
# Evaluation of Transformer-Generated Proxy Credit Default Swap Spreads

A collaboration with Svenska Handelsbanken AB

Master's thesis in Complex Adaptive Systems

Marcus Banér

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

# Evaluation of Transformer-Generated Proxy Credit Default Swap Spreads

A collaboration with Svenska Handelsbanken AB

MARCUS BANÉR


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2024

Evaluation of Transformer-Generated Proxy Credit Default Swap Spreads
A collaboration with Svenska Handelsbanken AB
MARCUS BANÉR



Supervisor: Richard Henricsson, Handelsbanken Capital Markets
Examiner: Mats Granath, Department of Physics

Master's Thesis 2024
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Evaluation of Transformer-Generated Proxy Credit Default Swap Spreads
A collaboration with Svenska Handelsbanken AB
MARCUS BANÉR
Department of Physics
Chalmers University of Technology

# Abstract

Following the 2007-2008 financial crisis, accurately assessing Counterparty Credit Risk (CCR) has become very important in the financial industry, especially in the Over-The-Counter (OTC) derivatives market. Credit Valuation Adjustment (CVA) integrates CCR into the pricing of OTC derivatives like Credit Default Swaps (CDSs) by using precise Probability of Default (PD) estimations, typically derived from CDS spreads.

When no liquid CDS spreads are available one uses proxy spreads instead. Traditional linear regression models for generating these, like the Nomura cross-sectional model, are commonly used in the industry but show problems in certain market environments. Therefore, this thesis evaluates the effectiveness of a Machine Learning (ML) model known as the Transformer for generating proxy CDS spreads and compares its outputs to the Nomura model.

Using actual, liquid market data for Western European financial companies, the Transformer model was trained to generate proxy spreads for five credit rating categories: AA, A, BBB, BB, and B. Results indicate that the Transformer model significantly outperforms the Nomura model, particularly in higher-rated categories, by generating spreads that are better aligned with the liquid market spreads they aim to simulate. Despite the promising results, further tests and analyses are needed to confidently be able to declare the Transformer model's superiority and potentially take it to production. This includes hyperparameter optimization, data diversification, and model interpretability improvements.

*Keywords:* Machine Learning, Transformer, Credit Default Swap, Proxy Spread, Counterparty Credit Risk

Utvärdering av Transformer-Genererade Proxy-Kreditswappspreadar
Ett samarbete med Svenska Handelsbanken AB
MARCUS BANÉR
Institutionen för Fysik
Chalmers Tekniska Högskola

# Sammanfattning

Efter finanskrisen 2007-2008 har en noggrann bedömning av motpartsrisk (CCR) blivit väldigt viktigt inom den finansiella sektorn, särskilt på marknaden för icke-standardiserade (OTC) derivat. Kreditvärderingsjustering (CVA) integrerar CCR i prissättningen av OTC-derivat som kreditdefaultswappar (CDSer) genom att använda uppskattningar av sannolikheten för konkurs (PD), vanligtvis härledda från CDS-spreadar.

I fall då det saknas likvida CDS-spreadar använder man sig istället av proxy-spreadar. Traditionella linjära regressionsmodeller för att generera dessa, som Nomuras tvärsnitssmodell, används ofta inom branschen men denna är problematisk under vissa marknadsförhållanden. Därför utvärderar denna uppsats effektiviteten av en maskininlärningings-(ML)-modell känd som Transformern för att generera proxy-CDS-spreadar och jämför dennes resultat med Nomura-modellen.

Med hjälp av likvid marknadsdata för Västeuropeiska finansbolag tränades Transformer-modellen för att generera proxy-spreadar för fem kreditbetygskategorier: AA, A, BBB, BB och B. Resultaten indikerar att Transformer-modellen avsevärt överträffar Nomura-modellen, särskilt i kategorier med högre betyg, genom att generera proxy-spreadar som bättre återspeglar de likvida marknadsspreadar som de ska simulera. Trots de lovande resultaten behövs ytterligare tester och analyser för att med säkerhet kunna fastslå Transformerns överlägsenhet och för att eventuellt kunna försätta den i produktion. Detta inkluderar optimering av hyperparameter, datadiversifiering samt förbättringar av modellens tolkbarhet.

*Nyckelord:* Maskininlärning, Transformer, Kreditdefaultswapp, Proxy-Spread, Motpartsrisk

# Acknowledgements

I would like to start by thanking my supervisor at Svenska Handelsbanken (SHB), Richard Henricsson, for his help during this project. I also want to thank my fellow SHB thesis student, Gustav Karlsson, for good discussions and cooperation during this spring. Moreover, I would like to thank previous master thesis student at SHB, Johan Luhr, for taking the time to explain his work. Lastly, I would like to express my gratitude to my family and friends for their support during my years at university. Thank you very much.

Marcus Banér, Gothenburg, June 2024

# List of Acronyms

Below the acronyms that have been used throughout this thesis are listed in alphabetical order:

| | |
|---|---|
| AI | Artificial Intelligence |
| CCR | Counterparty Credit Risk |
| CDS | Credit Default Swap |
| CNN | Convolutional Neural Network |
| CSV | Comma-Separated Values |
| CVA | Credit Valuation Adjustment |
| DNN | Deep Neural Network |
| EBA | European Banking Authority |
| EE | Expected Exposure |
| GPU | Graphics Processing Unit |
| GUI | Graphical User Interface |
| LSTM | Long Short-Term Memory |
| LGD | Loss Given Default |
| MAE | Mean Absolute Error |
| ML | Machine Learning |
| MSE | Mean Squared Error |
| MHA | Multi-Headed Attention |
| MTM | Market-To-Market |
| NLP | Natural Language Processing |
| OOS | Out-Of-Sample |
| OTC | Over-The-Counter |
| PD | Probability of Default |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |
| RQ | Research Question |
| SGD | Stochastic Gradient Descent |
| SHB | Svenska Handelsbanken |

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

This chapter will provide an introduction regarding the importance and relevance for conducting this thesis. Starting off, a background to the problem at hand will be discussed along with its importance for Svenska Handelsbanken (SHB). Moreover, a link to previous work conducted on the topic and what gaps these have left for future work is provided. Building on this, the aim of the thesis will be presented and its limitations discussed, resulting in a specification of the primary issues that have been investigated.

## 1.1 Background

In this section, an overview of the problems which this thesis addresses is provided. It starts by explaining why the need for accurate proxy CDS spreads are essential from a risk point of view and challenges encountered by traditional models for obtaining them. Moreover, the intersection between the machine learning (ML) domain and proxy CDS spread generation is discussed as well as how this thesis directly links to previous student projects at SHB.

### 1.1.1 The Importance of Accurate CDS Spreads

All major investment banks involved in the Over-The-Counter (OTC) derivatives market are heavily dependent on accurate depictions of the probability for counterparties being able to fulfill their commitments versus the bank. Essentially what this boils down to, is that in order to ensure a bank's robustness in periods of market instability, a necessary amount of capital must be allocated for protection against potential losses following counterparty defaults. The 2007-2008 financial crisis, with the collapse of major institutions such as Lehman Brothers [16], demonstrates a recent event during which efforts to this end proved insufficient. This has led to considerable efforts being taken in order to prevent similar events from occurring in the future.

A Credit Default Swap (CDS) is a type of financial contract that serves as insurance against the risk that a borrower will default on their obligations (further described in Section 2.1.1). Essentially, it allows one counterparty (the buyer) to pay another counterparty (the seller) a series of payments (called premiums) to protect against credit risk posed by a third party. The risk that for example the seller of a CDS is unable to uphold its contractual agreement is known as Counterparty Credit Risk

(CCR). This refers to the possibility that this counterparty which is involved in the CDS transaction might not fulfill its payment obligations. CCR is a critical concern in the over-the-counter (OTC) derivatives market, where vast sums are traded based on the promise of future payments. Linking to CCR, the concept of Credit Valuation Adjustment (CVA) comes into play. CVA represents an adjustment to the value of a financial derivative such as a CDS to reflect the risk of counterparty default. Thus, CVA essentially puts a price on the CCR, allowing institutions to account for the risk that the counterparty to a derivative might not uphold their end of the arrangement (CCR and CVA are both detailed in Sections 2.1.2 and 2.1.3).

Given these definitions, the accuracy of CDS spreads becomes important. A CDS spread represents the annual cost (expressed in basis points) that a buyer must pay to the seller to insure against the default of a reference entity (such as a corporation or government). For example, a CDS spread of 100 basis points means the buyer pays 1% of the notional amount per year (further described in Section 2.1.1). Spreads directly influence the CVA calculation (and therefore perception of the CCR) by indicating the market's current view on the Probability of Default (PD) for the counterparty of the transaction. An accurate CDS spread is thus essential for effective risk management, ensuring that financial institutions can appropriately hedge against potential defaults and make precise adjustments to the valuation of their derivative positions to reflect the risk of counterparty failure.

Liquid, counterparty-specific (so called single name) CDS spreads are important for financial institutions as they provide a clear, market-driven indicator of the perceived risk associated with a specific entity. These spreads are directly proportional to the PD, offering immediate insight into the financial health and stability of a counterparty. However, not all counterparties have actively traded CDS contracts, leading to a lack of liquid, single name CDS spreads. In these cases, proxy CDS spreads are used as an alternative metric to estimate the counterparty risk of such entities. As has been determined in the Basel III accord (which is an international regulatory framework for banks), proxy curves are to be constructed by analyzing the CDS spreads of similar liquid entities, considering common variables like industry sector, credit rating, and geographical region [11] [20]. These proxies allow financial institutions to approximate the CDS spreads for entities lacking liquid market spreads, which ensures a continuous ability to accurately calculate CVA in order to hedge against CCR, and comply with regulatory requirements.

### 1.1.2 Challenges with Traditional Models

Traditionally, linear regression models have been used to generate proxy CDS spread curves. One such model is the cross-sectional regression model introduced by the global financial services group Nomura [31] in 2013 (typically called the Nomura model, further described in Section 2.3.1). While this model has gained popularity and become common for generating proxy CDS spreads within the financial industry, it has proved problematic during periods of market volatility, resulting in inaccurate proxies.

### 1.1.3 Advancements in Machine Learning

As illustrated in a series of papers by Brummelhuis & Zhongmin ([12], [13], [15] and [14]), using non neural network based machine learning (ML) techniques improves on the accuracy of estimating proxy CDS spreads. Previous thesis projects at SHB have moreover showed that deep neural network (DNN) based ML-techniques also show great potential in improving the accuracy of the generated proxies, for example by using Recurrent Neural Networks (RNNs) [18] or Transformers [29].

### 1.1.4 Building on Previous Work

This thesis builds directly on the work presented in the 2023 thesis conducted in collaboration with SHB by Johan Luhr [29] in which it was shown that Transformers inspired by the original architecture presented in the 2017 paper *Attention is All you Need* [40] can be used to generate proxy CDS spreads. While the main outcome of that thesis was the proven ability of Transformer models to generate proxy CDS spreads, it did not go the full length in evaluating the performance of the proxy spreads against the actively traded liquid market spreads they aim to simulate. This however, along with a thorough comparison between Transformer generated proxy spreads and proxy spreads generated by traditional models, is where this thesis takes off.

## 1.2 Aim

The primary aim of this thesis has been to improve the evaluation of proxy CDS spreads generated by Transformer models, such as the ones constructed in [29], using market spreads from actively traded liquid counterparties. Should the Transformer-generated proxy spreads show close alignment with these liquid market spreads, especially compared to proxy spreads generated by traditional models, this would indicate a high model reliability. A successful evaluation could lead to more confidence surrounding the supposed superiority of Transformer models compared to traditional linear regression models for the purpose of generating accurate proxy CDS spreads. This outcome could also provide additional value for SHB since more accurate CDS spreads provide a better foundation for accurately perceiving CCR.

## 1.3 Limitations

The most apparent limitation of this thesis is that it has solely employed a Transformer model when producing proxy CDS spread curves, as opposed to other ML-architectures such as the RNN models used in [18]. Moreover, considering that this is a master's thesis supposed to be carried out during the course of one semester, an important limitation is the time-constraint of the project. As a consequence of this, some interesting challenges has been excluded given time and resources availability. Among other things, this includes:

- Expanding the data sets beyond certain sectors, regions and ratings.

- Extensive embedding analysis.
- Extensive hyperparameter optimization.
- Incorporating the models into SHB's systems or creating additional extensions such as a graphical user interface (GUI).

All these challenges are interesting and important, but they have been left for future work.

## 1.4 Specification of the Issue Being Investigated

The core objective of this thesis is to thoroughly evaluate proxy CDS spread curves generated by a Transformer model in order to determine if they propose a better alternative to traditionally used linear regression models, specifically the cross-sectional Nomura model [31], in simulating liquid market spread curves. The evaluation in this thesis thus seeks to answer the following research question (RQ):

- RQ: To what extent can Transformer-generated proxy CDS spread curves more accurately simulate market based CDS spread curves, compared to those produced by traditional linear regression models?

Given the context provided in the previous sections, including outcomes of previous works and financial context, the evaluation process of the Transformer generated proxy CDS spread curves will be centered around two primary data sources:

- Evaluation utilizing data on actively traded liquid market CDS spreads.
- Evaluation utilizing data on proxy CDS spreads generated by traditional models.

## 1.5 Methodology

Similar to most quantitative research, including previous theses at SHB ([29] and [18]), this project has followed the hypothetico-deductive method. This is a scientific approach that begins with an hypothesis or theoretical premise from which deductions are made. The deductions are then tested through empirical observation or experimentation. If observations align with the deductions, the hypothesis gains support; if not, it may be revised or rejected [39]. In this thesis, the hypothesis has been the research question. The primary data sources provided the basis for generating deductions, which involved creating Transformer-generated proxy CDS spreads and comparing them with these primary data sources. This process details how the theory was tested against empirical data. The workflow followed can be divided into the following subchallenges:

- *i.* Information collection.

- *ii.* Data collection and preprocessing.
- *iii.* Transformer modelling.
- *iv.* Evaluation of model output.

*Step i.* outlines the theoretical backbone of the thesis. Considering the complex nature of both the Transformer model as well as the financial context in which it has been applied, a lot of effort during the initial weeks of the project was spent gathering literature on these topics. The research of relevant evaluation methods was also an important part of this pre-study.

*Step ii.* is straightforward and involved gathering, sorting and preprocessing the data used in the training, testing and evaluation steps respectively. Two main categories of data have been handled in this thesis. The first include time-series data of counterparty specific, liquid (so called single name) CDS spreads. The second is time-series data of proxy CDS spreads generated by a traditional linear regression model (the so called Nomura model). An important note on this matter is that all the data in this project has been provided by SHB and is a commodity, which means no data will be shared publicly.

*Step iii.* involved putting a Transformer model which can handle the gathered time-series data effectively to use. The model that was used resembles the encoder-decoder model constructed in the 2023 SHB thesis by Johan Luhr [29]. Tools utilizing various ML-platforms in Python were used for this purpose, mainly the TensorFlow library [37] with the Keras API [36].

*Step iv.* was the most important step for this thesis since this is where the evaluation of the Transformer output took place. This step has included performing visual and statistical evaluations employing the primary data sources specified earlier, namely data on liquid, market CDS spreads and proxy CDS spreads generated by a traditional linear regression model. Out-of-sample (OOS) testing was utilized, where unseen data samples of real world market CDS spreads were compared to the model generated proxy spreads in order to determine model accuracy.

*Steps i.-iv.* have required an iterative procedure in alignment with the hypothetico-deductive method [39].

Chapters 3 and 4 will provide details regarding all of the above steps.

## 1.6 Report Guide

The thesis has the following chapter outline, made in conjunction with the steps presented in the previous section. Chapter 2 presents the theoretical framework necessary for understanding the financial, machine learning and statistical concepts under consideration. Chapter 3 describes the methodology surrounding the thesis, including information collection, data handling and conducted experiments. Chap-

ter 4 details the Transformer modelling that has been done for generating proxy CDS spreads. Chapter 5 presents the results and analysis of the conducted experiments. Chapter 6 involves a discussion regarding the outcomes of the results, and Chapter 7 brings the thesis to a close by presenting conclusions, ideas for future work and some final reflections.

# 2
# Theoretical Framework

This chapter aims at familiarizing the reader with the context in which this thesis is set by detailing relevant theoretical concepts. Initially the financial background will be outlined, focusing on the relevance for modelling accurate proxy CDS spreads. Then, important machine learning (ML) topics will be described with a key emphasis put on the Transformer architecture. Moving on, an overview of previous work done in the field of proxy CDS spread generation will be described and an emphasis will be put on how this thesis is directly connected to previous theses that have been done in collaboration with SHB. Lastly, some statistical background will be provided to introduce the reader to some important metrics that have been used. The theoretical framework presented here will thus make the subsequent chapters of the thesis more easily digestible from both a fundamental as well as holistic point of view.

## 2.1 Financial Context

Below three major financial concepts are described: the Credit Default Swap, Counterparty Credit Risk and Credit Valuation Adjustment. Knowing about these topics and their interplay is important for understanding why the work conducted in this thesis is of relevance to SHB.

### 2.1.1 Credit Default Swap

A derivative is a financial contract where opposite parties (the buyer and seller) enters into an arrangement to either make payments or to buy or sell an underlying security at a specific future time or times. Often derivatives are used as a tool to hedge against certain risks, such as an airline wanting to lock in a certain fuel price. Simpler derivatives with high trading volumes and liquidity, like options or futures, are usually traded through a centralized exchange which facilitates rapid and structured trades. Exchanges guarantees clearing and provides efficient price discovery and thereby reduces certain risks, such as the default of a counterparty of a transaction. More complex and less standardized derivatives contracts are typically not tradable via exchanges, but are instead private contracts bilaterally traded between two entities. These are known as Over-The-Counter (OTC) derivatives. Without the clearing function of an exchange, opposing parties in an OTC-derivative contract assumes a greater risk of the default of the other (further discussed in Subsection

2.1.2 below). Typically the OTC-derivatives market is subject only to a few major players such as corporate banks, big corporations and large financial institutions [20].

The credit default swap (CDS) is a kind of OTC-derivative [20]. As with most derivative instruments, a CDS has several layers of complexity which will not be covered here. Instead, an overview of the CDS structure will be presented and interested readers are referred to the cited references for more details about CDS contracts. Figure 2.1 below provides a simplified scheme of the CDS mechanism.


**Figure 2.1:** Simplified credit default swap (CDS) mechanism. Illustration created by the author using PowerPoint.

As Figure 2.1 aims to illustrate, a CDS contract is entered bilaterally between two entities: the CDS buyer and the CDS seller. Typically, the CDS Buyer is involved in some kind of arrangement with another entity (the reference entity) for which the CDS buyer carries a credit risk (i.e. default risk of the reference entity). The CDS buyer thus wishes to offset or "swap" this risk with an unrelated third party, which in this scenario would be the CDS issuer (or seller). In this scheme, the CDS buyer would pay recurrent premiums to the CDS seller for the promise that the seller reimburses the buyer in case the reference entity would default on its outstanding arrangement with the buyer. This scheme makes the buyer and seller so called counterparties in the CDS transaction. Effectively, a CDS contract protects (or "hedges") the buyer from possible losses that otherwise could occur in the event of a default. A situation where the protection offered by the CDS contract is enforced is commonly referred to as a credit event [22]. To be noted, this arrangement also institutes what is known as counterparty credit risk (CCR) which is the risk that the seller of the CDS will be unable to reimburse the buyer in the time of a credit event (more on CCR in Section 2.1.2).

The premium is determined via the so called "spread" which is the pricing component of a CDS expressed as a rate in basis points (bps) per annum of the notional amount of the contract. One bps is equal to one one hundredth of a percent. So if for example a CDS is quoted with a spread of 100 basis points on a notional of $10 million, the annual premium would be $100,000 ($10M x 100bps = $10M x 0.01 = $100K). Depending on the time the buyer wishes to hedge its position it can enter into CDS contracts of different tenors (times until maturity).

### 2.1.2 Counterparty Credit Risk

Counterparty Credit Risk (CCR) is the risk that an entity with whom one has entered into a derivatives contract will fail to fulfil their side of the contractual agreement. In practice this usually takes shape in the form of a default of the opposing entity (counterparty) in question, meaning it will not be able to pay the credit it is contractually obliged to, resulting in a credit risk for the institution on the other side of the contract. Typically, CCR is associated with the OTC-derivatives market, including products such as CDSs, interest rate swaps and securities financing transactions (for instance repurchase agreements) [20].

A good way of understanding CCR is to demonstrate how it differs from traditional credit risk, or lending risk as it is more appropriately denoted. Lending risk occurs when one party owes some amount to another party and fails to uphold the agreement to repay the loaned amount. It is applicable to many underlying assets, some common ones being mortgages, credit cards or bonds. Determining lending risk is quite straightforward. First of all, the notional amount at risk at any point in time is relatively known. For example this could be the outstanding balance for a mortgage on a house accounted for amortization, or the debt on a credit card which typically has a maximum usage facility. Secondly, lending risk is unidirectional meaning that only one party assumes the risk. This could for instance be demonstrated through a bond contract in which the holder assumes a significant credit risk, whereas the issuer faces no risk in the event that the buyer defaults [20].

CCR on the other hand differs from these dynamics. Most importantly, the future value of the contract is very uncertain due to its connection to underlying market variables. Take an interest rate swap for example. Its value is linked to how interest rates evolve over the term of the swap, meaning it is dictated by market behavior. This makes it difficult to confidently estimate its value at future points in time. Furthermore, CCR is a bilateral risk measure which means both parties of the contract (for example the buyer and seller of a CDS contract) faces the risk of loss. This is because derivatives often involves the exchange of payments over time based on the fluctuating behavior of the underlying asset. A consequence of this is that the value of the derivative contract can be both positive or negative at different times, meaning that each party is potentially both a creditor and a debtor at different points during the life of the contract [20].

As was pointed out in Chapter 1 it is crucial for institutions dealing with these kinds of financial instruments to effectively manage CCR. Common mitigation strategies include *collateral agreements* where cash or securities are posted against Market-To-Market (MTM) losses, *netting* which is arrangements allowing parties to combine or offset mutual obligations to reduce the total number of transactions and thereby minimize the CCR exposure, and *hedging* which is typically done via instruments such as the CDS [20] (see Section 2.1.1 for details about the CDS) . Most importantly, CCR management is a matter of ensuring financial stability. This is because defaults of entities with the size of the ones usually involved in these kinds of trades can have cascading effects across the financial system, something the 2007-2008 fi-

nancial crisis clearly showed with the ultimate collapse of major investment banks such as Lehman Brothers [16]. Institutions have to ensure they can withstand the default of a counterparty of a transaction without undergoing severe financial distress themselves.

### 2.1.3 Credit Valuation Adjustment

In practice, institutions operating in the OTC-derivatives market use what is known as Credit Valuation Adjustment (CVA) to account for CCR. This is required both by globally accepted accounting standards (such as IFRS 13) and for capital allocation purposes (as determined by Basel III) [20]. CVA is best though of as the quantification, or pricing, of the CCR associated with an OTC-derivative position and thus crucial for having a correct valuation of the position in the company's balance sheet. Originally, CVA's contribution to the valuation is expressed by Equation 2.1:

$$\text{Risky value} = \text{Risk-free value - CVA} \tag{2.1}$$

The risk-free value represents the value of the contract assuming no risk of counterparty default and CVA represents the adjustment made to account for this risk. The risk-free value is typically positive or zero and as Equation 2.1 shows CVA is subtracted from this value. As a result, the net valuation after accounting for CVA can be negative if the CVA is substantial, which would indicate a net liability rather than an asset. As Equation 2.1 shows, the Risk-free value and CVA components of a derivatives contract can be completely separated, meaning that each can be treated on its own. First of all, this means that the responsibilities of the valuation process can be divided between different desks within an institution, with each focusing on their respective part [20]. Moreover, it makes the outline of each component's construction easier to define. Moving on, only the CVA aspect of Equation 2.1 will be detailed since this part relates to CDS spreads.

