Nov 5, 2021). Available at: http://dx.doi.org/10.2139/ssrn.3594076.

[16] Goodfellow, I., Bengio, Y., & Courville, A. 2017. *Deep*

learning. Cambridge, Mass: The MIT Press.

[17] Graves, A. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin: Springer.

[18] Green, A. 2016. *XVA: Credit, Funding and Capital Valuation Adjustments*. Chichester: John Wiley & Sons Ltd.

[19] Gregory, J. 2020. *The xVA challenge: Counterparty Risk, Funding, Collateral, Capital and Initial Margin*. Chichester: John Wiley & Sons Ltd.

[20] Gregory, J. 2010. *Counterparty Credit Risk: The New Challenge for Global Financial Markets*. Chichester: John Wiley Sons.

[21] Hammer, B. On the Approximation Capability of Recurrent Neural Networks. *Neurocomputing*, 31(1–4):107–123, (Mar, 2000). Available at: https://doi.org/10.1016/S0925-2312(99)00174-5.

[22] Heckinger, R., Ruffini, I. & Wells, K. *Understanding Derivatives - Markets and Infrastructure, Chapter 3 - Over-the-Counter (OTC) Derivatives*. Federal Reserve Bank of Chicago, 2014. Available at: https://www.chicagofed.org/publications/understanding-derivatives/index.

[23] Huang, J., Chai, J. & Cho, S. Deep Learning in Finance and Banking: A Literature Review and Classification. *Front. Bus. Res. China* 14, 13 (2020). Available at: https://doi.org/10.1186/s11782-020-00082-6.

[24] Hull, J. 2012. *Options, Futures, and Other Derivatives*, 10th ed. Upper Saddle River, N.J.: Pearson/Prentice Hall.

[25] Hutchinson, J.M., Lo, A.W. & Poggio, T. A Nonparametric Approach to Pricing and Hedging Derivative Securities via Learning Networks. *The Journal of Finance*, (Jul, 1994): 49(3):851–889. Available at: https://doi.org/10.1111/j.1540-6261.1994.tb00081.x.

[26] IBM Innovation Explanations. *The man who challenge AI every day*. IBM. Available at: https://www.ibm.com/thought-leadership/innovation-explanations/

pin-yu-chen? lnk=ushpv18r1.

[27] Katz, G., Barrett, C., Dill, D., Julian, K. & Kochenderfer, M. Towards Proving the Adversarial Robustness of Deep Neural Networks. *Electronic Proceedings in Theoretical Computer Science* 257. 10.4204/EPTCS.257.3., (8 Sep, 2017). Available at: https://doi.org/10.48550/arXiv.1709.02802.

[28] Kingma, D. & Lei Ba, J. Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations*, (Jul, 2015). Available at: https://doi.org/10.48550/arXiv.1412.6980.

[29] Ko, C-Y., Lyu, Z., Weng, T-W., Daniel, L., Wong, N. & Lin, D. POPQORN: Quantifying Robustness of Recurrent Neural Networks. *ICML*, (Jun, 2019). Available at: https://doi.org/10.48550/arXiv.1905.07387.

[30] Lindholm, A., Wahlström, N., Lindsten, F., & Schön, T. 2022. *Machine Learning: A First Course for Engineer and Scientists*. Cambridge: Cambridge University Press.

[31] Mangal, R., Nori, A.V. & Orso, A. Robustness of Neural Networks: A Probabilistic and Practical Approach. *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, (15 Feb, 2019): 93-96. Available at: https://doi.org/10.48550/arXiv.1902.05983.

[32] Marsland, S. 2015. *Machine learning, An Algorithmic Perspective*, 2nd ed. Chapman & Hall, CRC Machine Learning Pattern Recognition.

[33] Prenio, J & Jeffery Y. *FSI Insights on policy implementation No 35. Humans keeping AI in check – emerging regulatory expectations in the financial sector*. Financial Stability Institute, Bank for International Settlements, Aug, 2021. Available at: https://www.bis.org/fsi/publ/insights35.pdf?fbclid=IwAR2wd2L4FU AqDbxUR ePvoahJua8rSXBSHJFDC9l19ptZKsKSCtL2O506sZw

[34] Ruf, J. & Wang, W., Neural Networks for Option Pricing and Hedging: A Literature Review. *Journal of*

Computational Finance, (Nov 13, 2019). Available at: http://dx.doi.org/10.2139/ssrn.3486363.

[35] Sengputa, B. & Friston, K. How Robust are Deep Neural Networks? *ArXiv abs/1804.11313*, (30 Apr, 2018). Available at: https://doi.org/10.48550/arXiv.1804.11313.

[36] She, J-H. & Grecu, D. Neural Network for CVA: Learning Future Values. *Economics of Networks eJournal*, (Nov 6, 2018). Available at: https://doi.org/10.48550/arXiv.1811.08726.

[37] TensorFlow. *TensorFlow API Documentation*, 2021. Available at: https://www.tensorflow.org/api_docs.

[38] Welack, S., Artificial Neural Network Approach to Counterparty Credit Risk and XVA. *ERN: Credit Risk*, (Jan 9, 2019). Available at: http://dx.doi.org/10.2139/ssrn.3312944.

[39] Weng, T-S., Zhang, H., Chen, P-Y., Yi, J., Su, D., Gao, Y., Hsieh, C-J. & Daniel, L.. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. *ArXiv abs/1801.10578*, (31 Jan, 2018). Available at: https://openreview.net/pdf?id=BkUHlMZ0b.

[40] Yu, F., Qin, Z., Liu, C., Zhao, L., Wang, Y. & Chen, X. Interpreting and Evaluating Neural Network Robustness. *ArXiv abs/1905.04270*, (10 May, 2019). Available at: https://dl.acm.org/doi/abs/10.5555/3367471.3367625.

[41] Zheng, S., Song, Y., Leung, T. & Goodfellow, I. Improving the Robustness of Deep Neural Networks via Stability Training, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (15 Apr, 2016): 4480-4488. Available at: https://doi.org/10.1109/CVPR.2016.485.

[42] Zhu, S., Chan, J. & Bright, D., Applying Machine Learning for Troubleshooting Credit Exposure and xVA Profiles. *Risk Management & Analysis in Financial Institutions eJournal*, (June 16, 2019). Available at: https://doi.org/10.2139/ssrn.3404863.

Master's Theses in Mathematical Sciences 2022:E31
ISSN 1404-6342

LUTFMS-3441-2022

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/