Looking closer at the CVA component of Equation 2.1, one may start by considering the formula for CVA computation. This is presented in Equation 2.2 below:

$$CVA = LGD \sum_{i=1}^{m} EE(t_i) \text{ x } PD(t_{i-1}, t_i) \tag{2.2}$$

In this formula, LGD stands for Loss Given Default and it is the exposure expected to be lost in case the counterparty (for example the issuer of a CDS contract) defaults. LGD is expressed as a percentage and it is the complement of the so called Recovery Rate of the transaction. Say for instance that one is expected to recover 60% on a transaction given a default, then the LGD would be 1 - 0.6 = 0.4 = 40%. EE stands for Expected Exposure and this is the estimated amount one might be exposed to or stand to lose at different future points in time if the counterparty defaults. Essentially, it is like looking into the future and guessing how much capital you could stand to lose over time. PD stands for Probability of Default and it expresses the likelihood that the counterparty defaults within the time period of consideration ($t_{i-1}$ til $t_i$ in Equation 2.2) [20]. Considering the time-dependency of the EE and PD components of Equation 2.2, the equation must be integrated

over time rather than simply using the respective components' averages in order to account for the precise distributions of the PD and EE. [20]. An important implication of Equation 2.2 is the direct proportionality between the LGD, EE and PD components to the CVA computation. Especially interesting for the purposes of this thesis is the PD component and its connection to the CDS spread, which will be further explained below.

As with all components of the presented topics so far, PD can be broken down into smaller and more detailed pieces of construction. However, in order to preserve only the necessary level for this thesis, the analysis will be kept only to the essentials for the context for which it is set (proxy CDS spread generation). Curious readers can find more details in the provided references, especially [20] which offers clear and concise explanations of these concepts. One important distinction when dealing with PD is the one between real-world and risk-neutral PD. Real-world PD is estimated from historical data associated with some credit rating. Risk-neutral PD on the other hand is market implied and derived from financial instruments such as the CDS [20]. From here on, this is what is meant by PD. Recall from Section 2.1.1 that an important component for CDS contracts is the so called spread, which is a rate used for computing the premiums paid by the buyer to the seller for the duration of the contract. Apart from this, the CDS spread also influences the computation of CVA (and thus pricing of CCR). Equation 2.3 below shows a simplified approximation used for determining PD between two sequential dates when for example quantifying a term such as CVA [20]:

$$PD(t_{i-1}, t_i) \approx exp(-\frac{s_{t_{i-1}} t_{i-1}}{LGD}) - exp(-\frac{s_{t_i} t_i}{LGD}) \tag{2.3}$$

In the approximation of PD shown in Equation 2.3, $s_t$ represents the CDS spread at time $t$. The terms $t_{i-1}$ and $t_i$ represents the start and end of the interval for which the PD is computed. The exponential functions (which yields outputs in the range $[0, 1]$ for real-valued inputs) then approximates the survival probabilities up to times $t_{i-1}$ and $t_i$ respectively, whereas the difference between these survival probabilities gives the probability of default within the interval $[t_{i-1}, t_i]$. For example, suppose that the spread level $s_t$ is 100 bps (=1%), the time periods are $t_{i-1} = 1$ year and $t_i = 2$ years and the LGD is 40%. Following the approximation by Equation 2.3, the scaled spreads would be for $t_{i-1} = 1$ year: $(\frac{0.01x1}{0.4}) = 0.025$, and for $t_i = 2$ years: $(\frac{0.01x2}{0.4}) = 0.05$. The exponential terms become: $exp(-0.025) \approx 0.9753$, and $exp(-0.05) \approx 0.9512$ respectively. The approximated PD between $t_{i-1}$ and $t_i$ then becomes: 0.9753 - 0.9512 = 0.0241=2.41 %.

As can be seen, the PD is proportional to the spread in such way that an increase in the spread will lead to an increase in PD and vice versa. Merging Equation 2.3 and Equation 2.2 one gets the following expression to approximate CVA [20]:

$$CVA = LGD \sum_{i=1}^{m} EE(t_i) \text{ x } (exp(-\frac{s_{t_{i-1}} t_{i-1}}{LGD}) - exp(-\frac{s_{t_i} t_i}{LGD})) \tag{2.4}$$

What Equation 2.4 shows is that the CDS spread has a proportional impact on the computed CVA. The bottom line of this outline is that since CVA essentially puts a

price on the CCR, which is important for the valuation process of OTC-derivatives as shown in Equation 2.1, the accuracy of CDS spreads becomes important for institutions dealing in such instruments. The more accurate the spread, the better will be the accounted CVA and thus CCR, meaning institutions will have a more robust foundation for ensuring capital is properly allocated in the event of a counterparty of a transaction (such as a CDS issuer) defaulting on its obligations. Previous work on proxy CDS spread generation (recall that proxy spreads are used when no liquid market spreads are available) will be further discussed in Section 2.3.

## 2.2 Machine Learning Context

The purpose of this section is to familiarize the reader with some concepts from the machine learning (ML) domain needed to comprehend the work presented in this thesis. For this purpose, an introductory part on the ML subfield of deep learning (DL) is provided in order to sort out some fundamental ideas and topics in the field. Moving on, the concept and progression of sequence models is outlined. Finally, the Transformer architecture, which is the backbone of the ML-modelling in this thesis, is presented and detailed.

### 2.2.1 Deep Learning

Fundamentally, Artificial Intelligence (AI) can be though of as the ability of computers to perform human-like tasks such as reasoning, decision-making or problem solving. The AI landscape encompasses several different approaches and subfields whose use cases and recognition differs. Among the most popular AI subfields today is Machine Learning (ML). In practice, ML-algorithms involve using historical data (typically referred to as training data) to build a mathematical models that makes predictions without being explicitly programmed on how to do so [9]. A large subfield of ML is what is known as deep learning (DL). DL represents classes of algorithms that are inspired by the function and structure of the brain, known as neural networks. DL structures algorithms in layers to create an "artificial neural network" that can learn complex, nonlinear patterns in the input data and from this make intelligent decisions on its own [19]. The ability to handle large and complex datasets better than traditional algorithms has made DL a significant and very popular approach within the ML-community today.

#### 2.2.1.1 Artificial Neural Networks

An artificial neural network is a computer structure in which small units of computation (commonly denoted as neurons) are connected in large networks in order to process input data to produce an output. Compared to the neural network of the brain, the artificial version contains highly idealised units of computation. These units (or neurons) connects to each other by weights and the output of each connected neuron is the weighted average of its inputs [30]. This relationship is illustrated by Equation 2.5 below:

$$s_i(t+1) = g(\sum_{j=1}^{N} w_{ij} s_j(t) - \theta_i) \tag{2.5}$$

In Equation 2.5, $s_i$ is a neuron with index $i$ which receives $N$ inputs from other neurons $s_j$. The neurons $i$ and $j$ are connected via the weight $w_{ij}$. This weight determines the strength of the connection between the two neurons and thus how much influence the input from neuron $j$ will have on the output produced by neuron $i$. The term $t$ represents a discrete time step in the iterative process of updating the neuron's state. At each time step $t$ the state of neuron $i$ is updated based on the weighted inputs from other neurons $j$.

As Equation 2.5 shows, the connected neurons produce a weighted sum which is adjusted by some threshold value $\theta_i$ for neuron $i$. The output of each neuron is transformed using a non-linear activation function, $g(b)$, which can introduce complex patterns and behaviors. The general notation aims to show that an activation function can be of several types, with $b$ being the local field representing the threshold adjusted input [30]. Functions such as the Rectified Linear Unit (ReLU) have become the default, providing a simple yet effective way to introduce non-linearity [19]. In the field of DL, the artificial neural networks have one or several hidden layers (thus the notation "deep") which allows for more complex representations of the input data. This layering produces what is known as a deep neural network (DNN) and Figure 2.2 shows a very simple schematic of such a network with one hidden layer of neurons.

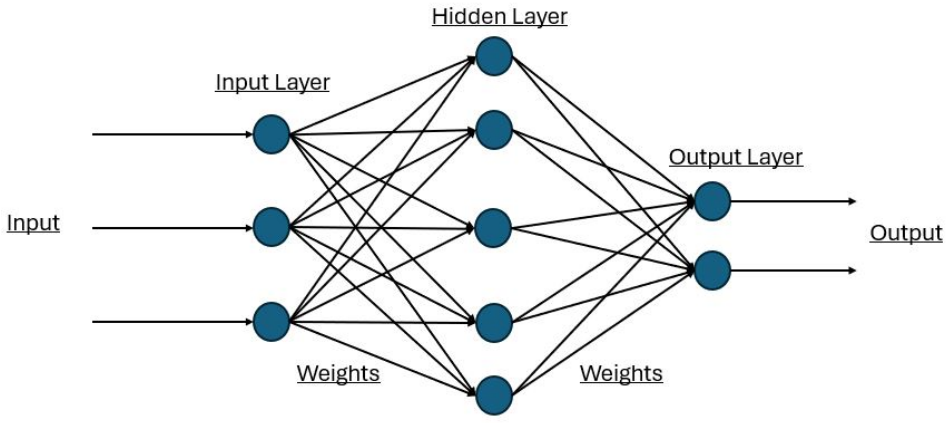


**Figure 2.2:** Simple schematic of a deep neural network (DNN). Illustration created by the author using PowerPoint, inspired by [30].

#### 2.2.1.2 Training and Optimizers

When training a DNN, a method known as backpropagation is employed which in very simple terms can be described as the process of adjusting the weights of the network (black arrows in Figure 2.2) backwards through the network based on the error of the network's predictions. When performing backpropagation during train-

ing, the network uses a loss function which computes the error between the predicted and the expected output of the network. Two common loss functions include Mean Squared Error (MSE) and Cross-entropy loss. MSE squares the difference between the current prediction and the expected output and divides this by the number of outputs (see Section 2.4.2 for details regarding MSE). Cross-entropy loss measures the difference between two probability distributions by calculating the total entropy (i.e. uncertainty or unpredictability) between the predicted probability distribution and the actual distribution and is often used in classification tasks to evaluate the performance of a model by penalizing predictions that differ from the true labels [32].

During training, the idea is to minimize the loss function (i.e. the error between predictions and expectations) so that the weights of the final trained network maps input to output as accurately as possible. For this purpose, one typically uses an optimizer with backpropagation. A common choice of optimizer is known as Stochastic Gradient Descent (SGD). In principle, SGD is quite straightforward and it follows Equation 2.6 below:

$$W_{t+1} = W_t - \eta \nabla J(W_t) \tag{2.6}$$

What Equation 2.6 shows is that the new weight states, $W_{t+1}$, will equal the old weight states, $W_t$, adjusted by the gradient of the loss function $J(W_t)$, with respect to the old weight states, $\nabla J(W_t)$. The term $t$ in this context represents the iteration step in the training process. Each iteration $t$ corresponds to an update of the weights based on the gradient of the loss function. For a randomly selected subset of data (thus "stochastic"), SGD computes the gradient of the loss function with respect to each weight and this gradient points in the direction of the steepest increase of the loss function. The gradient is multiplied by a learning rate, $\eta$, which controls how much the weights are adjusted with respect to the gradient [30].

Another optimizer which has become popular in recent years is Adam which stands for Adaptive Moment Estimation. While SGD is effective in many applications, the choice of the learning rate $\eta$ and the condition of the objective function $J(W_t)$ sometimes impacts its performance to a large extent. To address such issues, Adam computes adaptive learning rates for each parameter through the estimation of the first and second moments of the gradients. The resulting update rule takes into account and adapts the learning based on the average first moment (the mean) and the square root of the average second moment (the uncentered variance) to normalize the gradient. Detailed derivations of and explanations of Adam will not be covered here. It is enough to know that the introduction of these moment estimates allow Adam to adjust its learning rate based on the uncertainty (or variance) in the gradient, which makes it robust to variations in the gradient's magnitude which often leads to better performance on complex neural network architectures and datasets. Compared to SGD, Adam can also help in achieving faster training convergence and also require less tuning of the learning rate [26].

#### 2.2.1.3 Common Challenges in Deep Learning

In the context of DNN's, a problem that normally arises when successively performing backpropagation over several layers is exploding or vanishing gradients. Exploding gradients means that the gradients become too large which causes the training progress to diverge. Vanishing gradients on the other hand appears when gradients become too small, which causes the weight updates to become insignificant and that can lead to a slowdown or halt in the learning process of the network. A whole section could be devoted to this problem but for the purposes of this thesis it is enough to know that one typically uses some common techniques to the network architecture to tackle them. This for instance involves using non-saturating activation functions such as Rectified Linear Units (ReLU), sophisticated initialization methods, batch normalization and residual (skip) connections. An interested reader is referred to [19] and [30] for more in-depth explanations of these techniques.

Another problem in DL is the trained network's ability to generalize to unseen data. The common approach when training neural networks is to have one sample of data (the training set) which the network trains on (i.e. uses to update the weights). Then, another much smaller sample known as the validation set is used during training to see if the trained weights can perform accurate predictions on unseen data. The loss achieved on the validation set typically helps in determining whether the network has converged (reached a sufficient point of accuracy) or if further training is required. When the training is deemed completed, one passes an unseen test set (which again typically is much smaller than the training set) through it to determine the general performance of the network. Sometimes the network overfits which simply means that it learns the training set very well, but is poor at generalizing predictions to unseen data. In order to increase the generalization to unseen data, it is therefore common to employ regularization techniques [19]. On such technique is known as dropout which in practice means randomly setting some of the weight connections in the network to zero. What this results in is that the layers of the network are pointed to take more or less responsibility for the input to them by taking a probabilistic approach [41].

### 2.2.2 Sequence Models

As highlighted in the previous section, DNN's can have various architectures depending on the specific needs of the problem at hand. Convolutional Neural Networks (CNNs) for example is a kind of DNN architecture particularly useful in handling grid-like data such as images by applying what is known as convolution layers to the input [30]. Sequential data on the other hand, such as texts, speech or financial time series (important for the purposes of this thesis), requires different approaches considering the importance of order and dependency between elements. Sequence models is a type of DNN architectures designed to process sequences by maintaining a form of "memory" of previous inputs, thereby allowing them to capture time-dependent dynamics in the dataset [28].

The first simple form of a sequence model is a so called Recurrent Neural Network

(RNN). RNNs are used for sequential data processing, typically in areas such as Natural Language processing (NLP) and time series applications. Without going into the technical details of RNNs, which is outside the scope of this thesis, it is enough to know that an RNN feature shared weights across time which means the weights can maintain memory from previous inputs. This is achieved via an internal loop (which is a kind of "hidden layer" in the DNN architecture) which allows information to be passed from one step of the network to the next [28]. The use of RNNs is thus advantageous in applications where past information impacts future predictions and because they are simple and rather intuitive. However, RNNs suffer from the exploding (or vanishing) gradient problem discussed in Section 2.2.1 making it difficult for them to capture long-range dependencies in the data. Moreover, they are not parallelizable which makes them computationally inefficent and slow during training [33].

Due to the problem of handling long-range dependencies suffered by RNNs, another popular sequence model is the Long Short-Term Memory (LSTM) model. LSTMs build on the previously mentioned hidden layer of the RNN architecture by passing what is known as a cell state to the next time step. This cell state contains three so called gates: the forget gate which removes (or "forgets") irrelevant information from the cell state, the input gate which adds more important information to the cell state and the output gate which also adds more useful information to the cell state [24]. Without going into the technicalities of the gating mechanism, it is enough to understand that it allows an LSTM model to know when to forget, ignore or keep information from previous time steps [24]. The advantages of this, as mentioned, is that they can capture essential information over longer sequences compared to RNNs while also reducing vanishing or exploding gradients. However, since LSTMs are much more complex than RNNs, they become harder to train and require more computational resources. Moreover, similar to the RNN architecture, LSTMs process data sequentially which make them slow during training compared to if the data was to be processed in parallel [33].

Given the limitations mentioned with previously popular sequence models like RNNs and LSTMs, these have been challenged by another state-of-the-art DNN architecture that effectively mitigates many of these issues. This architecture is known as the Transformer. By combining parallel processing, an encoder-decoder structure and, perhaps most importantly, an attention mechanism, the Transformer has become more and more popular in various ML-applications, such as for example problems involving sequential data [28]. Below, a section on the original Transformer architecture is presented aimed at showcasing its functionality and advantages compared to other sequence models.

### 2.2.3 The Transformer Architecture

Transformers are DNN models that use what is known as self-attention to process, comprehend complex dynamics in, and make predictions on sequential data. As was briefly discussed in Section 2.2.2, Transformers stand out compared to traditional

sequence models such as RNNs or LSTMs in their increased ability to capture long-range dependencies in a dataset as well as their enabling of parallel data processing [23]. Originally, the Transformer architecture was designed for tasks in the field of Natural Language Processing (NLP) when first introduced in the 2017 paper "Attention Is All You Need" [40]. From there, its use cases have moved on to many other domains, such as computer vision, multi-modality, audio and speech processing, and signal processing [23]. Applying Transformers to time-series tasks is also interesting, given their increased ability to capture long-range dependencies as well as computational efficiency compared to traditional sequence models as discussed in Section 2.2.2. Typically, there are two main tasks performed on time-series data: forecasting and classification. Forecasting means predicting real-valued numbers from a given time-series dataset (commonly known as regression), and classification means categorizing the given time-series data into one or more target classes [8].

Although the specific use case for this thesis has to do with time-series regression, given that the prime incentive of the thesis is to use Transformers to generate proxy CDS spreads as outlined in Section 1.4, the following sections will provide an explanation of the original Transformer model first presented in [40] used for NLP tasks. More details on the slightly modified version employed in this thesis is however presented in Chapter 4. To be noted is that the Transformer is a quite complicated DNN which means a lot of the intricate, mathematical details involved in its different parts will not be covered here. Instead, the aim is to provide an easy-to-understand overview of the architecture and its main functionalities as they were presented in the original paper [40], which serves as the reference for the following explanation. However, the interested reader is prompted to read [40] in its entirety for a detailed description. Considering the Transformer's advance in recent years there also exists plenty of material on it in more beginner friendly formats. One example is the YouTube video in [38] which offers a clear and concise outline of the information presented in [40], accompanied by detailed illustrations.

The authors of [40] explains that the motivation behind creating the Transformer was the already mentioned limitation of previous recurrence based sequence models to capture long-range dependencies in datasets, as well as their inherent nature of processing data sequentially rather than in parallel which causes computational inefficiency and slow training times. Furthermore, they explain that other model architectures using so called attention mechanisms, such as the one first presented in [10], have been developed, but still with a dependency on recurrence and hence sequential data processing. In view of this, the Transformer instead ignores recurrence entirely and solely focuses on attention mechanisms. Before proceeding, it is important to know that the Transformer does not use the exact attention mechanism outlined in [10], but rather a specific variant of this called self-attention. The focus will therefore be on this instead of the original attention mechanism of [10].

Figure 2.3 shows a schematic of the original Transformer architecture. Below, an explanation of each part of the network is explained to demonstrate how the model uses self-attention and parallelisation in a encoder-decoder fashion to map input to

output.

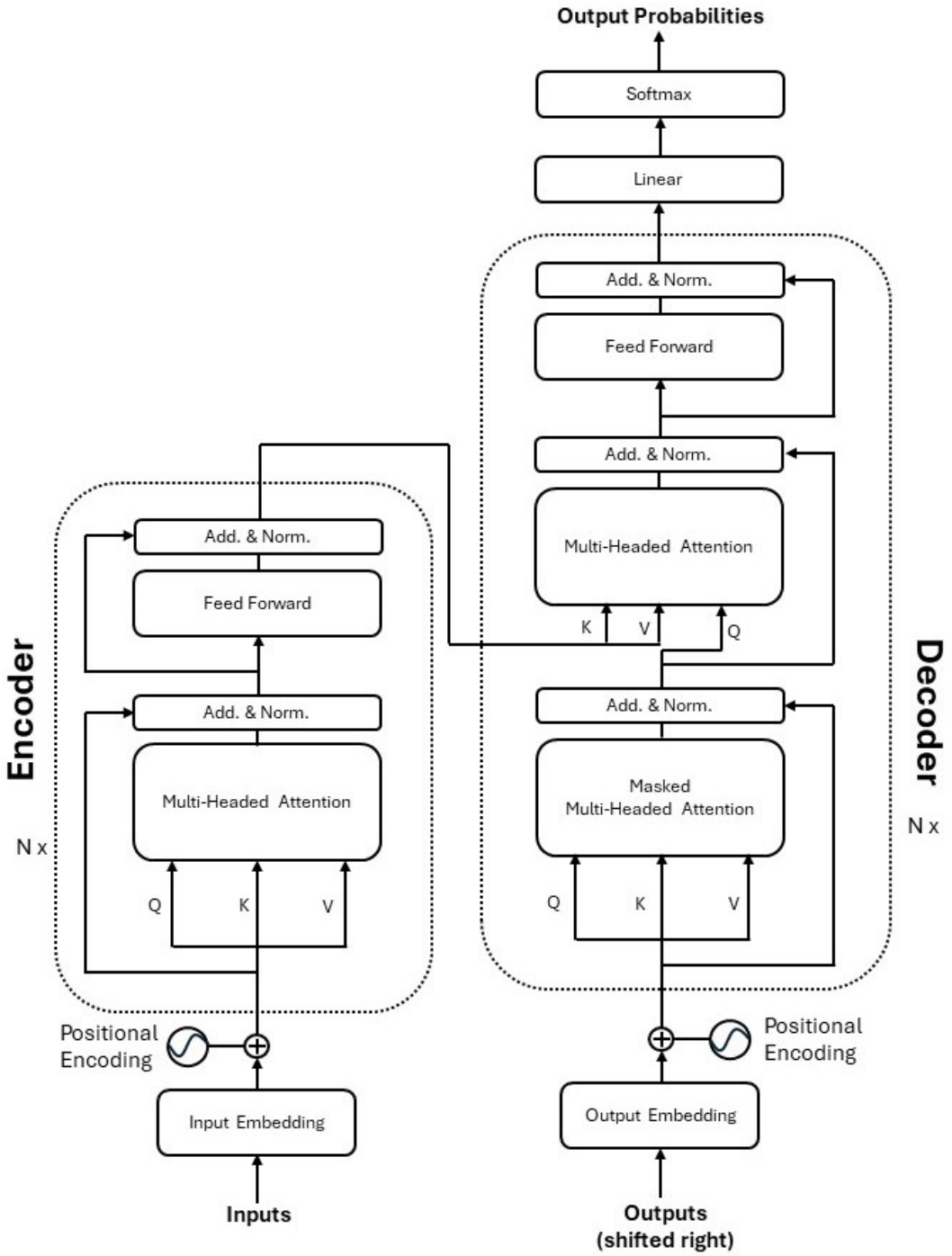


**Figure 2.3:** The original Transformer architecture. Illustration created by the author using PowerPoint, inspired by [40] and [38].

As illustrated in Figure 2.3, the Transformer uses an encoder-decoder structure. From a macro perspective this is how the original Transformer works: the encoder maps the input sequence, such as a sentence in an NLP application, to continuous

representations. The encoder output is then passed to the decoder which converts this into a target sequence, one symbol at a time. The decoder also uses all previously generated output to add additional context and memory when generating the target output sequence in a process called autoregression. Below, the encoder and decoder of the original architecture are detailed.

#### 2.2.3.1 The Encoder

First, the input is passed through an input embedding layer. Considering that neural networks, as with all computer models, learns with numbers, this layer is used to map each input token (which for example is a word in a sentence) to a continuous vector representation. After the input embedding layer, positional encoding is added to the vector representations of the input tokens. Since Transformers, as mentioned, do not utilize sequential processing as in recurrence models, all input passed to the network is processed in parallel. While this opens up for efficient computation, it however means one must add information about position into the input embeddings. In [40], the authors perform the positional encoding using sine- and cosine-functions since these possess linear properties which the model can easily learn:

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{2i/dmodel}}) \tag{2.7}$$

$$PE(pos, 2i) = sin(\frac{pos}{10000^{2i/dmodel}}) \tag{2.8}$$

What Equations 2.7 and 2.8 show is that for even dimensions of the embedding vector, the positional encoding is computed using the sine function and for odd dimensions it is computed using the cosine function. These encoded vectors are then added to the input embedding vector for the corresponding token to effectively include information about its position in the input sequence.

Next, the input is passed to the encoding layer. The purpose of this layer is to create an abstract, continuous representation of the learned features for the entire input sequence. As shown in Figure 2.3, the encoding layer is built up by two sub-modules: Multi Headed Attention (MHA) and a Feed-Forward (also denoted "Fully Connected") networks. Each of these modules also contains residual (or "skip") connections (the arrows circumventing the modules in Figure 2.3) as well as layer normalization. The MHA module applies the specific attention mechanism mentioned earlier called self-attention. This mechanism illustrated in Figure 2.4.

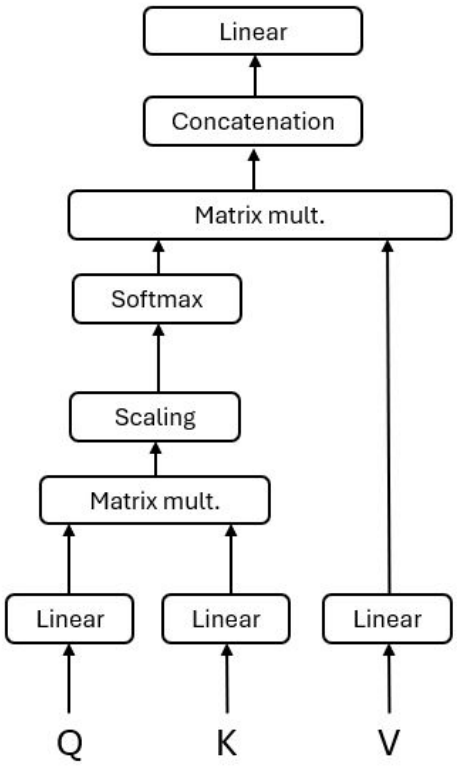


**Figure 2.4:** The self-attention mechanism. Illustration created by the author using PowerPoint, inspired by [40] and [38].

The self-attention mechanism allows the Transformer to associate every individual input token to all other input tokens in the input sequence. The embedded input is first mapped to vectors known as the Query (Q), Key (K) and Value (V) by being passed through separate, linear layers. Following this, the query and key vectors are multiplied to produce a score matrix which shows how much attention each input token should place to every other token in the input sequence. This matrix is then scaled down by the square root of the dimension of the query and key vectors in order to stabilize the gradients to avoid the exploding gradient problem discussed in Section 2.2.1, which allows for more stable training during backpropagation. The softmax function is then applied to the scaled score matrix to produce attention weights which are values in the range 0-1, mimicking probabilities. Large scores produces a high probability and vice versa, meaning the model will more easily know what tokens deserves the most attention. After this, the attention weight matrix is multiplied by the value vector to produce an output which, thanks to the different attention scores, will tell the model which tokens to focus more or less on.

Before applying the self-attention mechanism outlined above, the query, key and value vectors are split into N vectors. Each subset of query, key and value vectors then undergoes the same exact operations just described via its own "head", thereby explaining the "multi headed" part in MHA. As can be seen in Figure 2.4, the output vector of each self-attention head is concatenated into a single output vector which then passes through a final linear layer. The idea of MHA is to allow for the model

to learn new representations of the input via each application of the self-attention mechanism. Thus, one normally experiment with different numbers of heads.

As Figure 2.3 shows, the output of the MHA-module of the encoder is then passed through a Normalization layer to normalize the values. Before this however, one can see that the original input is added to the processed MHA output via a residual connection. The reason for this is to allow for gradients to flow freely through the network to avoid exploding gradients and thereby stabilize the training process. The normalized output is then passed through a Feed-Forward network for further learning before being passed through another Normalization layer (again utilizing a residual connection before normalization). Having done all this, the encoding part of the Transformer is then completed. As explained, the job of the encoder is to map the input sequence to continuous representations with attention information. This will allow for the next part, the decoder, to more easily know what parts of the input to focus on when producing an output. To be noted is that one can stack several encoding layers on top of each other (illustrated by the $Nx$ symbol in Figure 2.3) in order to get richer representations of the input sequences. In the original architecture six encoders and decoders were stacked respectively.

#### 2.2.3.2 The Decoder

The job of the decoder of the Transformer is to generate output sequences. Upon a quick glimpse of Figure 2.3, one notices that the decoder has similar sublayers to that of the encoder. It first has two MHA layers followed by a Feed-Forward network, repeatedly using residual connections and normalization. What is important about the decoder in the original Transformer, and which has been touched upon in the previous text, is its autoregressive capabilities. This means that simultaneously as it receives encoded attention-informed input from the encoder, it passes all previously produced parts of the generated output sequence back to itself to allow for a more contextualized output as a whole. The decoder stops generating output when an *<end>* token is passed. To get the general point one can look at an NLP example. If for instance an input such as:

*Where are you?*

is processed via the encoder and passed to the decoder, the decoder may generate a response like:

*<start> I am home . <end>*

token by token in an autoregressive manner, with the *<start>* token signalling the beginning of the output sequence and the *<end>* token the end of the sequence. To understand the workings of the decoder an explanation of each submodule will be outlined.

In the first part of the decoder, a similar kind of input embedding and positional encoding as that of the encoder is applied. Similar to the encoder, the self-attention mechanism is then applied to the query, key and value generated vector representations of the input in an MHA layer.

As highlighted in the beginning of this section, a strength of the Transformer is its

parallel processing abilities. During training, this means that the Transformer will generate the entire predicted output sequence at once, instead of token by token such as a recurrence model would do. However, since the order in which the output is generated matters, one must somehow prevent the model from seeing the future tokens when predicting what token should come next in the sequence. In its first MHA layer, the decoder thus shields (or "masks") parts of the input to prevent itself from conditioning on future tokens when predicting the next token of the output sequence. In the NLP example this would mean that when the MHA layer computes the attention scores for the token *"am"*, it should not have access to the token *"home"* since this appears at a future point in the sequence. Without going into the specific mathematics of the masking process, what is done is that the values of future tokens in the embedded vector representations of the decoder input are set to negative infinity, which effectively makes their attention weights equal to zero upon passing them through the softmax function (see Figure 2.4 for repetition on the self-attention steps). This limits the decoder to only seeing past outputs when determining what attention each token in the sequence should be appointed. An important note is that this refers to the computation of attention and intermediate outputs across all positions simultaneously *during training specifically*. The final sequence generation during inference (i.e. use of the trained model) is still sequential and autoregressive. Similar to the encoder, for each self-attention head, the masked attention weights are then multiplied by the corresponding value vector to produce the respective output vectors that are then concatenated into a single output vector for the entire MHA layer. This is then passed through a linear layer for further processing before being passed to the second MHA layer.

In the second MHA layer, which does not involve any masking operations, the decoder receives input from the encoder alongside the output from the decoder's first MHA layer. The encoder output serves as the key and value vectors respectively, while the output from the decoder's first MHA layer serves as query. The impact of this is that while keeping memory of the already produced output, the decoder receives additional information about what features of the encoder input to put focus on. Subsequently, the output from the second MHA layer is combined with the input of the first MHA layer using a residual connection, and then it undergoes normalization. It is then passed through a Feed-Forward network and and once again normalized.

The output from the decoder is then passed through a final linear layer which acts as a classifier in the original architecture (which, as said, was made for NLP tasks). The output dimension of the linear layer will depend on the number of classes the Transformer model is supposed to generate. So, if for example one has 1000 classes for 1000 tokens the output vector of the linear layer will have a dimension of 1000. This vector is then passed through a softmax layer which maps the values to the range 0-1 to mimmick probabilities. The index of the element with the highest softmax score then determines the predicted Transformer output for the current round of inference. This is then added to the already predicted output sequence and, again using autoregression, the process is iterated until the *<end>* token appears.

As the reader may now be aware, the Transformer is a very complex DNN architecture, composed of several complex submodules. To maintain focus on the overarching functionality and avoid overwhelming detail, certain specifics have deliberately been omitted. The key takeaway is that the Transformer leverages an encoder-decoder structure, self-attention mechanisms, autoregressive properties, and parallel processing to overcome traditional limitations in handling long-range dependencies in data, and thereby enhancing computational efficiency. While this section has concentrated on the original architecture, subsequent sections of the thesis will demonstrate the Transformer applications beyond NLP tasks.

## 2.3 Previous Implementations for Producing Proxy CDS Spreads

The following section will cover some previous work in the field of proxy CDS spread generation. To start off, the Nomura model which is commonly used by the financial industry will be discussed. Thereafter, some ML-implementations will presented and an emphasis will be put on the connection between this thesis and previous theses that have been done at SHB for this purpose.

### 2.3.1 The Nomura Model

As was described in Sections 2.1.1, 2.1.2 and 2.1.3, the spread is a principal component in the pricing process of a CDS contract. It is supposed to adjust the valuation of the CDS by reflecting the CCR associated with it at any point in time. For liquid CDS contracts (i.e. contracts that are traded frequently and in large volumes in the market), one can assume that the associated spread is well-aligned with market perception of the CCR. For illiquid CDS contracts, however, determining the spread becomes more challenging due to the lack of active market perception. Thus, the spread must be calculated in a manner that encapsulates the associated CCR as accurately as possible, despite the limited market perception. In the international regulatory framework for banks known as Basel III it is presented that when counterparties do not have associated liquid CDS spreads, financial institutions should instead use appropriate proxy spreads with regards to the rating, region and sector of the counterparty [11], [31], [20]. Following the need for accurate proxy CDS spreads that follows the mentioned regulatory demands, Nomura which is a global financial services group released a cross-sectional model in 2013 doing just that [31] (hereafter referred to as the Nomura model). Since then, the Nomura model has become common within the financial industry for the purpose of generating proxy CDS spreads.

The Nomura model initially aimed at improving the lacking results of the intersection model presented by the European Banking Authority (EBA) [17] by applying an alternative approach based on cross-sectional regression [31]. The main idea of the Nomura model is that the proxy spread for a counterparty is the product of

various factors. These include a *global factor* which is a universal adjustment applied across all related counterparties, as well as factors for the *sector, region, rating* and *seniority* that are adjustments based on the counterparty's specific characteristics. The formula for computing the proxy spread, $s_i^{proxy}$ for counterparty $i$ thus becomes:

$$s_i^{proxy} = M_{glob} M_{sector(i)} M_{region(i)} M_{rating(i)} M_{seniority(i)} \tag{2.9}$$

The calibration of the model is done using a regression approach with the logarithm of the factors. Upon calibration one aims to minimize the squared differences between model-generated proxy spreads and market spreads. Upon historical analysis it has been shown that the Nomura model avoids the erratic behavior observed in EBA's intersection model when market conditions change or in the case of undefined spreads due to cases where certain intersections of sector, region and rating does not exist [31].

### 2.3.2 Machine Learning Implementations

Some attempts have been made in the past to produce proxy CDS spreads using Machine Learning (ML). Although the approaches have differed, something common among the various implementations is their belief that ML-algorithms possess a higher potential in capturing important, non-linear patterns in the input data which other models (such as the log-linear regression approach of the Nomura model discussed in Section 2.3.1) fail to do. Below, a brief outline of some past approaches will be given.

In the series of papers published in 2019 by Brummelhuis and Luo [15], [14] the authors investigate the construction of proxy CDS spreads. Similar to the purposes of this thesis, their focus is on generating credible spreads for illiquid counterparties, but instead of focusing their analysis to a specific algorithm they expand their study to involve a comparison between various ML approaches including regression trees, neural networks, and ensemble methods. By constructing an input dataset composed of market data, financial statements, and macroeconomic indicators and subsequently training the mentioned models on it, they were able to show that ML models could outperform traditional linear regression models in predicting CDS spreads. For example, the authors benchmarked the results against the Nomura model which highlighted improvements in predictive accuracy and reliability in favor of the ML approaches.

Considering the importance that accurate proxy CDS spreads has to institutions such as SHB (see Section 2.1), previous thesis projects at the bank have also been dedicated to the task of generating them using different ML approaches. One such example is the 2023 thesis by Fageräng and Thoursie [18]. In their work, the authors showcased some scenarios where the Nomura Model fails to produce reliable proxies. Particularly, they found that this is the case during periods of volatile market conditions and for larger counterparties. As a response to this, the thesis proposed using two Long Short-Term Memory (LSTM) models trained on datasets including CDS spreads of similar counterparties. These models aimed to generate

better proxy CDS spread curves during periods of high market volatility considering their ability to capture nonlinear dynamics in the datasets which might be missed by traditional linear regression models such as the Nomura model. The general conclusions of the thesis is that the LSTM-models achieves a better performance than the Nomura model during instances where it fails, such as in the mentioned periods of high market volatility, although further improvement measures such as expanding the dataset and refining the model structure was identified [18].

Given the limited abilities of capturing long-range dependencies suffered by recurrence models like LSTMs (discussed in Section 2.2.2), Johan Luhr investigated the possibilities of generating proxy CDS spreads using the Transformer model (see Section 2.2.3) in his 2023 thesis made in collaboration with SHB [29]. To the best of the his knowledge, no prior work had been done on the application of Transformer models to the task of generating proxy CDS spreads before the work presented in his thesis [29], thus in a way making it a proof of concept. In his thesis, Luhr developed two Transformer models. The first, simpler model, only included the encoder part of the original Transformer architecture (see Section 2.2.3) and used data records from one counterparty at a time to produce one proxy spread per inference. The second, more advanced model, employed a modified version of the entire encoder-decoder architecture of the original Transformer along with data from three counterparties at a time to enhance the accuracy of the proxy spread generated. For both models, custom embeddings for incorporating features such as tenors, ratings and sectors as well as time-dependency details about the CDS spreads were used. What Luhr found was that the models effectively were able to produce proxy spreads whose curves were well-aligned with that of the Nomura model. It was also discovered that the more advanced, encoder-decoder model outperformed the simpler enocoder-based one. More details on the implementations made in [29] will be discussed in the subsequent chapters of this thesis.

What is important to mention with regards to [29], is that it lacked some important evaluation analysis needed to determine the Transformer model's supposed superiority over existing traditional methods such as the Nomura model. One especially important aspect of this, is that the target labels used during the training of the Transformer models in [29] consisted of proxy spreads rather than actual market spreads. The implication of this is that the resulting Transformer output became somewhat dependent on the same linear models it sought to replace, making it difficult to independently compare one to the other. Moreover, the thesis did not provide thorough statistical tests between model output and corresponding, market spreads, but instead limited the comparisons to be between Transformer generated proxies versus linear regression generated proxies. This meant it lacked an analysis in which each method's resulting proxies independently were compared to that of the real-world market data they are supposed to simulate, which would make it easier to draw firm conclusions on which model is the superior one and thus should be favored in the task of producing reliable proxies. These limitations therefore form the basis and justification for the work presented in this thesis and why it is an important addition to the work presented in [29]. To better be able to tell

whether or not Transformer models outperform corresponding models for the task of generating proxy CDS spreads, this thesis will thus put a core emphasis on the evaluation part of the Transformer generated output against both market and proxy spread data, as well as solely using data from actively traded, liquid CDS contracts instead of existing proxy data when training the Transformer. More details on the implementations and experiments made in this thesis along with their connection to [29] are presented in Chapters 3 and 4.

## 2.4 Statistical Context

This section provides an explanation of the primary statistical metrics used in the evaluation of this thesis's results. These include Spearman's Rank Correlation Coefficient, Mean Squared Error and Mean Absolute Error. Each metric is detailed in its own section below.

### 2.4.1 Spearman's Rank Correlation Coefficient

Spearman's rank correlation coefficient ($r_s$) measures the strength and direction of association between two ranked variables. It is nonparametric which means it does not assume that the associated variables follow a normal distribution. Compared to the commonly used Pearson product-moment correlation, which assesses linear relationships, Spearman's correlation instead assesses monotonic relationships whether these are linear or not [27]. A monotonic relationship fulfills one of the following behaviours:

- As the value of one variable increases, so does the value of the other variable.
- As the value of one variable increases, the other variable value decreases.

By using ranks instead of the raw data values, Spearman's correlation can be applied to data that does not meet the assumptions of parametric tests, such as normal distribution and linearity [27]. The concept of ranking variables is a straightforward procedure. Assuming you have two samples containing values for each variable, the ranking is applied in such a way that highest value of each sample gets the rank 1, the second highest the rank 2, et cetera. At the occurrence of two identical values in a sample, the rank of each becomes the average of the rank that they would have otherwise occupied. Table 2.1 shows a simple example of two data samples and their respective ranking.

**Table 2.1:** Example of ranking of two variables.

| **Sample 1 (value)** | **Sample 1 (rank)** | **Sample 2 (value)** | **Sample 2 (rank)** |
|---|---|---|---|
| 12 | 4.5 | 3 | 1.5 |
| 17 | 1 | 8 | 5 |
| 12 | 4.5 | 3 | 1.5 |
| 10 | 3 | 5 | 3 |
| 9 | 2 | 6 | 4 |

The formula for computing Spearman's rank correlation coefficient then follows Equation 2.10:

$$r_s = \frac{cov(R(X), R(Y))}{\sigma_{R(X)}\sigma_{R(Y)}} \tag{2.10}$$

where $R(X)$ and $R(Y)$ denotes the rank variables, $cov(R(X), R(Y))$ is the covariance of the rank variables, and $\sigma_{R(X)}$ and $\sigma_{R(X)}$ are the standard deviations of the rank variables [27]. In case there exists no paired ranks, the formula can be reduced to Equation 2.11, where $d_i$ is the difference in paired ranks and $n$ is the number of cases:

$$r_s = 1 - \frac{6\Sigma d_i^2}{n(n^2 - 1)} \tag{2.11}$$

The correlation coefficient $r_s$ takes continuous values ranging from -1 to +1, where +1 indicates a perfect positive relationship between the ranks, 0 indicates no relationship between the ranks and -1 indicates a perfect negative positive relationship between the ranks [27].

### 2.4.2 Mean Squared Error

In regression analysis, a common measure used to evaluate the accuracy of a model is Mean Squared Error (MSE). MSE calculates the average of the squares of the errors, where an error is the difference between the actual target value and the predicted value by the model. The formula for MSE is presented in Equation 2.12, where $a_i$ is the ith actual target value, $p_i$ is the corresponding predicted value, and $n$ is the number of observations:

$$MSE = \frac{\sum_{i=1}^{n}(a_i - p_i)^2}{n} \tag{2.12}$$

A low MSE means small errors on average and high model accuracy, and vice versa for large errors [34]. Due to the squaring procedure of Equation 2.12, MSE puts more emphasis on large errors which makes it extra sensitive to outliers. Moreover, it provides a smooth gradient, which is appreciated for optimization algorithms used in ML-models [7].

### 2.4.3 Mean Absolute Error

Another important metric in regression analysis is Mean Absolute Error (MAE). MAE measures the average of the absolute errors between predicted and actual values. The formula for it is given in Equation 2.13, where again $a_i$ is the ith actual target value, $p_i$ is the corresponding predicted value, and $n$ is the number of observations:

$$MAE = \frac{\sum_{i=1}^{n}|a_i - p_i|}{n} \tag{2.13}$$

Different to MSE, MAE does not square the errors. Therefore, all errors are treated equally. This makes MAE more intuitive and straightforward as it represents the

average magnitude of errors in the same units as the original data. Often one may therefore include both MAE and MSE when evaluating the accuracy of a model's predictions as each metric offers different insights. While MSE is more sensitive to outliers due to the squaring of errors, MAE provides a more balanced view by looking at the absolute differences instead [7].

# 3
# Methods

The following chapter outlines the methodology for each step conducted throughout the thesis work, thus detailing the workflow which was presented in Chapter 1. First, a summary regarding the collection of information about the topics relevant to the thesis is given. Next, an overview is provided on what data was used in the work along with some of the initial preprocessing applied to it. Finally, a description of the experiments conducted is presented along with the primary software and hardware used. In the subsequent chapter, details about the Transformer model that was used in the experiments along with how the data was preprocessed before being passed to it are given.

## 3.1 Information Collection

The work on this thesis began in January 2024, starting with researching relevant information to understand the details of the proposed project. By recalling that the aim of the thesis has been to evaluate the performance of Transformer generated proxy CDS spreads, project relevant information was needed from three primary domains: finance, machine learning and statistics.

The process for obtaining information on each context was twofold. First of all, a lot of explanatory value was provided firsthand by the staff in the Model Validation and Quantitative Analysis department at SHB, who work daily at the intersection of these topics. Secondly, literature was collected, covering mediums such as text books, research papers and online articles.

Apart from processing general information, an in-depth analysis of the SHB thesis on Transformer models conducted prior to this, i.e. the 2023 thesis by Johan Luhr [29] mentioned earlier, was made. This included dissecting all parts of that project as well as understanding the associated code and data used in its experiments. A lot of help on this was offered by Johan Luhr himself who took the time to explain his work and subsequent results. This preparatory work helped pave the way for this thesis.

## 3.2 Data Collection and Initial Preprocessing

Upon gaining an understanding for the thesis context as well as previous work, the workflow then progressed onto data collection and initial preprocessing. As

have already been mentioned, all data for the thesis was provided directly by SHB. Since the data is a commodity it has however not been possible to share it publicly. Therefore, no extracts from the datasets will be shown, but rather a description of the data characteristics needed to understand what utility they have offered will be provided. Two primary categories of data have been employed in this thesis: counterparty specific (so called single name) market CDS spread data and proxy CDS spread data. Each will be covered in the following sections.

### 3.2.1 Single Name CDS Spread Data

One category of data used in this thesis was single name CDS spread data. This data was used for training the Transformer model and during evaluation of its output. While the specifics for those procedures are detailed in Chapter 4, this section explains the initial preprocessing that was applied to it.

The single name data contained counterparty specific, end of day liquid market CDS spreads along with additional information such as counterparty, rating, region, sector and tenor for the associated CDS contract. The data is delivered in daily CSV-files and a period of four years between the beginning of 2020 until the end of 2023 was collected. Before the Transformer modelling the data had to undergo a substantial amount of preprocessing.

First of all, a lot of data had to be removed in order to concentrate the analysis to a certain subset. This had several reasons. An apparent first one was that many of the supplied columns in the original raw data files were not of interest for the purposes of this work. By recalling from Section 2.3.1 that the Basel III accord states that financial institutions should use proxy CDS spreads appropriate with regards to the rating, region and sector of the illiquid counterparty for which they are to be used, these were the columns of data that were collected apart from the single name spreads themselves. Another reason for initial data removal was due to data amount and liquidity in different subsets of the raw data files. The idea behind this was first presented in [29] and it is that by having a comprehensive dataset of liquid contracts, the data will be less prone to containing errors that might affect the models negatively during training. Data containing spreads for CDS contracts with the region and sector categories set to "Western Europe" and "Financials" respectively best met these constraints and were therefore kept. Moreover, this intersection of region and sector is the one most relevant to SHB.

Having cleaned the files and after performing some additional formatting to ensure each row contained the wanted information for each counterparty, the daily CSV-files were merged into a single, large CSV-file which would then go on to be used in the experiments. This file chronologically contained information on business date, counterparty, spreads, et cetera needed when constructing input data and output labels for the Transformer model (as mentioned, this procedure is detailed in Chapter 4). Table 3.1 shows what columns were present in the single name dataset after the initial preprocessing.

**Table 3.1:** Row structure for single name data.

| Column Name | Containing |
|---|---|
| BusinessDate | Date for spreads |
| issuerName | Associated counterparty |
| Sector | Counterparty sector |
| Region | Counterparty region |
| Rating | Counterparty rating |
| 6M | Spread for six month tenor |
| 1Y | Spread for one year tenor |
| ... | ... |
| 30Y | Spread for thirty year tenor |

The final columns of Table 3.1 included the daily spreads for each tenor for the specific couterparties. The following tenors were considered, where "M" stands for months and "Y" stands for year:

- 6M, 1Y, 2Y, 3Y, 4Y, 5Y, 7Y, 10Y, 15Y, 20Y and 30Y

It is also noteworthy to mention that both the Sector and Region columns only contained a single category each ("Financials" and "Western Europe" as mentioned above), while the Rating column could take various categories (for example AA, A, BBB, et cetera). As mentioned earlier, Chapter 4 will provide a detailed explanation of how the single name data was preprocessed for creating model input as well as output target labels.

### 3.2.2 Proxy CDS Spread Data

The second category of data used in this thesis was proxy CDS spread data. It is important to note that the proxy spread data has in no capacity been part of the training of the Transformer model (as Chapter 4 will show). Since part of the thesis's aim has been to evaluate whether Transformer generated proxy CDS spreads are able to outperform traditional methods for generating them, proxy data was instead collected in order to have a benchmark to compare with. This is an important difference compared to the the prior thesis on Transformers at SHB [29], in which the proxy spread data was used as the actual labels during training (see Section 2.3.2 for details about this and how it differs to approach taken here).

The model used to generate the proxy CDS spread data collected in this thesis is based on the Nomura model described in Section 2.3.1. In short, the model applies cross-sectional regression to produce proxy CDS spreads using various weight factors. In the following, the proxy CDS spread data will therefore be referenced to as the Nomura model data.

Similar to the single name data, the Nomura model data was provided in daily CSV-files. Data was collected from October 2020 until the end of 2023. The format

of the Nomura model data was quite similar to that of the single name data in that it could be categorized based on rating, region, sector and tenor. However, by definition, instead of having specific counterparties as issuers, this dataset contained indices specific to certain intersections of rating, region and sector. The implication of this was that for each business date, the Nomura model dataset contained less data than the corresponding single name dataset since each index corresponds to all counterparties of the same rating, region and sector (hence "proxy"). Given the similar format of the raw data files, the cleaning, formatting and merging of the daily CSV-files into a large file encapsulating the entire dataset could be done in a similar fashion to that of the single name data. Table 3.2 shows how each row of the Nomura model dataset was structured after the initial preprocessing.

**Table 3.2:** Row structure for the Nomura model data.

| **Column Name** | **Containing** |
|---|---|
| BusinessDate | Date for proxy spreads |
| issuerName | Associated proxy index |
| Sector | Associated sector |
| Region | Associated region |
| Rating | Associated rating |
| 6M | Spread for six month tenor |
| 1Y | Spread for one year tenor |
| ... | ... |
| 30Y | Spread for thirty year tenor |

## 3.3 Experiments

Following the data collection and initial preprocessing, the experiments of the thesis were planned and conducted. Experimentwise, this thesis has involved a comprehensive scheme of training and testing several instances of a Transformer model with identical architecture and hyperparameter settings, but different output target labels for the purpose of proxy CDS spread generation. Subsequently, evaluations of the results for the model outputs have been carried out using statistical metrics and visual representations. The outlines for these two phases are provided in the following sections.

### 3.3.1 Training of Transformer Model

In the first experimental phase of this thesis, an approach to generate proxy CDS spreads using a Transformer model was implemented. The primary objective was to determine if the accuracy of the proxy spreads generated by the Transformer could outperform that of the Nomura model.

#### 3.3.1.1 Training Setup

The training involved distinct subsets of the single name dataset, each classified by credit ratings. This segmentation was designed to allow focused analysis on how well the model predicted spreads for counterparties with different credit rating. Five credit rating categories were used:

- AA
- A
- BBB
- BB
- B

For each rating category, several model instances were trained to predict proxy CDS spreads tailored to the specific financial profile associated with that rating. This was done using target labels that corresponded to the rating of interest when training each model (the construction of target labels is detailed in Section 4.2.1.1). The architecture and hyperparameters were fixed across all model instances and rating categories in order to maintain uniformity in the experiment. This allowed for a direct comparison of model performance based solely on the variation of the rating category it was generating spreads for.

An important aspect of the methodology was the use of an Out-Of-Sample (OOS) approach for each model instance during training and testing to ensure the robustness of the model against unseen data. What this meant in practice was that for each model instance in each rating category, a counterparty (denoted company during modelling) was held out of the training process. The model was then trained on a dataset which excluded the data from the selected OOS-test company to prevent the model from learning the specific patterns of it, basically making sure that its data was unseen by the model during training. The withheld data of OOS-test company was then used as the test set for that model instance. The idea with this approach is that it tested each model's ability to accurately predict proxy CDS spreads for a counterparty it had not been exposed to during training. This strategy was meant to simulate realistic conditions where the Transformer model needs to generate accurate proxy CDS spreads for illiquid counterparties that lack market spreads.

Ten individual model instances were trained in each rating category except for the B category, in which only six models were trained. This was due to the fact that in comparison to the other rating categories, few counterparties with a B rating existed in the single name dataset and out of the ones who did, only a select few had enough data records to be worth considering. To be mentioned once again, the specifics of the Transformer model's architecture, the data preprocessing, and the hyperparameter settings are detailed in Chapter 4.

To be noted, an additional test with an increase in training time was also conducted.

This was only performed for the AA category and only three such model instances were trained and evaluated. The results and details of this test is presented along with all other results in Chapter 5.

#### 3.3.1.2 Statistical Significance Through Multiple Runs

The idea behind training multiple models per rating category (i.e. several model instances) was to ensure the statistical significance and robustness of the results obtained. Training many independent models (of same architecture, but using different datasets) helps in lowering the risk of anomalies due to specific dataset characteristics, which is important in order to avoid performance biases introduced by potential outliers or non-representative issuer behaviors that may appear in a specific dataset. This is particularly important in case the model is to be used for generating proxy CDS spreads for actual illiquid counterparties in the future. Additionally, employing multiple models allows for a more sound statistical analysis. With more runs to evaluate in each category, this approach ensures that the model's capabilities is generalizable and not a result of specific data anomalies related to the training and test sets selected for a specific model instance. Lastly, a varied set of test cases provides a robust framework when comparing the performance against established benchmarks (like the Nomura model in this case). The whole point is that we wish to ensure that potential conclusions are supported by a sound amount of empirical evidence rather than theoretical assumptions.

### 3.3.2 Performance Evaluation

When all the model instances of all of the rating categories had been trained, each was tested on its respective test set which contained data for a counterparty of the same rating which the model had been trained to generate spreads for and which was held out-of-sample during training (the so called OOS-test company for that model instance). The following statistical metrics, which may be recalled from Section 2.4, were computed for each model's predictions, as well as for the corresponding proxy spreads collected for the Nomura model benchmark:

- Spearman's Rank Correlation Coefficient
- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)

In order to look at how closely the order of the Transformer model's proxy spreads and the Nomura model's proxy spreads respectively matched the rank order of the actual spreads, the Spearman's rank correlation coefficient was used. This coefficient is useful because it does not assume that the relationship between the variables is linear nor that the samples are normally distributed (see Section 2.4.1). The coefficient was computed for each model in each rating category individually and then the average for the entire sample (i.e. all models of the same rating category) was computed.

MSE was used to measure the average of the squares of the errors between model

generated proxy spreads and actual market spreads. As explained in Section 2.4.2, MSE is useful for assessing the quality of a model because it penalizes larger errors more than smaller ones, which enhances the focus on model accuracy with respect to outliers in particular.

MAE was also used, but instead of looking at the squares of errors it measures the average of the absolute magnitude of the errors. Since it evaluates on absolute values and not squares, all individual differences have equal weight. MAE was therefore deemed useful in order to provide a clear representation of error magnitude in units that are easy to understand and relate to, while MSE was used to signal the presence of outliers instead.

For each respective rating category, the following plots were also generated for the best performing Transformer model instance of the category to get a visual perception of model performance:

- Error plots showing the deviations from the actual market spreads for the proxy spreads generated by the best performing Transformer model instance, as well as for the corresponding spreads of the Nomura model.
- A plot showing the actual market spreads, Transformer generated proxy spreads, and Nomura model proxy spreads for all tenors on a single day, i.e. full spread curves.
- A plot showing the evolution over time of the actual market spreads, best performing Transformer model's proxy spreads, and Nomura model's proxy spreads for various tenors.

The results of the evaluation are presented in Chapter 5. Before that, Chapter 4 will provide the details about the Transformer model employed throughout all model instances and categories. First however, the software and hardware tools used in the thesis are detailed below.

## 3.4 Software and Hardware

All the programming performed in this thesis has been done utilizing the Python programming language [5]. For data preprocessing the Pandas [4] and NumPy [3] libraries have primarily been used. The construction and training of the Transformer model was done using the TensorFlow [37] library with the Keras API [36]. Visualizations were made using the Matplotlib library [2] and the statistical evaluations mostly relied on the SciPy library [6]. In terms of hardware, the training of the models were done on an Nvidia Quadro A6000 Graphics Processing Unit (GPU).

# 4
# Transformer Modelling

In this chapter, the Transformer model used for generating proxy CDS spreads for the experiment outlined in Section 3.3.1 is presented. First, an outline for the model implementation is explained, followed by the preprocessing scheme for constructing input data to the model as well as output target labels for the training and test sets respectively. Thereafter, the details of the model itself is presented and finally the hyperparameters used in the experiments are displayed and discussed.

## 4.1 Outline of Modelling

The Transformer modelling in this thesis is almost identical to the modelling of the encoder-decoder model introduced in the 2023 thesis on Transformers at SHB [29], which was briefly discussed in Section 2.3.2. As was mentioned there, a core difference in the implementation of that model to the one presented here is the data used as labels during training. This and additional differences will be detailed in the following subsections.

The model to its core is widely influenced by the original Transformer architecture first presented in [40] and which was detailed in Section 2.2.3, with some important differences applied. Most importantly, compared to the original Transformer architecture which was developed for NLP-tasks with the purpose of generating text sequences as output, the aim of this model is to generate proxy CDS spreads. By feeding input sequences consisting of previous single name CDS spreads and additional information regarding rating, region, sector, tenor and issuer (or "company" as it is named here), the model predicts the next spread in the sequence.

An important thing to keep in mind is that the same exact procedure in terms of preparing input and output data, architectural design and hyperparameter settings have been used for all the separate model instances trained over the five rating categories mentioned in Section 3.3.1, namely: AA, A, BBB, BB and B. This effectively means that the only difference for each instance of training and testing of the model detailed below has been the rating category it has been trained to predict spreads for, and the various out-of-sample (OOS) companies that has been used to test each respective model instance.

## 4.2 Preprocessing

As was explained in Section 3.2.1 of Chapter 3, the single name dataset, which contained real-world market data consisting of spreads for CDS contracts belonging to liquid counterparties, has been used to train the model. This is because the proxy spreads generated by the model were aimed to simulate real-world market spreads as closely as possible. The single name dataset consisted of daily spreads for 11 tenors (i.e. times until maturity for the associated CDS), as well as information on the business date, counterparty, sector, rating and region for each record (see Table 3.1).

The data first needed to be formatted correctly for neural network architectures built in TensorFlow. Additionally, the input data and output labels had to be structured so the model could learn proxies for specific intersections of rating, region, and sector. These preprocessing steps were required before every instance of training the model. Despite the seemingly extensive procedure, the actual implementation was quite fast due to the use of the Pandas and NumPy libraries in Python.

The first step during preprocessing was to construct a list of all the unique counterparties (denoted companies here) in the entire dataset, regardless of rating (called the "Full Company List"). Secondly, another list was constructed which contained all the companies that had the rating which the model was currently trained to predict proxy spreads for (called the "Test Company List"). These two lists were then used collectively to prepare input sequences and corresponding labels for each model instance of the current rating category. Algorithm 1 shows the pseudocode for this procedure. Since the preprocessing of the training and test data was handled in a slightly different way, the procedure for each will be described in their own sections below.

**Algorithm 1** Preprocessing and initialization of Transformer training

```
Algorithm: Preprocessing and Initialization of Transformer Training
Input: Single Name Dataset, Test Company List, Full Company List
Output: Input Sequences and Corresponding Output Target Labels to Transformer Model
for each OOS-company in Test Company List do
    Split dataset into training and test sets by current OOS-company
    Remove current OOS-company from the Full Company List
    Add average columns (target labels) to the training set
    Split training set into an input dataset and output target label dataset
    Filter output dataset and OOS-test set by current rating
    Add target label columns (next day) to the test set
    Save test set
    Prepare lists to store inputs and target labels for training and validation sets
    for each company in the Full Company List do
        Merge input data with corresponding output target labels
        Split the merged sample into training and validation samples
        Divide the training and validation samples into sequences of size five
        Store input and output windows for each sequence as NumPy arrays
        Append input and output arrays for each sequence to corresponding lists
    end for
    Concatenate arrays across all companies for training and validation sets
    Embedd input before passing it to the Transformer model
end for
```

### 4.2.1 Training Data

The construction of the training data set (as well as test data set which the next section will detail) followed Algorithm 1 above. As Algorithm 1 shows, the preprocessing began with removing the data for the current model instance's out-of-sample (OOS) company from the single name dataset as well as removing that company from the full company list. This was done to ensure that the model did not learn any information from it since this company's data was later used as the test set for that model instance.

#### 4.2.1.1 Constructing Target Labels

In the next step, labels were generated for each row of the single name dataset. The target label spreads were constructed by using averages of the input spreads on the basis of tenor and business date. This means, that for every row in the single name dataset (which before the operation had the format described in Table 3.1 of Section 3.2.1), 11 new columns were added (one for each existing tenor) containing the average spread of that specific tenor for all the rows in the dataset with the same business date and rating as the current row. So, for example, an arbitrary row in the dataset would show all the spreads for different tenors for a specific company with a specific rating on a specific business date. The tenor column "6M" for instance would contain the six month spread for that company on that business date. After the label adding operation, that same row would have a new corresponding label column named "6M_Label" which contained the average spread value for the "6M" column of all other rows in the data set that shared the specific business date and rating of the row that was currently being processed. Such a corresponding label column was added for all tenors and processed for all rows in the entire dataset. As was explained in Section 2.3.2, this construction of target labels was done to train the model on generating spreads as similar as possible to real market spreads. As was explained there, this approach differed completely to that of [29], in which proxy spreads were used as target labels for the model.

#### 4.2.1.2 Filtering on Rating Category

After adding target labels to the training set, it was split into two distinct datasets based on the newly added label columns. The first contained the input data which is all the data except for the label columns, while the second contained the label columns and corresponding business date and rating columns for later identification. Next, a filtering maneuver was performed over the output labels. Remember that each model instance was to be trained to generate spreads for a certain rating category. This means that for the training output labels the incentive was only to retain the records corresponding to the rating of interest for that specific model run. Therefore, the output (label) dataset was filtered to only keep records that had the rating for which the current model instance was trained to generate spreads for.

After filtering the output labels to ensure that the model generates spreads for a certain rating, the input data was sorted into input sequences which would be fed to

the Transformer model. In line 13 of Algorithm 1, one sees that the preprocessing occurs over every company present in the Full Company List (i.e. all companies present in the entire single name dataset, apart from the OOS-company extracted for testing). Two important implications from this should be kept in mind here. First of all, this meant that the model would be trained on one company at a time, processing all its data before pursuing to the next (different to [29], in which three companies were processed at once). Secondly, by looping over the full company list, this means the model would receive records containing different ratings than the one it was currently trained to generate spreads for (recall the construction of the output labels). This is important because even though the model was trained to generate spreads for a specific rating, we still want it to be able to learn patterns from other ratings since that may benefit its generative capabilities. An additional column called "Group ID" was therefore also added to the data to explicitly provide the model with information on this. The "Group ID" column contained an integer representing the specific intersection of rating, region and sector for that record. So for example, all records with one specific combination of rating, region and sector would have a specific integer in the Group ID column, while all records with another combination of rating, region and sector would have another integer in the Group ID column. Effectively, this forms a way to map all input combinations to a specific type of output proxy spread curve (based on rating, region and sector) and this procedure was first presented in [29].

#### 4.2.1.3 Building Sequences

Moving on in the inner loop of Algorithm 1, the first step was that the output target labels dataset were merged to the now company specific input dataset on the basis of matching business date. After this, the merged dataset was split into a training set and validation set. As was explained in Section 2.2.1, these two sets are used together during training of an ML-model, where the training set is the one the model learns from whereas the validation set is a much smaller sample unseen to the model during training used to evaluate how well the model can predict on unseen data after each update of the network weights. Basically, it is a way to follow the progression of the training to determine if the model is good enough or if it should train for longer. An important detail here is that because the data is of time-series nature (which means it follows a sequential, chronological order in terms of business date) the split was performed without shuffling to ensure the model was unable to learn patterns using future data points. The ratio between these samples was set to 0.9, meaning that the first 90% of the data was used for training and the remaining 10% for validation.

Next, both the training and validation sets were divided into smaller sequences. The sequence size was set to five, meaning that a sequence contained consecutive spreads for five business dates whereas the Transformer model then generates the spread of the sixth business date. The sequencing was done on the basis of tenor, which means first sequences for all the spreads for the six month tenor along with the additional information regarding company, rating, sector and group ID were generated, then the spreads and additional information for the one year tenor were sequenced, then the two year tenor, et cetera until all the input for all the company

data were sequenced. For each sequence, every column was then stored in its own NumPy array, which in turn was passed to a list of arrays. An important thing to note here is that similar to what was done with the group ID column, all the categorical data (i.e. business date, rating, region and sector) were mapped to a unique integer in this step to prepare for the embedding applied to them before passing the input to the Transformer model. More on this is covered further down in the section.

Parallel to generating input sequences, the corresponding labels for each input sequence were also sequenced. This procedure was straightforward. Since the model is supposed to predict the spread for the immediate business date followed by the last business date of the input sequence, the label for that sequence was set to be the spread value of the corresponding label column for the current tenor one day into the future. So, if for example the "3Y" tenor was being processed and the final business date of the current input sequence was "2020-10-01", then the output label for that input sequence would be the spread value of the "3Y_Label" column for the business date "2020-10-02", i.e. one day into the future (note however that in practice the dates were expressed as integers in a sequence at this point). After all the companies in the single name dataset were processed in this way, the lists of arrays containing information on the input sequences were concatenated for the training and validation sets respectively. Table 4.1 shows how an input sequence would look like after being processed in the way mentioned above.

**Table 4.1:** Structure for input sequence before embeddings were added.

| **Spread** | **Tenor** | **Comp.** | **Seq.** | **Sect.** | **Rat.** |
|---|---|---|---|---|---|
| Float | Int. | Int. | Int. | Int. | Int. |
| Float | Int. | Int. | Int. | Int. | Int. |
| Float | Int. | Int. | Int. | Int. | Int. |
| Float | Int. | Int. | Int. | Int. | Int. |
| Float | Int. | Int. | Int. | Int. | Int. |

As can be observed in Table 4.1, a column with information on the region was not included since this did not differ between the various inputs and thus did not add any additional information (recall from Section 3.2.1 that only the region "Western Europe" was used in this thesis). However, since the same can be said about the sector column (the sector was limited to financial companies), this might as well have been removed. Moreover, the group ID column was not included at this stage either. This is because this information is not passed to the model before the decoder-phase, in which it replaces the company column. However, more on why this was done is detailed in Section 4.3 on the model implementation and can be disregarded at this point.

Another noteworthy thing about Table 4.1 is that apart from the spread values themselves, all additional input was as described earlier mapped to integer values at this stage. Now, the reason for doing this is that it simplifies the embedding applied before passing the input to the Transformer model.

#### 4.2.1.4 Embeddings

As was explained in Section 2.2.1 of the Theoretical Framework, neural networks apply mathematical operations such as gradient descent, normalization and apply non-linear activation functions, all of which benefit from real-valued precision. Therefore, even though it is possible to feed integers directly into a neural network, embeddings are used to provide a higher-dimensional and continuous representation of the input data. This allows the network to learn more intricate patterns and relationships.

In this thesis's implementation, similar to what was done in [29], this was achieved using the Keras Custom Embedding Layer in Python [1]. The embedding procedure was performed as the last step of the preprocessing described in Algorithm 1. The Keras Embedding Layer converts the integer-mapped data into dense float vectors, essentially mapping the discrete numerical values into a high-dimensional space. This provides a representation that the neural network can process more effectively. Therefore, all the data except the spreads themselves (which were already of type float, see Table 4.1) were passed through a separate embedding layer before being passed to the Transformer model. The dimensions of the embeddings in this implementation were directly influenced by [29]. This meant that the tenor, company, sector and group ID integer representations were mapped to a three-dimensional vector space, while the sequence embedding mapped the one-dimensional integers into a two-dimensional vector space. This differs to the original Transformer of [40], in which values produced by cosine and sinus functions were added to encoded tokens to account for sequential encoding (see Section 2.2.3). A benefit of using the Keras Custom Embedding Layer is that it performs the embedding as long as integers are passed to it. Therefore, all that had to be done before employing it was to map the categorical data to an integer representation (as was described earlier in this section). Table 4.2 shows the structure of a row of the sequence input matrix passed to the Transformer model after embedding of the inputs were performed. $T_e$ represents the tenor embedding, $C_e$ the company embedding, $Sq_e$ the sequence embedding, $Sec_e$ the sector embedding and $R_e$ the rating embedding.

**Table 4.2:** Row of an input sequence after embeddings were added.

| Spread | $T_e$ | $T_e$ | $T_e$ | $C_e$ | $C_e$ | $C_e$ | $Sq_e$ | $Sq_e$ | $Sec_e$ | $Sec_e$ | $Sec_e$ | $R_e$ | $R_e$ | $R_e$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

In summary, the preprocessing to the Transformer model described above produced the input in the form of NumPy arrays (sequence matrices) with dimensions *5 x 15*. Each array contained five consecutive spreads with embedded, company specific, information. This input was then passed through the model which finally generated a single proxy spread value one business date into the future for the tenor currently being processed. Details about this procedure are given in Section 4.3, where the Transformer model used in this thesis is described.

### 4.2.2 Test Data

As was explained in Section 4.2.1 above, some steps of the preprocessing of the test set for the model was performed in a slightly different way than for the training set

and will therefore be briefly detailed below.

First of all, in lines 5-6 of Algorithm 1, one can see that all the data records for the current OOS-company used as test set were removed from the single name dataset and the Full Company List. This OOS-company specific data was the data used for testing the model and therefore it was crucial that it remained unseen to the model during training. Given that each model instance has been trained to generate proxy spreads for a certain rating category, it was necessary to ensure that the test set only contained records where the rating was equal to that which the model was to generate spreads for. Now, it may appear as though this was already ensured given that the Test Company List, from which the test set (i.e. current OOS-test company) was obtained only contained records with the targeted rating (see line 9 of Algorithm 1). However, it was observed upon inspection of the single name dataset that companies could change rating over time and given the fact that the test set was obtained directly from the single name dataset it had to be ensured that potential instances where the OOS-test company had another rating in the past (for example if it used to have an A rating but a BBB rating now) were removed from the test dataset. Therefore, a filtering of the records in it to remove potential instances with an undesired rating was performed. As mentioned above, every record of the training dataset that possibly could contain information for the same OOS-test company under a different rating were also removed by eliminating the company from the Full Company List.

After filtering for rating, output labels were added to the test set. This was done in a different way than for the training set. Since the test set was used to evaluate the Transformer model's ability to generate the next days spread for that specific company, no need for averages were necessary. Instead, the label for each row in the test set was simply set to be the corresponding spread value one business date ahead.

Upon extracting the test set and after labels had been added to it, it was saved to a separate file (line 11 of Algorithm 1). When the training of the model was completed, the test set was then loaded and the additional preprocessing needed to pass it to the model was performed identically to the additional steps of the preprocessing described in Section 4.2.1 above.

## 4.3 Transformer Model

The Transformer model used in this thesis is as mentioned identical in architecture to the encoder-decoder model presented in the 2023 thesis by Johan Luhr [29]. Basically it is the original Transformer from [40] which was described in Section 2.2.3, but revised in order to predict proxy CDS spreads instead of text sequences. Considering this, the reader is referred to Section 2.2.3 and [40] for the details regarding the mechanisms of each part of the Transformer, whereas this section will instead focus on major differences applied to certain parts of it in order to achieve the desired output.

### 4.3.1 Architectural Differences to the Original Transformer

Regarding the Transformer architecture used in this thesis, which to be mentioned again is based on the one presented in [29], a first significant difference from the original Transformer (shown in Figure 2.3 in Section 2.2.3) is the removal of masking in the first multi-head attention (MHA) layer of the decoder. Originally, this masking was important to ensure that the decoder did not see future time-steps while generating text sequences, thereby operating in a unidirectional manner. This feature is important for step-by-step sequence prediction tasks, for example in NLP-problems (such as a language translation), where each output word must solely rely on previously generated words (the autoregression explained in Section 2.2.3). In contrast, the architecture used here omits this masking within the first MHA layer of the decoder, enabling the attention mechanism to access all input sequence positions in a bidirectional manner. The idea behind using bidirectional self-attention without masking MHA was applied in [29] as well and it is that it allows the model the ability to utilize all available information of the input sequence simultaneously. Remember that the aim of the model used here is to generate the most accurate CDS spread prediction one day ahead for each new input sequence passed to it (which again consists of the spreads for the five previous days relative to the one it is predicting for). Therefore the autoregressive feature is not needed because the model only predicts for the next day based on the current input sequence and without passing its own output back to itself. Masking the input to the decoder thus becomes unnecessary and instead the model is allowed to see and use information from the entire input sequence when predicting the spread for the next step following that sequence.

Furthermore, the output generation of the original Transformer was changed to suit regression needs. Originally, the Transformer model outputs a sequence of tokens, with a softmax function at the end to determine the next token in a sequence. This model is however designed to output a single continuous value, specifically a proxy CDS spread. Therefore, the final layers in this model consist of a global pooling layer, a series of dense layers, and ultimately a single dense layer that outputs a continuous value (the predicted spread). The Transformer architecture used is shown in Figure 4.1 in Section 4.3.3.

### 4.3.2 Training Differences to the Original Transformer

In addition to architectural modifications, adaptations relating to the training of the Transformer model were also implemented (again replicating what was done in [29]). Instead of using Stochastic Gradient Descent (SGD) as the optimizer, the Adam optimizer was used. Detailed in Section 2.2.1, Adam is preferred for its adaptive learning rate capabilities. Furthermore, Mean Squared Error (MSE) was used as loss function instead of Cross-entropy loss. MSE is the a common choice for regression tasks which aims at minimizing the error between continuous predicted values and actual values, such as in the generation of proxy spreads versus market-based target spreads which is the purpose of this implementation. Cross-entropy loss, on the other hand, is more common in classification tasks (such as the NLP task of the original Transformer), where outputs are discrete categories and the

model instead predicts the likelihood of each category.

### 4.3.3 Schematic of Revised Transformer Model

Figure 4.1 shows a schematic of the revised version of the Transformer architecture used in this thesis. A final remark is that instead of stacking 6 encoders and decoders on top of each other respectively, as in the original implementation, 4 of each were used here (see also Section 4.3.5 for an overview of the used hyperparameters).

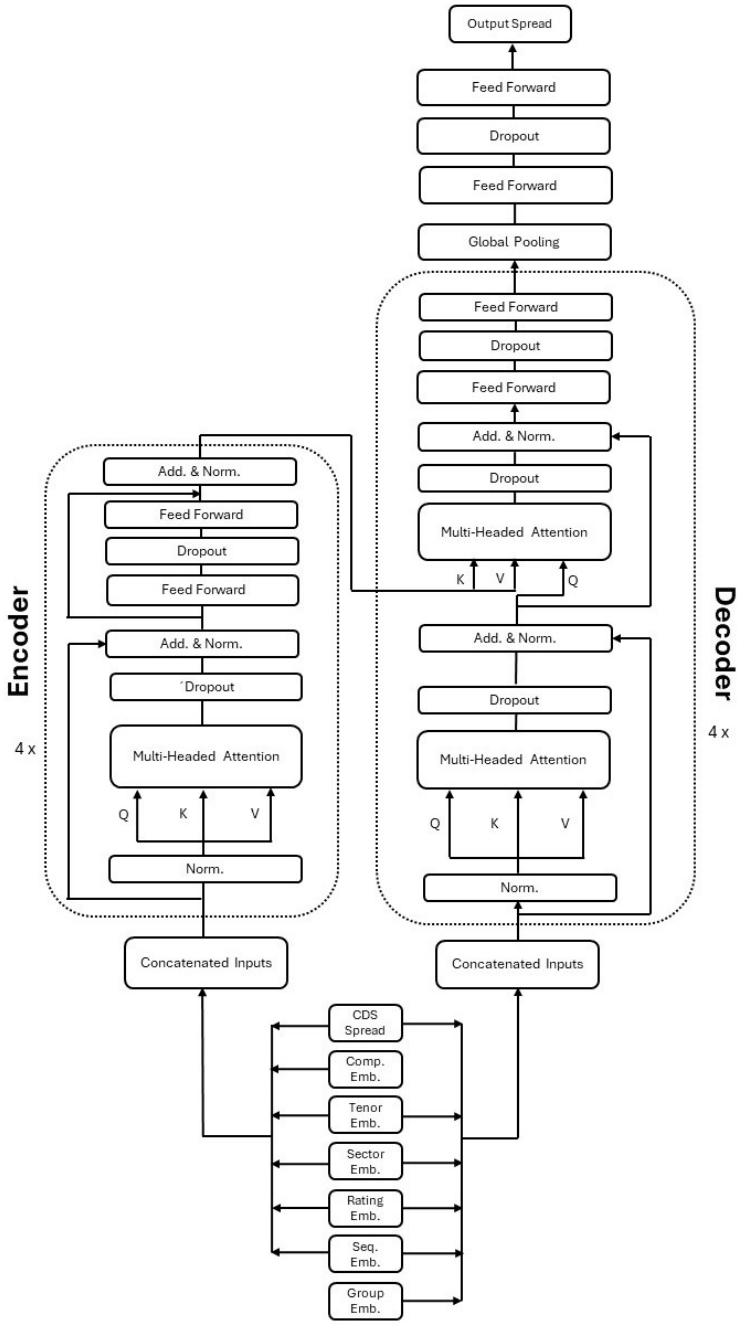


**Figure 4.1:** The Transformer architecture employed in this thesis. Illustration created by the author using PowerPoint, inspired by [29].

### 4.3.4 Passing of Inputs

A final remark, which was discussed briefly in Section 4.2.1, has to do with how the inputs were passed to the Transformer in this implementation. In the bottom of Figure 4.1 one can observe that the encoder receives the embedding for the company currently being processed (labeled "Comp. Emb.") and not the group ID embedding, while the opposite is true for the decoder. This concept was first presented in [29] and has been kept in this implementation. The idea with this is to allow for a mapping between all types of input combinations of rating, region and sector to any type of output target.

Recall that each input record has a unique value associated with what "proxy group" it is categorized by given its rating, region and sector combination. For example, a financial company with rating AA located in Western Europe belongs to one group, while for example an Asian manufacturing company with rating BB belongs to another. Now, as the generation of proxy CDS spreads are done on the basis of rating, region and sector in alignment with the Basel III recommendations (see Section 2.3.1), the group ID embedding essentially provides information on what kind of proxy group the specific company record currently being processed belongs to. Since the decoder is supposed to predict proxy spreads for a targeted group (for example AA rated financial companies in Western Europe), the idea is to let it know what kind of group is currently being processed in order to map potential relevant information from it that can help in the generation of spreads for the targeted one. The aim is therefore to be able to provide the Transformer with CDS spread data of any proxy group, regardless of the one spreads are currently being generated for, with an anticipation that it learns relevant information from it.

### 4.3.5 Hyperparameters

The choice of hyperparameters for the Transformer model implemented in this thesis were selected on the basis of two counts. First of all, since an overall aim was to replicate the model presented in [29], this greatly influenced the choice of the hyperparameters. As explained in [29], there is a difficulty in selecting optimal settings for hyperparameters and often this requires detailed and computationally expensive analyses (such as a grid search) which was out of the scope for this thesis (recall the limitations of Section 1.3). Therefore, the best performing settings of the hyperparameters as presented in [29] were kept to the greatest extent possible, which leads to the second count.

In Section 3.3.1 regarding the outline of the thesis experiment, it was explained that approximately ten individual model instances were trained per each of the five rating categories. Given the vast amount of model instances and time constraints of the thesis, a decrease in training time for each model instance was therefore made. This was done by lowering the number of epochs and early stopping criterion compared to [29]. The final hyperparameter settings used in all the instances of training the Transformer model, regardless of rating category, can be seen in Table 4.3. Applying these settings to the architecture shown i Figure 4.1 produced a Transformer model

consisting of 780 349 parameters in total.

**Table 4.3:** Summary of the hyperparameters used for the Transformer model.

| **Hyperparameter** | **Value** |
|---|---|
| MHA Head Sizes | 256 |
| No. of MHA Heads | 4 |
| Encoder & Decoder Blocks | 4 |
| Feed Forward Dimension | 4 |
| Dropout Factor | 0.4 |
| Batch Size | 64 |
| Epochs | 50 |
| Early Stopping Factor | 0.1 |

# 5
# Results and Analysis

In this chapter, the results of the experiments described in Chapter 3 are presented. Five categories of models have been trained following the outline of Chapter 4, each with the purpose of predicting proxy CDS spreads for counterparties of a specific credit rating. To reiterate from Section 3.3.2, the following results are provided for each category:

- Statistical metrics for the average, best and worst performing model of the category.
- Error plots showing the deviations from the actual market spreads for the proxy spreads generated by the best performing Transformer model instance, as well as for the corresponding spreads of the Nomura model.
- A plot showing the actual market spreads, Transformer generated proxy spreads, and Nomura model proxy spreads for all tenors on a single day, i.e. full spread curves.
- A plot showing the evolution over time of the actual market spreads, best performing Transformer model's proxy spreads, and Nomura model's proxy spreads for various tenors.

The chapter also includes a comparison between the results of the different categories as well as the results obtained when exposing the AA category of models to an increased training time.

## 5.1 AA Category

Ten model instances were trained within the AA category, each using different OOS-target companies as test sets. Table 5.1 shows Spearman's Rank Correlation Coefficient, the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) averaged over all ten Transformer model instances as well as for the best and worst performing model instance of the AA category (with Transformer denoted TF). The values inside the parenthesis show the corresponding metrics of the Nomura model for the same period. To be noted, the ranking of the models from best to worst was done using a joint score of the three metrics, with a Spearman coefficient close to +1 in combination with a low MSE and low MAE warranting a higher score (see Section 2.4 for details about each metric).

**Table 5.1:** Statistical metrics for the AA category.

| Metric | TF Average (Nomura) | TF Best (Nomura) | TF Worst (Nomura) |
|---|---|---|---|
| Spearman Coefficient | 0.86 (0.72) | 0.96 (0.74) | 0.82 (0.58) |
| MSE (bps) | 40.14 (71.14) | 18.64 (92.01) | 75.72 (99.75) |
| MAE (bps) | 4.75 (6.06) | 3.47 (7.30) | 5.18 (6.15) |

The reported metrics of Table 5.1 indicates that the Transformer model outperforms the Nomura model across average, best, and worst scenarios when generating spreads for this category, indicating a consistently better ability to rank CDS spreads in order of risk. The best Transformer model (0.96) significantly surpasses the corresponding Spearman coefficient of the Nomura model (0.74), suggesting that the Transformer model, when at its best, is much closer to an ideal ranking. Moreover, the best Transformer model shows an MSE of 18.64 bps, much lower than Nomura's 92.01, highlighting much better accuracy in the best-case scenario. The worst Transformer model's MSE (75.72) is also lower than Nomura's (99.75), indicating that even the least accurate instance of the Transformer model is more reliable than the least accurate Nomura model when generating spreads for AA rated counterparties according to these tests. Regarding MAE, the Transformer models outperform the Nomura benchmark across all scenarios, also showcasing the overall better accuracy achieved by the Transformer.

Figure 5.1 shows the error distributions of the best performing Transformer model of the AA category (blue bins) when predicting spreads for various tenors, as well as the corresponding errors of the Nomura model (red bins).

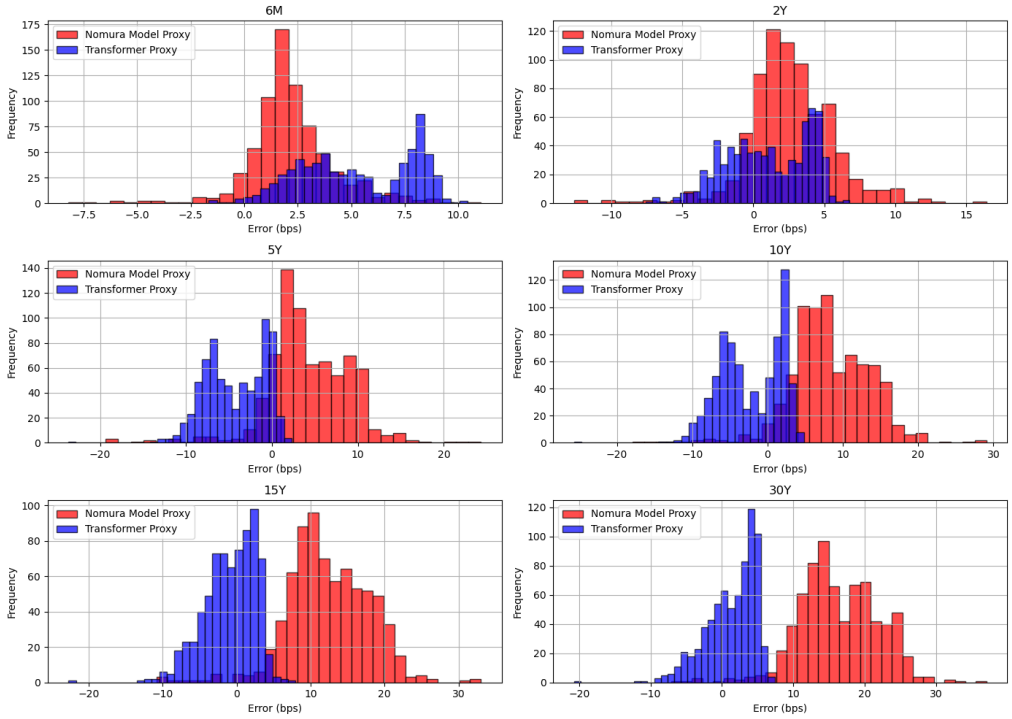


**Figure 5.1:** Error distributions for different tenors for best model of the AA category.

What the histogram's in Figure 5.1 show, is that the Transformer model generally has a tighter distribution of errors around zero for most tenors compared to the Nomura model, suggesting more accurate and consistent performance. The Nomura model's error distribution tends to be broader and more skewed, especially in longer tenors such as for 15 and 30 year maturities, which might indicate less stability or higher variability in performance for those cases.

In Figure 5.2, full spread curves over all tenors on a single day is shown. The green curve shows the target curve of the OOS-company used as test set, the blue curve the proxy curve produced by the best performing Transformer model and the red curve the proxy curve of the benchmark Nomura model.

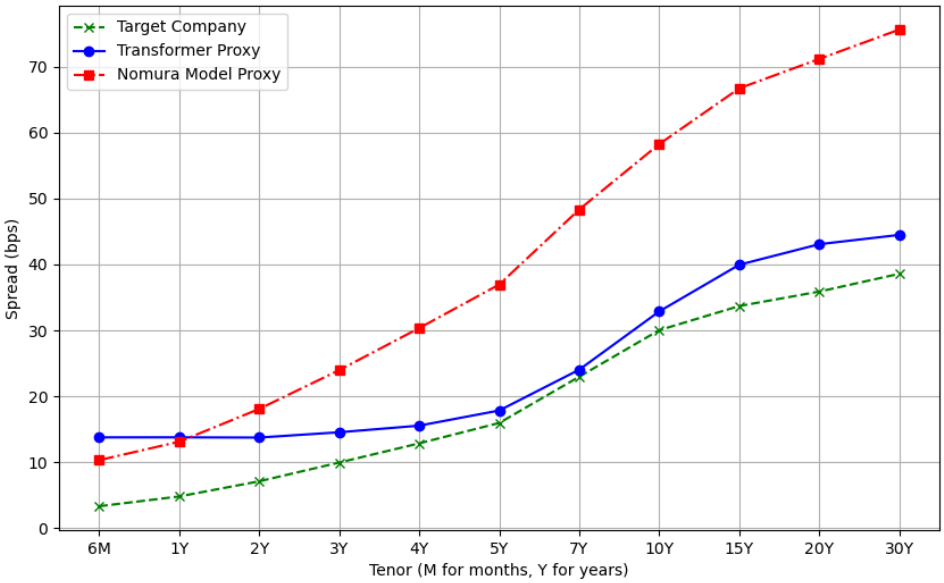


**Figure 5.2:** Target and proxy spread curves for all tenors on a single day for the best model of the AA category.

On this particular day, the Transformer model's spreads closely follow the target company's spreads across all tenors as can bee seen in Figure 5.2. In contrast, the Nomura model generally seems to overestimate the spreads, particularly in mid to long tenors (from 5Y and onwards).

Finally, Figure 5.3 shows the progression over time for the spreads of various tenors. Again, the green curves shown the actual market spreads of the OOS-target company, the blue curves the proxies of the best performing Transformer model and the red curves the corresponding proxies of the Nomura model.

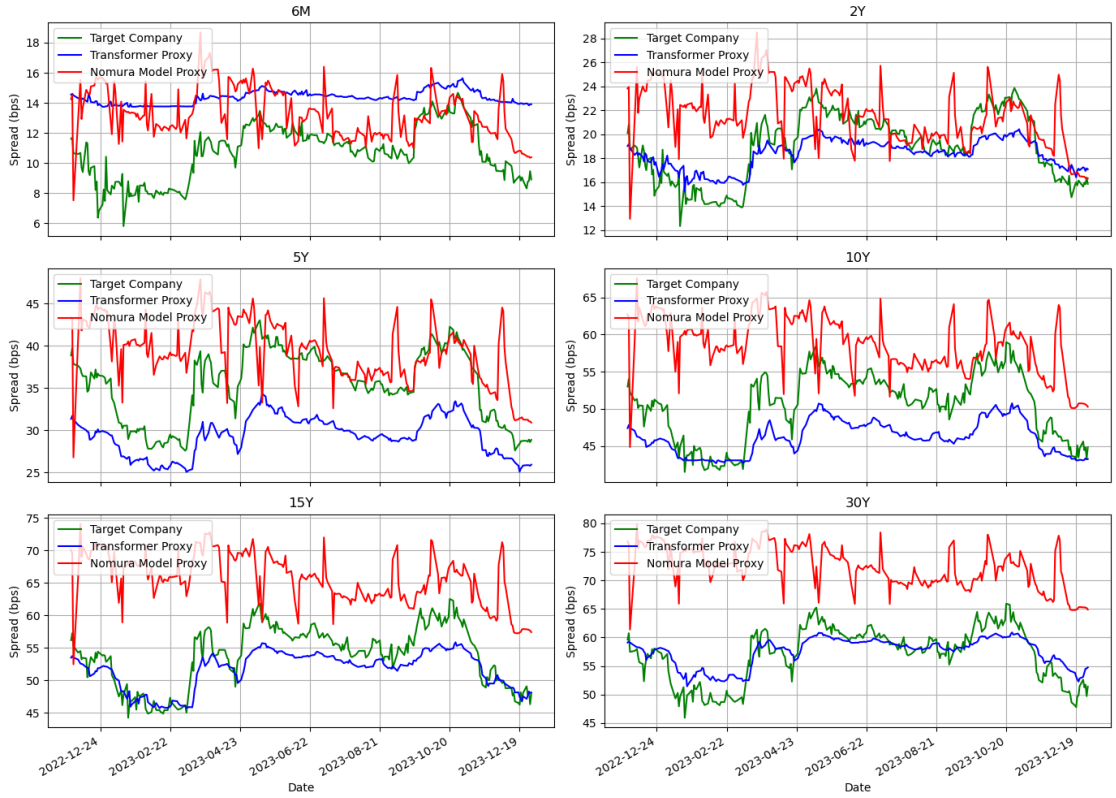


**Figure 5.3:** Time series plots of target and proxy spreads for different tenors for the best model of the AA category.

The Transformer model tracks the target company's spread changes over time with reasonable accuracy, maintaining closer proximity than the Nomura model. The Transformer also appears to adjust more rapidly than the Nomura model which suggests better responsiveness to market dynamics. However, the Transformer and Nomura models have different approaches to proxy CDS spread generation, which likely affects their behavior over time. The Transformer generates proxy CDS spreads by learning from historical sequences to try and capture temporal trends and shifts, and the Nomura model generates proxy CDS spreads based on cross-sectional factors without focusing on temporal trends. Despite these differences it felt reasonable to include the Nomura model alongside the Transformer in this evaluation since it is a common industry choice for generating proxy CDS spreads for illiquid counterparties, thus making it a benchmark to compare with. However, the difference in methodology should be kept in mind when evaluating the results to be as fair as possible when reviewing the strengths and weaknesses of each model compared to the other.

## 5.2 A Category

Similar to the AA category, ten model instances were trained for the A category as well. The average, best and worst performance indicated by the three statistical metrics tested are presented in Table 5.2.

**Table 5.2:** Statistical metrics for the A category.

| Metric | TF Average (Nomura) | TF Best (Nomura) | TF Worst (Nomura) |
|---|---|---|---|
| Spearman Coefficient | 0.92 (0.75) | 0.96 (0.81) | 0.94 (0.70) |
| MSE (bps) | 134.43 (270.46) | 91.99 (81.67) | 494.87 (1175.69) |
| MAE (bps) | 8.12 (11.52) | 7.09 (6.24) | 14.86 (29.72) |

What Table 5.2 shows, is that the Transformer model consistently outperforms the Nomura model in terms of Spearman coefficient across average, best, and worst cases, indicating a stronger ability to correctly rank the CDS spreads. What is especially interesting, is that the best (0.96) and worst (0.94) Transformer models of the sample show less variability in performance compared to the Nomura model's 0.81 and 0.70, suggesting a higher consistency in the Transformer's ranking ability compared to the benchmark. Furthermore, the average Transformer MSE is much lower compared to Nomura's. To be noted here is that both models exhibit a high worst-case MSE, but the Transformer's is notably lower, indicating it maintains better control over errors under less ideal conditions compared to the Nomura model. The average of the Transformer models in the A category also achieves a lower MAE (8.12) compared to the Nomura model (11.52), indicating an effectiveness of the Transformer model in minimizing absolute prediction errors when generating proxy spreads for A rated counterparties.

Figure 5.4 shows the error distributions of the best performing model of the A category along with the corresponding error distributions of the Nomura model.

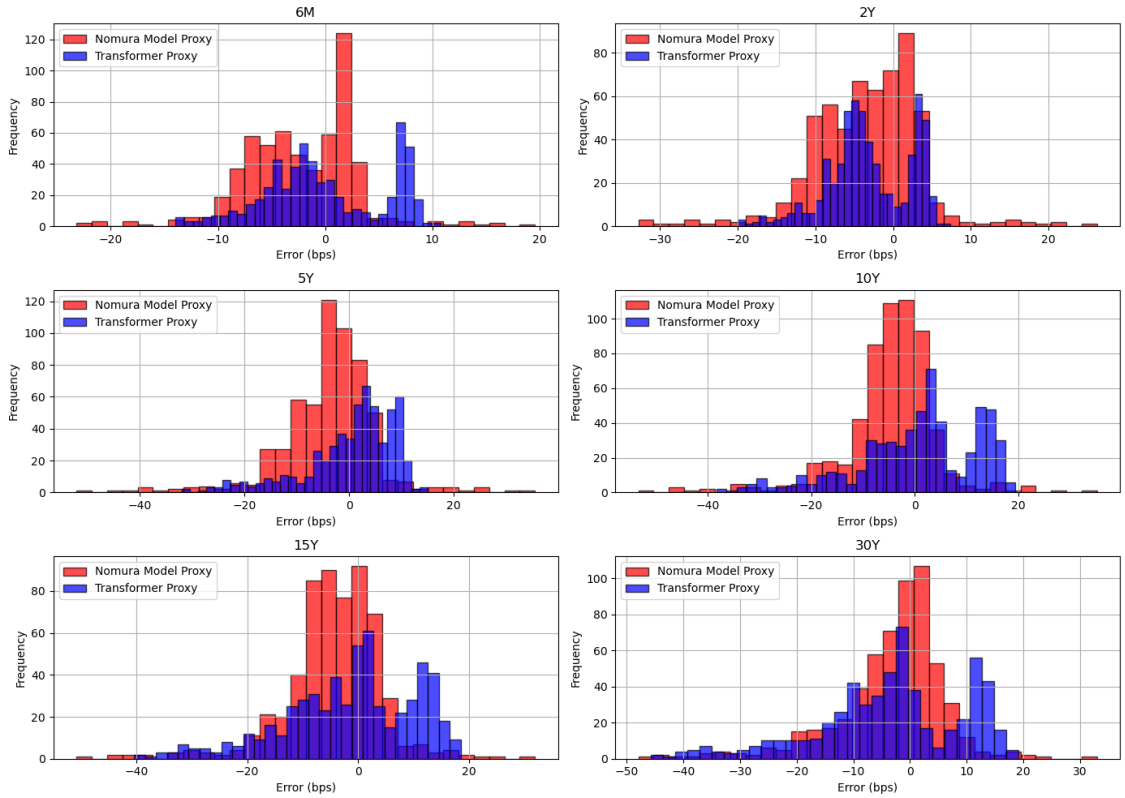


**Figure 5.4:** Error distributions for different tenors for best model of the A category.

The histograms of Figure 5.4 shows that the Transformer model typically has a more concentrated distribution around zero, which indicates more accurate and consistent outputs in these cases. For both models however the error frequencies span further from zero with longer tenors.

Figure 5.5 show the full spread curves for a single day for the best performing model instance of the A category, as well as for the target OOS-company and the Nomura model.

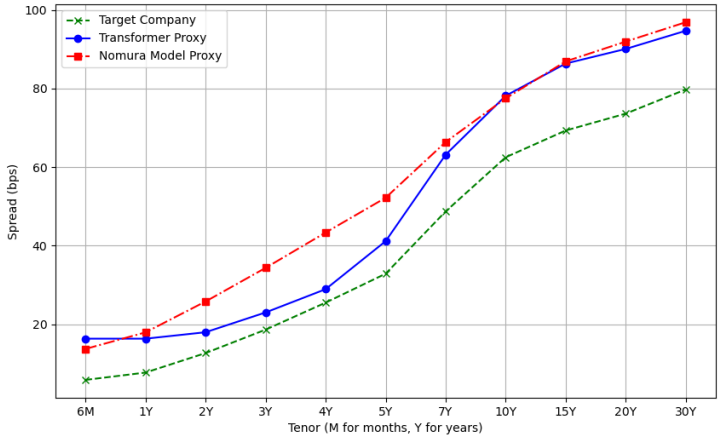


**Figure 5.5:** Target and proxy spread curves for all tenors on a single day for the best model of the A category.

On the displayed day, the Transformer model's proxy spreads across all tenors rather closely align with the target company's actual spreads, while the Nomura model tends to overestimate the spreads similar to what was done in the AA category.

Moreover, the spread progression over time for the best model of the A category along with the corresponding actual target spreads and benchmark Nomura model spreads are shown in Figure 5.6.


**Figure 5.6:** Time series plots of target and proxy spreads for different tenors for the best model of the A category.

The plots in Figure 5.6 demonstrate the Transformer's ability to closely track the target company's spread fluctuations over time compared to the Nomura model, especially for shorter tenors.

## 5.3 BBB Category

Also for the BBB category ten model instances were trained using different OOS-companies as test sets. Table 5.3 presents the statistical metrics for the average performance across the BBB sample, as well as for the best and worst performing models. Similar to before, the corresponding benchmark values of the Nomura model are enclosed within the parenthesis.

**Table 5.3:** Statistical metrics for the BBB category.

| Metric | TF Average (Nomura) | TF Best (Nomura) | TF Worst (Nomura) |
|---|---|---|---|
| Spearman Coefficient | 0.91 (0.66) | 0.89 (0.42) | 0.62 (0.28) |
| MSE (bps) | 415.74 (470.18) | 194.62 (495.98) | 983.13 (1577.74) |
| MAE (bps) | 16.35 (15.44) | 11.00 (17.83) | 22.69 (29.91) |

One observes in Table 5.3 that the Transformer model significantly outperforms the Nomura model in Spearman correlation across all cases, showing its robust ability to accurately rank the risks as the grade of the credit rating category begins lower. For example, the worst-case performance for the Transformer (0.62) more than doubles that of the Nomura model (0.28). What is furthermore noticeable here compared

to the AA and A rating categories, is that both models (Transformer and Nomura) show higher MSE values which implies that the size of the errors increase with lower graded rating categories, but the Transformer's average and best MSE are considerably lower than those of the Nomura model in this category. The same can also be said regarding MAE, with the Transformer still outperforming the Nomura benchmark, but with larger overall errors for both the Transformer and Nomura model compared to the AA and A categories which indicates potential issues to get accurate spread values in this category.

Interestingly, one can then proceed with a visual observation of the error distributions of the best performing Transformer model and corresponding error distributions of the Nomura model as shown in Figure 5.7.

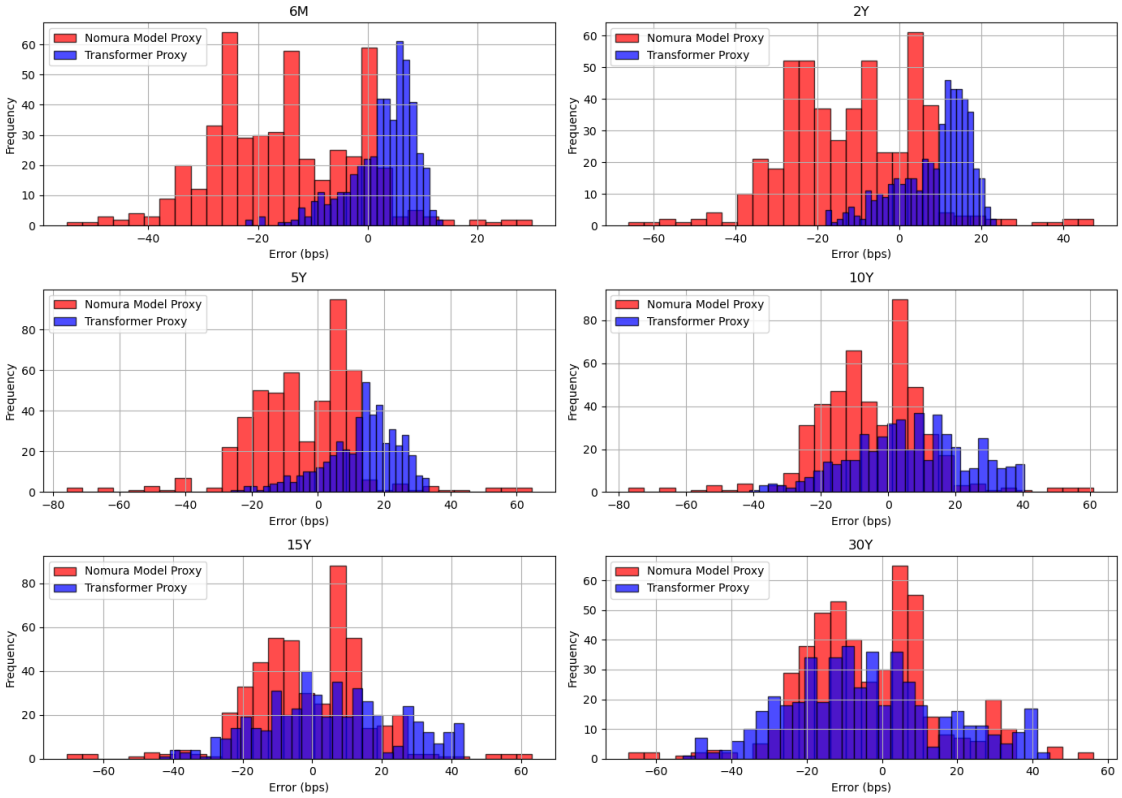


**Figure 5.7:** Error distributions for different tenors for best model of the BBB category.

The Nomura model shows a wider error distribution across most tenors, particularly in the 2Y and 10Y cases, which indicates less precision. Also, as mentioned earlier, the errors begin to increase in size compared to the results for the higher credit ratings for both models, but with the distribution being more narrowed around zero for the Transformer model on average compared to the benchmark Nomura model.

Similar to before, an example of a single day spread curve across all tenors for the best performing Transformer model instance of the BBB category along with the targets and Nomura benchmark can be observed. These are shown in Figure 5.8.

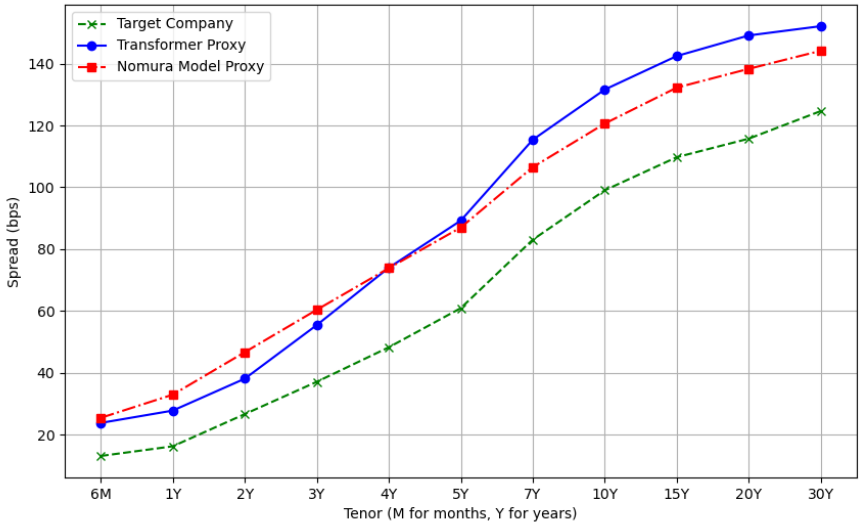


**Figure 5.8:** Target and proxy spread curves for all tenors on a single day for the best model of the BBB category.

One notices that for this particular day, both the Transformer and Nomura predictions rather closely match the target company's actual spreads across all tenors. Interestingly, one may therefore also look at how the progression looks over time in the different tenors as illustrated in Figure 5.9.

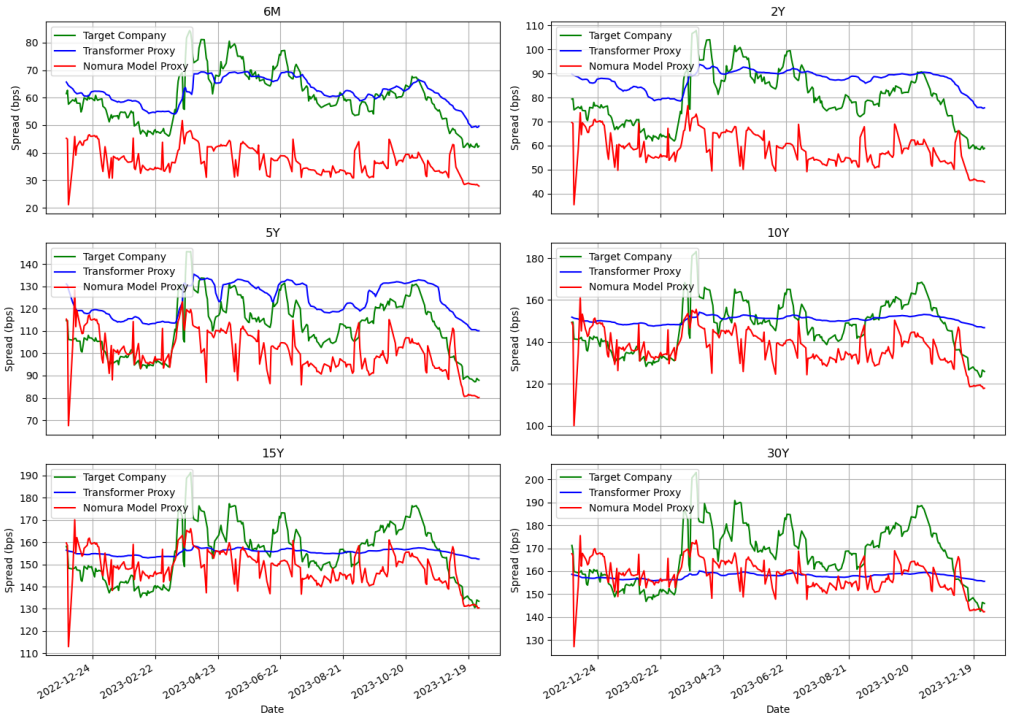


**Figure 5.9:** Time series plots of target and proxy spreads for different tenors for the best model of the BBB category.

Both models struggle during periods of sharp peaks (e.g., mid-March in the 15Y plot). Moreover, the profile of the Transformer model's curves flatten considerably as the tenor begins to increase, perhaps indicating that it has a harder time capturing those dynamics in the BBB rating category compared to the previous rating categories. The Nomura model shows more overall volatility compared to the Transformer and generally underestimates the spread values of the target company, but show better tracking of the target curve's profile in the longer tenors compared to the Transformer in this category.

## 5.4 BB Category

The BB category also consisted of 10 model instances in total, each tested on a unique OOS-target company similar to the procedure of the previously presented rating categories. Table 5.4 show the average, best and worst results for the Spearman coefficient, MSE and MAE.

**Table 5.4:** Statistical metrics for the BB category.

| **Metric** | **TF Average (Nomura)** | **TF Best (Nomura)** | **TF Worst (Nomura)** |
|---|---|---|---|
| Spearman Coefficient | 0.71 (0.51) | 0.99 (0.84) | 0.24 (0.72) |
| MSE (bps) | 2708.06 (5492.17) | 936.52 (2006.78) | 6715.9 (5671.68) |
| MAE (bps) | 36.38 (62.15) | 21.97 (33.12) | 54.4 (54.89) |

For the BB category, as shown in Table 5.4, the Transformer model shows very good performance in the best-case scenario with a Spearman coefficient of 0.99, substantially higher than the Nomura's benchmark at 0.84. There is however quite high variability in the Transformer's performance, with the worst case dropping to 0.24 compared to Nomura's 0.72. The best Transformer model achieves a lower MSE of 936.52 compared to Nomura's 2006.78, indicating better accuracy, but the worst-case MSE is quite high for both models and particularly severe for the Transformer, showing possible vulnerability when generating spreads for BB rated counterparties. The Transformer model reports lower MAE across average and best scenarios compared to the Nomura benchmark, highlighting more consistent and accurate predictions, although the worst-case MAE does not outperform the Nomura model by much (54.4 as opposed to 54.89). Considering the vast difference between the average, best and worst case performances of the Transformer model in this category it should possibly be reviewed with some skepticism before firmly declaring its general superiority over the Nomura benchmark.

Similar to the previous categories, some visuals have been included for the best performing Transformer model instance of the BB category as well. First of all, Figure 5.10 shows the error distributions across various tenors.

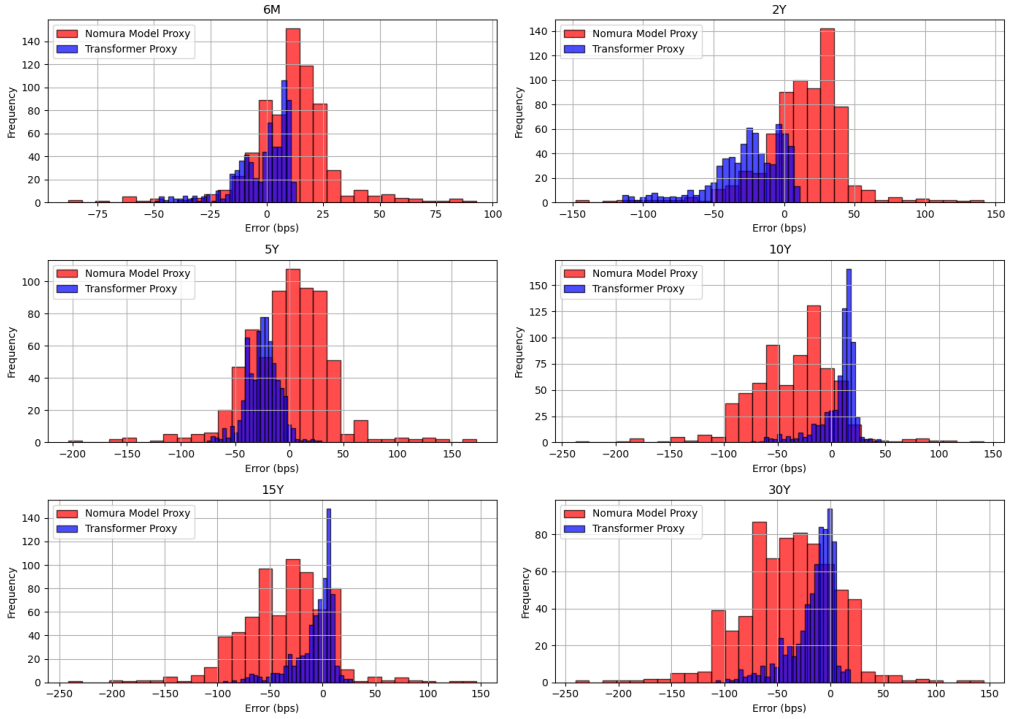


**Figure 5.10:** Error distributions for different tenors for best model of the BB category.

Figure 5.10 shows some frequency of large errors in the Transformer's spreads, especially in the 2Y and 5Y tenors where a skewness towards the negative side of the spectrum can be observed. However, the Transformer generally maintains a much more narrow distribution around zero compared to the Nomura model when predicting spreads for BB rated counterparties.

Figure 5.11 show a full single day proxy spread curve across all tenors for the best model of the BB category along with the target company's actual market spreads and Nomura model proxies for the same day.

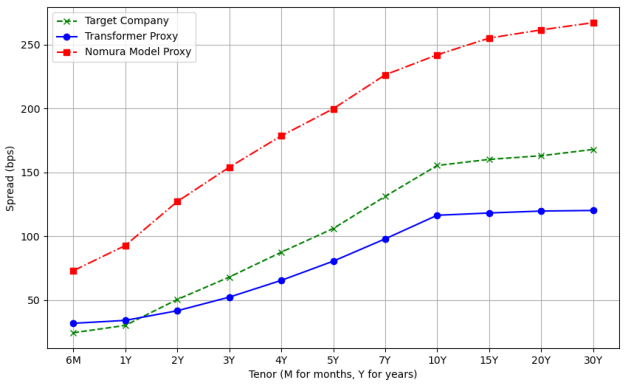


**Figure 5.11:** Target and proxy spread curves for all tenors on a single day for the best model of the BB category.

One this specific day, the Transformer model closely aligns with the target company's actual spreads, especially in shorter tenors, suggesting good single day-specific performance. One also notices that compared to the Nomura model, the shape of the Transformer predicted curve is more well-aligned with the target curve, which can perhaps be traced back to the very high Spearman coefficient of this model which indicates that the curve dynamics have been learned well.

The very high Spearman coefficient of the best BB model can also be perceived visually in the time series plots of Figure 5.12.

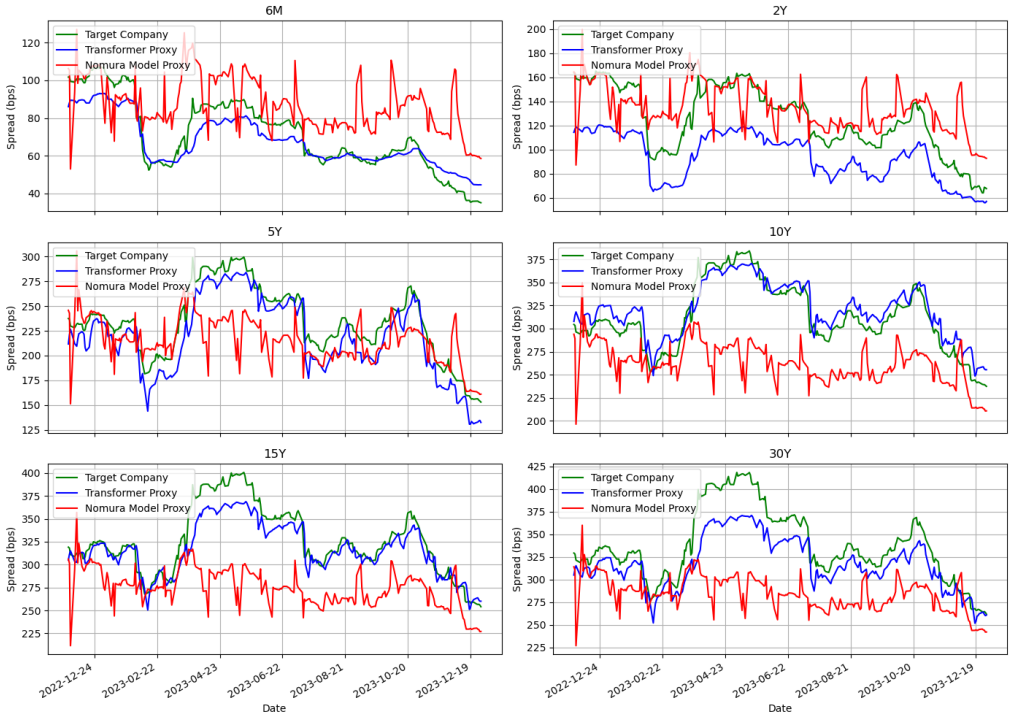


**Figure 5.12:** Time series plots of target and proxy spreads for different tenors for the best model of the BB category.

In Figure 5.12 one clearly sees that the Transformer aligns very well with its targets throughout all tenors. Especially well results can be observed in the 10Y tenor, whereas the 2Y tenor for example shows a similar profile but a consistent discrepancy in absolute values compared to the target curve. The Nomura model however shows a much more fluctuating and erratic profile for all tenors in this category.

## 5.5 B Category

The final category of models were the ones trained to predict proxy CDS spreads for B rated counterparties. This rating category contained the least amount of liquid counterparties which meant that the amount of data to generate labels from as well as OOS-companies to extract as test sets for the different model instances were the most scarce out of the five categories of focus. In total, six model instances were

trained for the B category and the average, best an worst results are presented in Table 5.5. Considering the much smaller sample of trained models in this category compared to previous ones the statistical significance of these results should be regarded with less certainty.

**Table 5.5:** Statistical metrics for the B category.

| **Metric** | **TF Average (Nomura)** | **TF Best (Nomura)** | **TF Worst (Nomura)** |
|---|---|---|---|
| Spearman Coefficient | 0.86 (0.38) | 0.97 (0.81) | 0.88 (0.30) |
| MSE (bps) | 16157.5 (23808.87) | 18363.65 (11811.84) | 57953.45 (30095.14) |
| MAE (bps) | 92.47 (123.02) | 89.99 (74.99) | 219.51 (91.97) |

In Table 5.5 one notices a consistently high performance of the Transformer in generating accurate proxy spreads for B rated counterparties compared to the Nomura model benchmark. The Transformer consistently outperforms the Nomura model in terms of Spearman coefficient across all cases, particularly in the best case with 0.97 compared to the Nomura model's 0.81. This demonstrates a strong ability to rank the proxy CDS spreads accurately. Especially in the worst-case scenario, the Transformer maintains a high Spearman coefficient of 0.88 which far beats the benchmark of 0.30 for the same period. In terms of MSE one notices that while the average of the Transformer models outperforms the Nomura model, the best and worst cases do not which might indicate that more severe outliers greatly impacts the metric in these cases. The same can be said regarding MAE. It is noteworthy here to once again repeat that the best and worst model instances for each of the categories were determined by a joint score over all three metrics, which stands to explain why the best model for example does not necessarily achieve a better score on each metric individually compared to the average over all models as can be observed here.

Moving on to some visual inspections, again one can start by observing the error distributions of the best performing model of the category over various tenors. This can be observed in Figure 5.13.

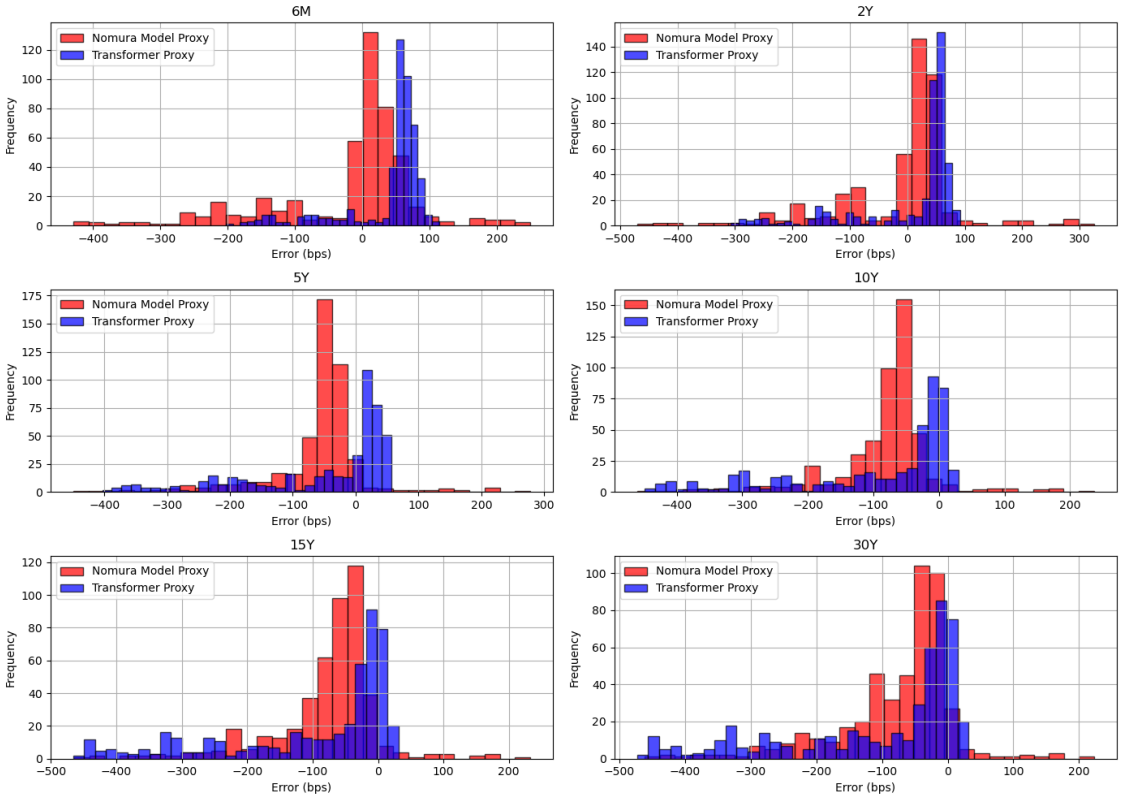


**Figure 5.13:** Error distributions for different tenors for best model of the B category.

Aligning with the reported MSE and MAE metrics, one notices in Figure 5.13 how the frequency of large errors is quite high for both the Transformer and the Nomura model. The histograms are generally skewed to the negative side of the spectrum indicating that the spread values tend to be underestimated. For the Transformer model, these dynamics are particularly noticeable in long tenors with an increase of the large error frequencies.

Next, the single day spread curves of the best Transformer proxies, the corresponding Nomura proxies and actual target spreads of the OOS-company can be observed in Figure 5.14.

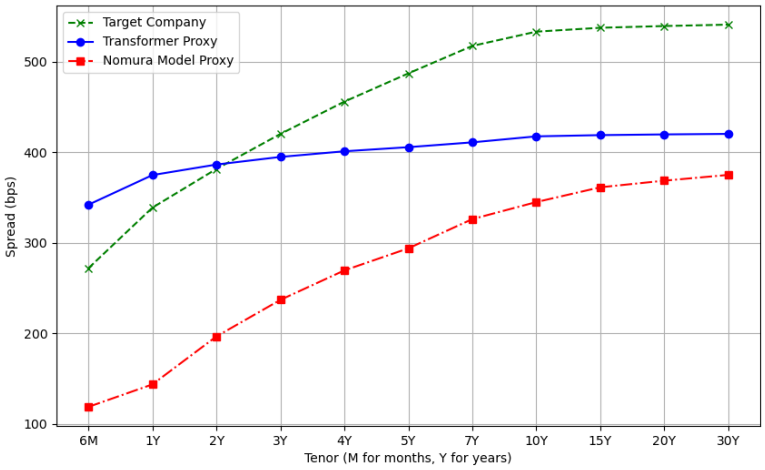


**Figure 5.14:** Target and proxy spread curves for all tenors on a single day for the best model of the B category.

As was noted in the error distributions of Figure 5.13, what the curves in Figure 5.14 also show is an underestimation of the actual spreads on this particular day, both by the Transformer and Nomura model. The Transformer is closer in absolute terms, but for longer tenors its curve profile seems less aligned compared to the Nomura model.

Finally, one can observe the progression over time for the spreads of some of the tenors generated for B rated counterparties. This is presented in Figure 5.15.

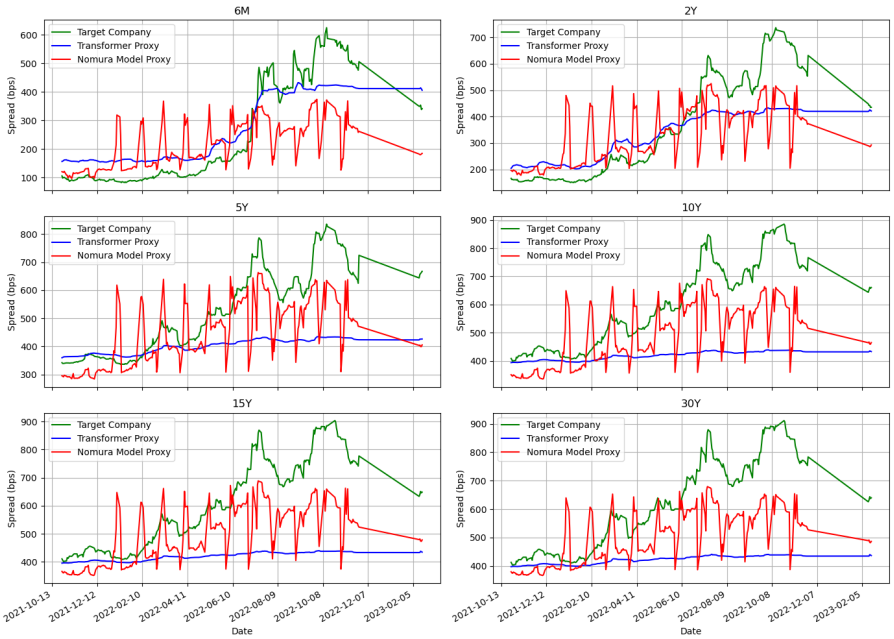


**Figure 5.15:** Time series plots of target and proxy spreads for different tenors for the best model of the B category.

Compared to previous rating categories, one clearly observes difficulties in tracking the targets for both the Transformer and Nomura model, with both models generally underestimating the actual spreads in absolute terms as the previous figures have also shown. However, the shapes of the curves differ. While the Nomura model fluctuates frequently and does not capture the profile of the target curve very well, the Transformer model has a much more even progression. In the shorter tenors of 6M and 2Y one can see that the Transformer follows the target profile rather well until approximately July 2022. Thereafter, it has troubles following the actual target spreads. For the longer tenors one can see that the Transformer slightly has the same upwards trend as the targets, although the absolute difference between the curves is quite high as has already been discussed. The strong fluctuations observed for the Nomura model may be due to factors such as less liquid spread data in this category to base the estimations on compared to higher graded rating categories, and higher volatility in general in lower-rated categories that are typically less liquid. Additionally, the model's reliance on frequent recalibration based on cross-sectional factors can likely cause sudden changes in less liquid markets, which potentially can explain the instability seen in the graphs.

## 5.6 Comparison Across Rating Categories

Following the prior analysis of the model categories individually, some brief comparisons between them can be made. Table 5.6 show the average performance of the Transformer in each category in terms of Spearman's Rank Correlation Coefficient, MSE and MAE alongside each other. Figure 5.16 show the same metrics visualized in bar charts (note that the y-axes of the two rightmost panels are presented in logarithmic scale).

**Table 5.6:** Average statistical performance of each model category.

| **Model Category** | **Av. Spearman Coefficient** | **Av. MSE (bps)** | **Av. MAE (bps)** |
|---|---|---|---|
| AA | 0.86 | 40.14 | 4.75 |
| A | 0.92 | 134.43 | 8.12 |
| BBB | 0.91 | 415.74 | 16.35 |
| BB | 0.71 | 2708.06 | 36.38 |
| B | 0.86 | 16157.5 | 92.47 |

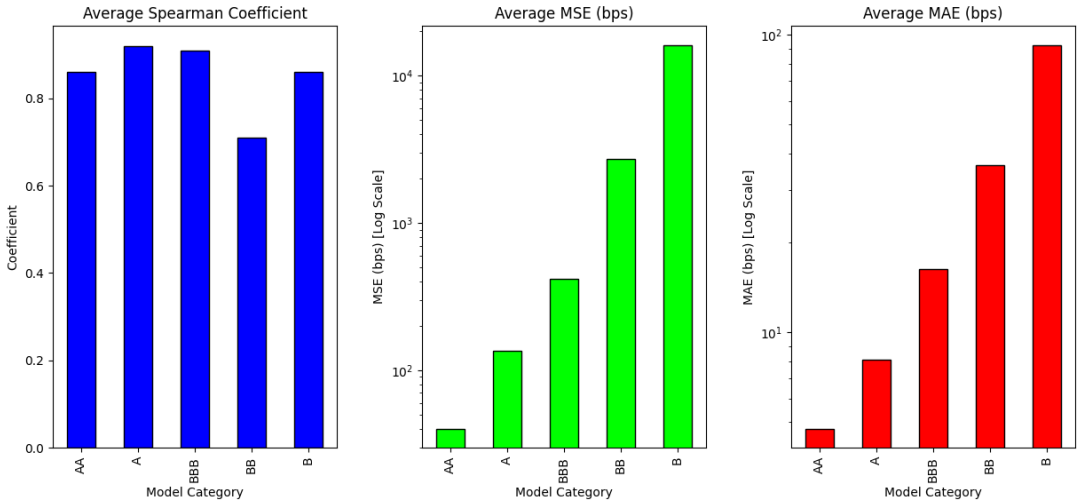


**Figure 5.16:** Visual representation of the average statistical performance of each model category.

Looking at Table 5.6 and Figure 5.16, one can notice a trend in performance drop as the rating grade decreases. First of all, the correlation with the respective targets is generally quite high across ratings as shown by the Spearman coefficient. This indicates that the models in each category has achieved pretty good results in predicting the rank order of the data accurately. The BB model however shows a rather big deviation to the others which might indicate a weaker or less consistent rank correlation in its outputs on average in these runs. Looking at the error metrics on the other hand there is a clear trend in that errors increases as the rating grade decreases. AA for example, which is the rating category with highest investment grade, shows the best performance with the lowest MSE of 40.14 bps on average, which highlights its accuracy in outputs with the least error magnitude, while the rating category of lowest investment grade, B, shows a very high MSE of 16157.5 bps on average, indicating large error magnitudes. The same can also be observed in MAE where AA again shows the best performance with 4.75 bps, compared to B's 92.47 bps.

Apart from the metrics observed above one can also look at the data amount of each rating category in the training dataset. Recall from Section 4.2.1 that all models were trained on data from any rating category, but with target labels targeted to a specific one. Upon merging the training inputs with the labels of the targeted rating category, this meant that the amount of training records available for each category depended on the number of available labels for it since the label and input dataframes were merged on the basis of similar business dates. Table 5.7 below shows the number of training records of each rating category, and Table 5.8 shows the number of available counterparties per rating category.
When doing an observation of Table 5.7, one sees that the amount of available training data decreases along with the investment grade of the rating category. Moreover, this seems to be related to the fact that there is a larger amount of

**Table 5.7:** Training records per rating category.

| Rating Category | Training Records |
|---|---|
| AA | 47 771 |
| A | 48 637 |
| BBB | 48 637 |
| BB | 42 153 |
| B | 21 041 |

**Table 5.8:** Number of counterparties per rating category.

| Rating Category | No. of Companies |
|---|---|
| AA | 26 |
| A | 43 |
| BBB | 41 |
| BB | 32 |
| B | 13 |

companies (counterparties) in the high investment grade cases compared to the lower ones, as seen in Table 5.8. The difference in available training data may thus serve as part of the explanation to the variation in performance over the different categories which the Transformer model was trained to generate proxy CDS spreads for.

## 5.7 Increased Training Time

An additional experiment was conducted in which the training time was increased in order to see what potential effect it had on model performance (see Section 3.3.1.1). In all the previous categories, the hyperparameters of the model were fixed to be able to draw general conclusions about model performance across different categories. These hyperparameters were presented in Section 4.3.5. For this experiment with increased training time, the number of epochs was set to 1000 instead 50 (see original hyperparameters in Table 4.3), to allow the model a lot of time for convergence. The factor for the early stopping criterion remained at 0.1, effectively allowing a patience of 100 epochs for the experiment with increased training time.

The experiment with an increase in training time was only done for the AA category. Considering the longer training time, only three models were trained in total, which means less statistical significance of the results which should be kept in mind. Table 5.9 shows the Spearman's Rank Correlation Coefficient, MSE and MAE achieved on average as well as for the best and worst performing model instance in this test.

**Table 5.9:** Statistical metrics for the AA category with increased training time.

| Metric | TF Average (Nomura) | TF Best (Nomura) | TF Worst (Nomura) |
|---|---|---|---|
| Spearman Coefficient | 0.92 (0.71) | 0.95 (0.61) | 0.89 (0.77) |
| MSE (bps) | 24.16 (99.82) | 27.23 (84.41) | 30.21 (123.04) |
| MAE (bps) | 3.95 (7.68) | 4.28 (7.43) | 4.52 (8.29) |

Reviewing the results of Table 5.9, one sees that across the average, best and worst performing model the Spearman coefficient is generally quite high, especially compared to the Nomura model. On average, a Spearman coefficient of 0.92 is achieved, demonstrating good model performance in capturing the rank order. This can also be compared to the average performance of the AA model without an increase in training time, which achieved a Spearman coefficient of 0.86 (see Table 5.1). The best model with an increase in training time achieved 0.95 which roughly matches that of the best performing model without increased training time (0.96). The MSE and MAE is reported lower compared to the Nomura model across average, best and worst case. Compared to the AA category with normal training time one can see a slight outperformance in MAE on average although not by much (3.95 compared to 4.75, see Table 5.1). What is interesting is that the average MSE is roughly half that of the same category with normal training time (24.16 compared to 40.14) which suggest less severity of outliers.

Looking at the error distributions of the best performing model with increased training time as presented in Figure 5.17 one can gain some visual insights of its results.

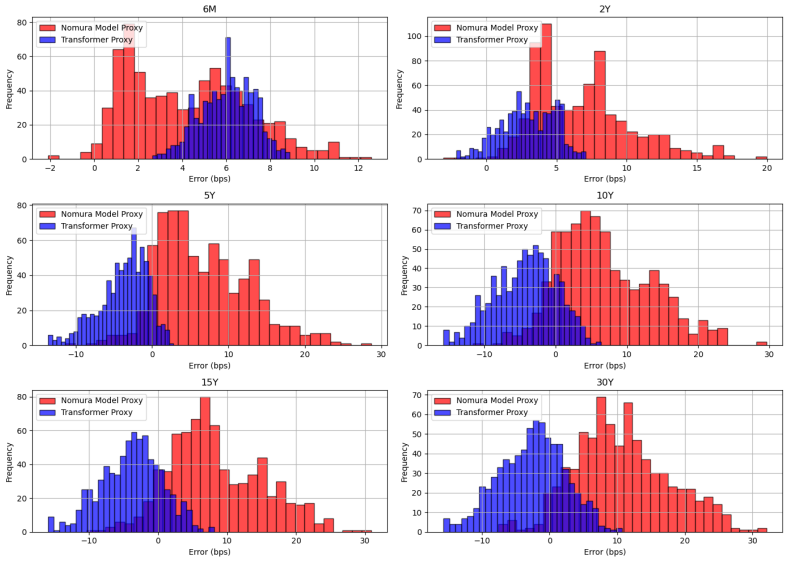


**Figure 5.17:** Error distributions for different tenors for the best model of the AA category with increased training time.

First of all, compared to the Nomura benchmark, a much more even distribution is observed with less outliers. This can be also be seen compared to the same category with normal training time (see Figure 5.1 for comparison). This visually complements the observation of the big MSE difference discussed in the paragraph above. The histograms of Figure 5.17 are a bit skewed however, with distributions indicating an overestimation of the spreads for shorter tenors (6M and 2Y), and underestimation for longer tenors (5Y, 10Y, 15Y and 30Y). The distribution best centered around zero is that of 30Y, while for the normal training time this is the case for the 15Y tenor instead (again, see Figure 5.1 for a comparison with normal training time).

Figure 5.18 show full spread curves for the best performing model with an increased training time along with its target and Nomura model benchmark.

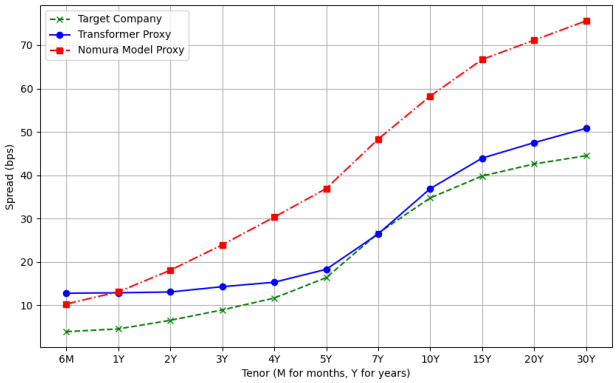


**Figure 5.18:** Target and proxy spread curves for all tenors on a single day for the best model of the AA category with increased training time.

One sees that the model accurately follows a similar profile to that of the target and particularly accurate performance can be observed in the 5Y and 7Y tenor, whereas the shorter and longer tenor spreads are a bit offset in absolute terms. Compared to the Nomura model however, both the curve profile and absolute difference to the target appears to be better for the Transformer model.

Finally, Figure 5.19 shows the progression of the spreads for various tenors over time for the best AA model instance with increased training time along with its target and Nomura benchmark time series curves.

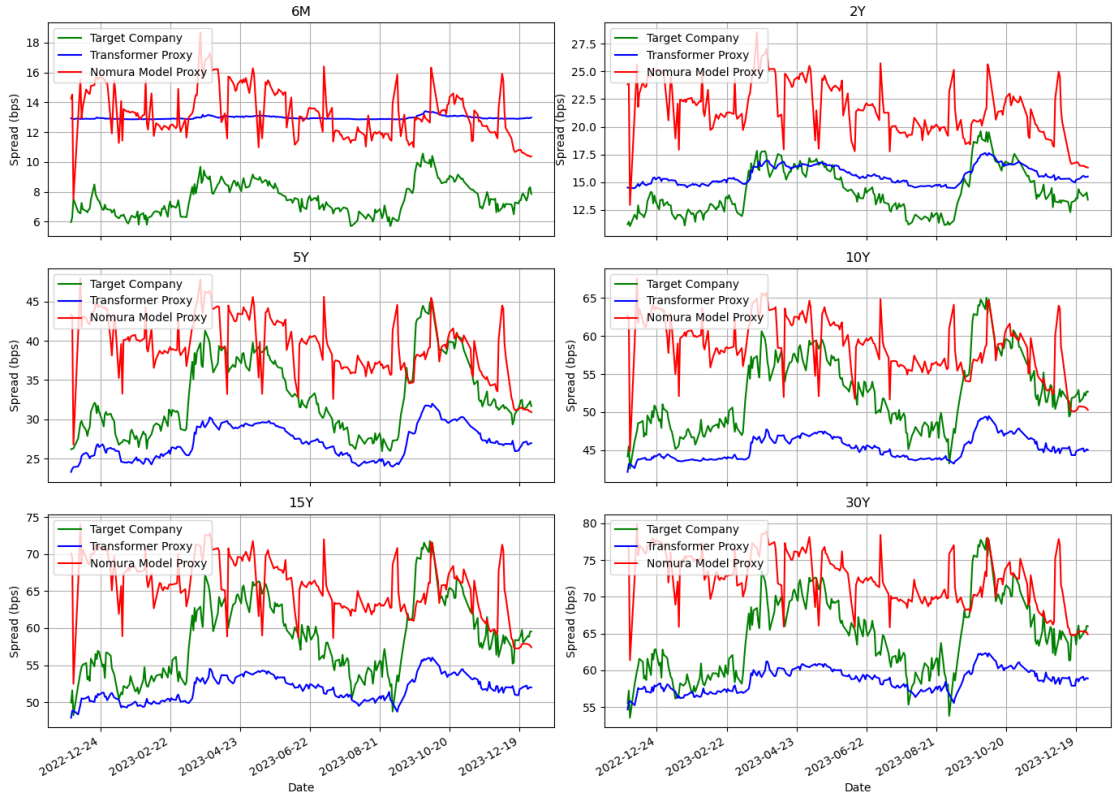


**Figure 5.19:** Time series plots of target and proxy spreads for different tenors for the best model of the AA category with increased training time.

Although the Transformer appears to have difficulties in generating exact spread levels across all tenors it generally follows the profile of the target curve, one exception being for the 6M tenor. The Nomura model show much more erratic and fluctuating behaviour in comparison though and tend to overestimate spread levels compared to the target curve in most instances.

# 6
# Discussion

In this chapter, a discussion is carried out regarding the outcome of the thesis in relation to the aim and issue of investigation presented in Sections 1.2 and 1.4. All in all, this boils down to answering the research question (RQ) posed in the beginning of this thesis:

- RQ: To what extent can Transformer-generated proxy CDS spread curves more accurately simulate market based CDS spread curves, compared to those produced by traditional linear regression models?

And attempt at answering this as well as a discussion regarding various aspects of the thesis implementation are provided here.

## 6.1 Discussion on Model Performance

When comparing the performance of the Transformer model relative to the cross-sectional Nomura model benchmark, the results presented in Chapter 5 reflect a positive outcome in favor of the Transformer. By observation of the Spearman coefficient, MSE and MAE metrics along with the presented visualizations it is clear that in the experiments conducted here, the Transformer model has demonstrated superior ability in generating accurate proxy CDS spreads across the various rating categories. Unlike the Nomura model, which generally showed more volatile behavior, the Transformer model produced better curve profiles and higher accuracy with respect to error magnitudes and outliers. This likely suggests that the Transformer is better at capturing important aspects of the input data, providing a more reliable and stable prediction mechanism than the Nomura model's cross-sectional regression approach. Under the limitations of the experiments conducted in this thesis the outcomes are therefore overall deemed positive and promising with respect to the Transformer model's ability to more accurately generate proxy CDS spreads for counterparties of different rating, compared to traditionally used linear regression models such as the Nomura model.

There are however several points regarding the results which must be addressed in order for general conclusions to be drawn. Starting off, by reviewing the performance across the generation of proxy spreads for the various rating categories, one sees a notable decline in model performance correlated with the decreasing grade of the rating the model predicts proxy spreads for. Despite a generally high average Spearman correlation coefficient, the magnitude of errors and the presence of

outliers increased in lower-rated categories as shown by the increase of both MAE and MSE. This variability shows a divergence in performance stability. Especially interesting is that the difference in performance between the best and worst model of each category became higher as the rating grade decreased. This was also the case for the Nomura benchmark. The statistical metrics reported showed less variability between best and worst model instance for the higher graded rating categories, which indicates a more stable performance in general amongst them. The reason for this outcome can depend on multiple things but an important one is believed to lie in the difference of data quality and quantity for the various categories. As was noted in Section 5.6, higher graded rating categories (such as AA, A and BBB) contained more individual counterparties, as well as data records, to train on compared to the lower graded ones. Since ML-models in general perform better as the availability of training data increases, this perhaps serve as part of an explanation to the observed performance drop.

On this note it is also in its place to discuss the statistical significance of the observed results in Chapter 5. As was mentioned there, ten model instances were trained per rating category with different out-of-sample (OOS) companies used as test sets for each model instance, except for the lowest graded rating category (the B category) for which only six models were trained due to limitations in data availability. Optimally, a greater sample of individual model instances would provide a better ability to draw general conclusions about average performance. Since this was not possible due to the time and computational limitations of this project one should therefore be aware of this when reviewing the outcomes of the results. Most importantly, for the B category which only contained a sample of six individual model instances, the statistical significance is deemed quite low meaning a more comprehensive analysis is in its place in order to be able to draw firm conclusions about the Transformer model's ability for generating accurate proxy CDS spreads of this category. Furthermore, this implementation has only covered a single sector and region (financial companies in Western Europe). This means that the findings reported here can not with certainty be generalized to the Transformer's generative abilities of proxy CDS spreads for similar rated companies of different region and sector belongings without exposing the model to such data during training.

Another important aspect to consider is the trade-off between model complexity and performance, with an additional emphasis on interpretability, when contrasting the Transformer model implemented in this thesis with the benchmark Nomura model. As detailed in Chapter 4, the Transformer is a deep neural network (DNN) which in this implementation contained 780 349 parameters in total. This complexity likely enabled it to capture complex patterns and nuanced relationships in the training data, leading to more accurate predictions of proxy CDS spreads compared to the much simpler Nomura model. However, the complexity of the Transformer model comes with some costs, particularly in the computational resources required for training it and its inherent lack of interpretability. Deep learning models such as the Transformer, which are often described as "black boxes", do not offer much insight into their decision-making processes [25]. As the architecture deepens, un-

derstanding how inputs are mapped to outputs becomes more difficult, complicating the analysis of the model's decision-making for humans to understand. In contrast, the Nomura model is more interpretable. It uses parameters that are related to observable market characteristics such as global, sector, region, rating, and seniority factors, upon which cross-sectional regression is applied (see Section 2.3.1). Each component of the model can be detailed and explained, which provides an advantage in contexts where regulatory compliance and stakeholder communication require transparency. Therefore, despite the Transformer model's promising results in predicting proxy CDS spreads compared to the traditional Nomura model, its practical deployment might be challenged by these interpretability issues, potentially complicating regulatory approval and stakeholder acceptance.

## 6.2 Discussion on Implementation Challenges and Improvements

Another topic of discussion is the implementation choices taken in this thesis. First of all, one can look at the hyperparameter settings which were fixed throughout the runs over the different rating categories. These were presented in Section 4.3.5. As was touched upon there, selecting optimal hyperparameters for ML-implementations is a tricky task which often requires multiple iterations and subsequent analysis of the outcomes. In this thesis, the choices were directly influenced by the hyperparameters used in the prior thesis on Transformers at SHB [29]. By having a uniform approach to hyperparameters, the comparison between results within and between the different rating categories was therefore made easier. However, this approach may have limited the model's ability to optimize performance when generating proxy spreads for each specific rating category.

An initial point to be made regarding hyperparameters was the training time allowed for the models. As was presented in Section 4.3.5, the number of epochs for each model in each category was fixed at 50 epochs in total. Added to this was an early stopping factor set at 0.1, effectively meaning that if the training spanned 5 epochs in a row without any improvement in validation loss (reported in MSE) the training would stop. Now, the reason for setting this fairly low number of epochs was due to the fact that ten model instances were trained per rating category (six for the B category), and with a total of five categories this meant that 46 model instances were trained in total. One typically wants a model to receive a lot of training time since it allows its gradients to escape local minimums, which are plateaus in performance that at a glimpse may appear as convergence points but which actually are not the model's best performing state. However, it is difficult to set an optimal epoch number beforehand since it is pretty much impossible to know when during training a model will fully converge. In this case, the choice of epochs was therefore ultimately the result of a trade-off between the training time per model versus practical feasibility considering the many models which were trained in order to ensure statistical significance of the findings. As reported in Section 5.7 of Chapter 5, an attempt at increasing the training time was however made for the AA category. In

that run, three models were trained, each receiving 1000 epochs of training time with the same early stopping factor applied (effectively meaning a patience of 100 epochs of unimproved validation loss before early stopping). However as was shown in Section 5.7, the performance did not substantially improve because of this (although the results should be reviewed with some caution for these runs given that only three model instances were trained and thereby not offering much of a statistical significance to begin with). None of the models spanned for all of their total epochs though, whether they were set to 50 as in the original experiment or 1000 as in the one with increased training time. Instead, early stopping occurred at various points during their respective training runs. What can therefore be said is that for the experiments conducted in this thesis, the number of epochs did not appear to be the single most important hyperparameter to focus on in order to achieve a better performance. Instead, it is the author's belief that potential causes for the early convergence observed should be reviewed a bit more closely.

Pondering on this, the appearance of early convergence of an ML-model (which as mentioned above was observed here) may suggest potential overfitting or inability of the model to fully capture the underlying patterns of the training data. Therefore, a dynamic approach to selecting optimal hyperparameters, possibly through methods such as grid search, could potentially enhance the models' accuracy and ability to generalize. Simpler approaches could also be applied, such as varying the dropout factor to reduce potential overfitting or using an adaptive learning rate instead of a fixed one for more efficient convergence. Furthermore, monitoring for signs of exploding or vanishing gradients that could show whether adjustments in the network architecture could be needed with respect to the early convergence noted during training could perhaps also help in making the model even better.

Regarding the network architecture of the Transformer model used in this thesis it was, as explained in Section 4.3, deliberately made to replicate that of [29], which in turn was constructed to be as similar as possible to the original architecture of [40]. This choice was made in order to see whether or not it could perform as well on the liquid market data used as labels during training in this thesis as it did on the proxy data used in [29]. However, this does not necessarily make it the best architecture for optimal performance. For example, the early convergence discussed previously could perhaps be reduced by lowering the complexity of the model. This could for instance be achieved by minimizing its depth. In this thesis's implementation, four encoders and decoders were stacked on top of each other, making the overall architecture quite deep and complex. Perhaps lowering the number would allow for a more straightforward flow of data, and thus simplifying the model's learning task. However, one should be aware that the risk taken with a decrease in complexity is that it may limit the model's ability to capture nuanced patterns necessary for accurate outputs. Moreover, jumping back to the discussion on potential issues with vanishing or exploding gradients, one approach to handle that issue could be by using more residual connections in the network to allow for a more seamless flow of the data. The outcome of such a change would be interesting to analyze. However, due to time constraints, this thesis was unable to thoroughly experiment with all

of these alternative approaches in model architecture and hyperparameter settings (keep in mind the many model instances that were run). Future projects could explore these changes and evaluate their outcomes, making it an interesting next step in this field of research.

Another point for discussion is the construction of target labels which are extremely important when training ML-models. Target labels tell the model what it is supposed to learn, directly affecting the quality of its outputs. In this thesis, it was decided early on to use the average of all liquid market spreads of similarly rated counterparties for each business date as target labels. To reiterate from Section 4.2.1 of Chapter 4, if a model was trained to predict spreads for BB rated counterparties for example, each record in the training dataset would have label columns (one per tenor) corresponding to the average spread value of all spreads of similar tenor with a BB rating for the same business date. The reason for choosing the average was that it intuitively accounts for information from all similar records into a single, uniform value reflecting the entire sample. However, this may not have been the optimal choice. Averages can be skewed by outliers, meaning that exceptionally low or high spread values on a single date could disproportionately affect the label value for that date. If this occurred multiple times, it could have caused big issues for the model when learning to predict spreads accurately. Given this, an alternative approach to constructing target labels could have been considered. For example, analyzing the distribution of spreads in the training dataset for each rating category on each business date to check for outliers might for example have suggested using the median instead of the average to get a more representative target value. This analysis was however not conducted in this thesis. Therefore, exploring other methods for producing optimal target labels could also be an interesting direction for future work on using ML-methods to generate proxy CDS spreads.

# 7
# Conclusions and Future Work

In this chapter, the conclusions of the thesis are presented. Additionally, ideas and suggestions for future work on proxy CDS spread generation are provided, particularly focusing on the use of Transformer models for this task. Finally, some of the author's final reflections on the thesis work is provided.

## 7.1 Conclusions

In this thesis a Transformer model was used to generate proxy CDS spreads which were then evaluated against actual, liquid market CDS spreads to determine accuracy. Additionally, a benchmark comparison was made against proxy CDS spreads produced by the cross-sectional Nomura model. Experiments were conducted to generate proxy spreads for five credit rating categories (AA, A, BBB, BB, and B) independently. Multiple model instances were run for each rating category, with each instance tested on a separate out-of-sample (OOS) test set for the purpose of obtaining statistically significant results. Overall, the Transformer model outperformed the Nomura model across all rating categories. Both the Transformer model and Nomura model showed a trend of decrease in accuracy as the rating grade decreased though.

Within the scope and limitations of this thesis, the outcome is deemed positive as the Transformer model has shown clear superiority over the Nomura model benchmark in producing accurate proxy CDS spreads. However, it is not deemed possible to draw general conclusions about the Transformer's superiority over traditional models based on these results alone due to this thesis's scope and limitations. Because of this, future evaluations with more varied input data, different hyperparameter settings, and potentially changes to the Transformer model's architecture are believed to be necessary. Moreover, issues related to the Transformer's complexity and lack of interpretability should be addressed and analyzed before implementing it in a production environment.

## 7.2 Future Work

Due to the limitations of this thesis project, primarily its time constraints, several aspects were left unexplored, as discussed in Chapter 6. Therefore, some suggestions for future work are presented here.

### 7.2.1 Hyperparameter Optimization

A first thing that potentially could lead to a better performance of the Transformer model in its current state is a thorough hyperparameter analysis. As has been explained, the same hyperparameters were used throughout the experiments of this thesis in order to facilitate easy comparison amongst rating categories. However, by applying a rigorous testing scheme in which different combinations of hyperparameters is tested one could perhaps find settings that provide more optimal performance in generating accurate proxy spreads. Especially interesting would be if it turned out that different hyperparameter settings provide the best results when generating spreads for different rating categories. One approach for conducting a hyperparameter analysis could be to use grid search. Essentially, grid search involves defining a parameter grid, which can be thought of as a dictionary containing different hyperparameter values to be tested, and using cross-validation to evaluate the model's performance for each parameter combination [21]. To be noted however, grid search can be a very time-consuming and computationally expensive process, especially when using large datasets and models where the number of hyperparameter combinations are many (such as in this case). This should be taken into consideration before proceeding with such a task.

### 7.2.2 Data Expansion and Diversification

Another interesting idea for future work would be to train and test the model using a broader dataset than the one used in this thesis and analyze if it enables more accurate and general predictive abilities. A first thing could be to include data from regions beyond Western Europe and sectors beyond the financial industry, which was the targeted group used here. Furthermore, simply training on a larger dataset spanning over a longer time period is also believed to further enhance the model's predictive abilities. Especially interesting would be to evaluate the model's performance in times of high market volatility, such as a market crash, to see if its predictions remains as stable as they proved to be compared to traditional models in this thesis's results even under extreme market conditions.

Moreover, expanding the dataset to include additional information beyond the sector, region, and rating variables could perhaps help the Transformer model learn even more nuanced and complex patterns, potentially leading to more accurate spread predictions. For example, incorporating information similar to the so-called option greeks could be beneficial. Without going into details, these financial metrics known as Delta, Gamma, Theta and Vega measure different dimensions of risk in options trading [35]. Since a CDS contract is a type of financial derivative similar to options, it also has sensitivities to underlying factors. By incorporating such sensitivities as features into the model, it could potentially learn even more relevant information which could perhaps enhance the accuracy of the proxy spreads it generates. However, this approach would require a much more rigorous effort in terms of data gathering and preprocessing, as well as altering the model architecture to accommodate these additional variables, than was taken in this thesis.

### 7.2.3 Computational Efficiency and Model Scalability

An additional idea for future work relates to the scalability and efficiency of the Transformer model compared to existing models that are used for generating proxy CDS spreads. While the Transformer model showed promising results in this thesis, its computational demand is quite high and if it was to be scaled up it would probably be even higher. This can perhaps become an obstacle if the Transformer is ever to be taken into production. Because of this, evaluating the trade-offs between computational costs versus performance gains is an important analysis to be made. Future work could for example explore more computationally efficient variants of the Transformer architecture than the one used here, which optimally maintains a high performance as well as reduces computational demand.

### 7.2.4 Regulatory Compliance and Model Interpretability

A final suggestion for future work has to do with interpretability and regulatory demands. As explained in Section 2.3.1, the Basel III accord requires that models that generate proxy spreads consider the sector, region, and rating variables of similar counterparties for which they are to be used for. However, due to the black-box nature of deep neural network (DNN) models discussed in Section 6.1, regulatory demands may perhaps come to place even more emphasis on the ability to explain how such models arrive at their outputs in the future. Because of this, looking into explainable AI frameworks could be a way to help explain the model's decision-making process when determining a spread and thereby make its internal workings more interpretable. For example, this can include things such as feature importance analysis, model simplification, or other methods. Not only can this help in complying with potential future regulatory demands, but it is also important from a developers point of view to be able to pinpoint how the model that is used maps input to output to the greatest extent possible. This is also important from an ethical point of view, since an understanding of how the model processes the data passed to it can help in highlighting if any biases with respect to the data may occur in its decision-making. What this essentially boils down to is that future work on model interpretability can serve many additional purposes beyond just getting a really good model up and running. It ensures that ethics and regulatory compliance is considered alongside development and such an approach, built on trust and transparency, paves the way for a sustainable way of working with new technologies.

## 7.3 Final Reflections

This thesis has combined finance, machine learning (ML), and statistics to evaluate a new method (the Transformer model) for generating proxy CDS spreads. While the work has been very interesting for the author, particularly on a personal level in learning about industry approaches to advanced ML-modeling, data preprocessing, and statistical evaluations, understanding the thesis's broader implications and its place in a larger context has been the most rewarding aspect of this work. Particularly regarding the impact that proxy CDS spreads have on accurately assessing

counterparty credit risk (CCR), which is of high importance for financial institutions to get right. Much is left for future work on this topic, but hopefully this thesis has contributed with valuable insights regarding the potentials that the Transformer model offers for the task of proxy CDS spread generation compared to traditional alternatives.

# Bibliography


[1] Keras documentation: Embedding layer. `https://keras.io/api/layers/core_layers/embedding/`. Accessed: 2024-04-20.

[2] Matplotlib - visualization with python. `https://matplotlib.org`. Accessed: 2024-04-20.

[3] Numpy - scientific computing with python. `https://numpy.org`. Accessed: 2024-04-20.

[4] Pandas - python data analysis library. `https://pandas.pydata.org`. Accessed: 2024-04-20.

[5] Python programming language - official website. `https://www.python.org`. Accessed: 2024-04-20.

[6] Scipy - fundamental algorithms for scientific computing in python. `https://scipy.org/`. Accessed: 2024-05-19.

[7] N. Acharya. Choosing between mean squared error (mse) and mean absolute error (mae) in regression: A deep dive. Medium, August 2023. Accessed: 2024-05-19.

[8] S. Ahmed, I. Nielsen, A. Tripathi, S. Siddiqui, G. Rasool, and R. Ramachandran. Transformers in time-series analysis: A tutorial. arxiv 2022. *arXiv preprint arXiv:2205.01138*, 2022.

[9] D. Amballa. Introduction to ai, ml, dl. *Towards Data Science*, Sep 2023.

[10] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[11] Basel Committee on Banking Supervision. Minimum capital requirements for market risk. Technical report, Bank for International Settlements, January 2019. Revised February 2019.

[12] R. Brummelhuis and Z. Luo. Arbitrage opportunities in cds term structure: theory and implications for otc derivatives. *arXiv preprint arXiv:1811.08038*, 2018.

[13] R. Brummelhuis and Z. Luo. Bank net interest margin forecasting and capital adequacy stress testing by machine learning techniques. *Available at SSRN 3282408*, 2019.

[14] R. Brummelhuis and Z. Luo. *CDS proxy construction via machine learning techniques part I: methodology and results.* SSRN, 2019.

[15] R. Brummelhuis and Z. Luo. *CDS proxy construction via machine learning techniques part II: parametrization, correlation, benchmarking.* SSRN, 2019.

[16] Economics Observatory. Why did lehman brothers fail? `https://www.economicsobservatory.com/why-did-lehman-brothers-fail`, 2023. Accessed: 2024-02-12.

[17] European Banking Authority. Eba final draft regulatory technical standards on credit valuation adjustment risk for the determination of a proxy spread and the specification of a limited number of smaller portfolios. Regulatory Technical Standards EBA/RTS/2013/17, European Banking Authority, Dec 2013.
[18] L. Fageräng and H. Thoursie. Modelling proxy credit cruves using recurrent neural networks, 2023.
[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
[20] J. Gregory. *The XVA challenge: Counterparty credit risk, funding, collateral, and capital*. John Wiley & Sons, Incorporated, 3rd edition, 2015.
[21] H. Hassan. Using grid search for hyper-parameter tuning. *Medium*, October 2023.
[22] A. Hayes. What is a credit default swap and how does it work? *Investopedia*, 2024.
[23] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications*, page 122666, 2023.
[24] S. Jayawardhana. Sequence models & recurrent neural networks (rnns). *Towards Data Science*, Jul 2020.
[25] W. Kenton. What is a black box model? definition, uses, and examples. `https://www.investopedia.com/terms/b/blackbox.asp`, April 2024. Accessed: 2024-05-17.
[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[27] Laerd Statistics. Spearman's rank-order correlation, 2024. Accessed: 2024-05-19.
[28] B. Liquet, S. Moka, and Y. Nazarathy. The mathematical engineering of deep learning, 2024.
[29] J. Luhr. Modeling credit default swap spreads with transformers: A thesis in collaboration with handelsbanken, 2023.
[30] B. Mehlig. *Machine learning with neural networks: An introduction for scientists and engineers*. Cambridge University Press, 2021.
[31] Nomura International plc. Cross-sectional method for cva calculation. Nomura Research, 2013. Accessed: 2024-01-24.
[32] C. Pere. What are loss functions? *Towards Data Science*, Jun 2020.
[33] O. F. Rokon. Rnn vs. lstm vs. transformers: Unraveling the secrets of sequential data processing. *Medium*, Sep 2023.
[34] K. Stewart. Mean squared error. Encyclopaedia Britannica, 2024. Accessed: 2024-05-19.
[35] J. Summa. Option greeks: The 4 factors to measure risk. *Investopedia*, April 2024.
[36] TensorFlow Team. Keras: A guide to the Keras API in TensorFlow. `https://www.tensorflow.org/guide/keras`, 2023. Accessed: 2024-02-12.
[37] TensorFlow Team. TensorFlow: An end-to-end open source machine learning platform. `https://www.tensorflow.org/`, 2023. Accessed: 2024-02-12.

[38] The AI Hacker. Illustrated guide to transformers neural network: A step by step explanation. YouTube, 2020. Available at: https://www.youtube.com/watch?v=4Bdc55j80l8.

[39] The Editors of Encyclopaedia Britannica. Hypothetico-deductive method. https://www.britannica.com/science/hypothetico-deductive-method, March 2020. Accessed: 2024-01-24.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[41] H. Yadav. Dropout in neural networks. *Towards Data Science*, Jul 2022.

**DEPARTMENT OF PHYSICS**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY