



RESEARCH ARTICLE

Multiple yield curve modeling and forecasting using deep learning

Ronald Richman¹  and Salvatore Scognamiglio² 

¹Old Mutual Insure and University of the Witwatersrand, Johannesburg, South Africa

²Department of Management and Quantitative Studies, University of Naples “Parthenope”, Naples, Italy

Corresponding author: Ronald Richman; Email: ronaldrichman@gmail.com

Received: 29 February 2024; **Revised:** 30 June 2024; **Accepted:** 10 August 2024; **First published online:** 4 October 2024

Keywords: Deep learning; multiple yield curve modeling; Nelson–Siegel model; attention models; transfer learning; interest rate risk; value-at-risk; asset–liability management; Solvency II, IFRS 17; Real-world modeling

Abstract

This manuscript introduces deep learning models that simultaneously describe the dynamics of several yield curves. We aim to learn the dependence structure among the different yield curves induced by the globalization of financial markets and exploit it to produce more accurate forecasts. By combining the self-attention mechanism and nonparametric quantile regression, our model generates both point and interval forecasts of future yields. The architecture is designed to avoid quantile crossing issues affecting multiple quantile regression models. Numerical experiments conducted on two different datasets confirm the effectiveness of our approach. Finally, we explore potential extensions and enhancements by incorporating deep ensemble methods and transfer learning mechanisms.

1. Introduction

Yield curves are used for a wide variety of tasks in actuarial science and finance for deriving the present value of future cashflows within valuations that apply a market-consistent approach. A market-consistent approach to liability valuation is required by modern solvency regulations, such as Solvency II, while updated accounting standards, such as the recently introduced IFRS 17, require the use of credit and liquidity-adjusted yield curves for discounting liabilities, including both life and nonlife insurance liabilities. Modern actuarial valuation practices for insurance-related liabilities can be categorized according to the type of underlying liability that needs to be valued; these are liabilities that result only from insurance risk and liabilities that result from a combination of both insurance risk and market risk. Expected claims payments, which comprise the majority of the liabilities entered into by nonlife insurers and a significant portion of life insurance liabilities, are an example of the former type of liabilities, whereas the valuation of options and guarantees embedded into life insurance contracts, such as guaranteed benefits linked to market values in variable annuity contracts, are an example of the latter type of liabilities. Importantly, insurance risk-related liabilities cannot usually be hedged in financial markets, whereas market-linked insurance benefits often can be hedged with financial instruments. For the former type, in both Solvency II and IFRS 17, liability valuation is usually performed by, first, projecting the real-world expected cashflows resulting from insurance contracts, and, second, discounting these cashflows using the discount rates implied by the yield curve, since the yield curve replicates the market-consistent valuation of these cashflows, see Europeen (2012) for a practice-orientated overview and Kemp (2009) for a detailed review. Finally, since the projection of these liabilities is done under the real-world probability measure \mathbb{P} , that is not a risk-neutral valuation, a last step is to add a risk margin to the liability value to recover an approximate market-consistent value. For liabilities with a direct link to market variables, on

the other hand, the actuarial valuation is performed in a slightly more complex manner, first, by valuing the expected cashflows under a risk-neutral measure \mathbb{Q} , then, by discounting the expected value of these back to the valuation date by using the discount rates from the yield curve. Beyond valuation of liabilities, insurance products with long contract boundaries, for example, those contracts which operate over a periods more than a single year, will also usually be priced on a discounted basis. Thus, since the major liability line item on insurer's balance sheets depends on the yield curve, having good forecasts of the yield curve is important for both life and nonlife insurers. Moreover, insurers, and other entities, that report on these discounted liabilities are exposed to the risk of changes in the interest rates in their markets, which translate directly into changes in the solvency of these entities. Therefore, managing this risk of adverse changes in yield curves – which we refer to as interest rate risk in what follows - is an important task within actuarial work, which is usually considered in the context of corresponding changes in the asset portfolio backing these liabilities, changes in the value of which may act as an offset. This process is, therefore, usually referred to as asset–liability management (ALM). Moreover, insurers are required to hold capital to ensure that their solvency is adequately protected in most solvency regimes, such as Solvency II. To measure the extent of the interest rate risk, as well as the corresponding capital needed to be held, insurers and other financial institutions often rely on modeling the uncertain future evolution of the yield curve using a variety of different models. Once the yield curves have been modeled, the models are used to derive scenarios for the future evolution of the yield curves, which are then applied to derive capital requirements, which are referred to as the Solvency Capital Requirements (SCR) in Solvency II. Notably, to derive the SCR, potential changes in the yield curve need to be assessed using real-world probabilities, in other words, under the \mathbb{P} measure. For non-hedgeable insurance-linked liabilities, the risk arising from changes in the yield curve is derived by discounting these using both the original yield curve and a yield curve derived in the scenario of interest, which is the 1-in-200 year downside scenario in a Solvency II context. The difference between these quantities is taken as the SCR requirement for interest rate risk. For hedgeable liabilities containing benefits linked to market variables, the process is more complex, often performed in practice using a nested Monte-Carlo valuation approach in which the yield curve is first projected under the real-world measure \mathbb{P} and then, conditional on this projection, a risk-neutral valuation of the cashflows is performed under the risk-neutral measure \mathbb{Q} . The projection step is often called the “outer step” of the valuation procedure and the risk-neutral step is often called the “inner step.” In this work, we focus on the projection of the yield curve under the real-world measure, which is useful both for valuation and SCR calculations of non-hedgeable insurance-linked liabilities, as well being useful for performing the outer step of nested Monte-Carlo valuation. Moreover, other components of Solvency II and quantitative risk management within insurers require real-world stresses and scenarios for interest rate risk management, for example, the Own risk and Solvency Assessment (ORSA). For this purpose of projecting the yield curve under the real-world measure \mathbb{P} , we distinguish between unconditional and conditional approaches to yield curve modeling: the former approach calibrates models of the yield curve evolution using historical data at a point in time, derives stresses based on these, and then applies these stresses without recalibrating these based on current market conditions. This approach underlies, for example, the standard formula approach of the Solvency II regulation. On the other hand, the conditional approach uses current market information to recalibrate yield curve stresses; this approach is often taken in internal model approaches within the Solvency II regulation.

Modeling interest rate risks is made more difficult due to the complexity of requirements of recent accounting standards, as well as the interconnected nature of financial markets across asset classes and geographies. The recent IFRS 17 standard departs from a purely market-consistent valuation approach by requiring insurers to use yield curves that are modified to correspond to the financial characteristics of the liabilities being valued, as well as the asset portfolios backing these. While we do not explain this in detail, in short, insurers must derive yield curves consisting of the (credit) risk-free interest rate, as well as an allowance for an illiquidity premium. Thus, insurers reporting under IFRS 17 calibrate several different yield curves for discounting liabilities, the evolution of which will differ depending on both how risk-free rates and illiquidity premia change over time. Another reason that insurers may need

to model the evolution of multiple yield curves is due to their investing in assets with different levels of credit quality; to manage the risk of the asset portfolio, it is often necessary to calibrate multiple yield curves which take account of credit-risk premia and model the (joint) evolution of these. Finally, insurers operating in multiple geographic jurisdictions need to manage the interest rate risk arising from changes in the different yield curves used in these markets. In all of these scenarios, it is not sufficient merely to model the dynamics of each yield curve on an independent basis, since this approach will not capture the correlation between asset classes and geographies and may lead to misstated estimates of risk and capital; rather the joint future evolution of the complete set of yield curves used by the insurer must be modeled.

In this work, we focus on exactly this problem of jointly modeling and forecasting multiple yield curves for interest rate risk management, ALM, and derivation of capital requirements; we note that this is done on the real-world basis and not the risk-neutral basis which is useful for option valuation. Our research considers a concept of “multiple yield curves” that differs from the standard literature. While conventional literature refers to curves with varying structures by maturity, also known as tenor-dependent yield curves (Grbac and Runggaldier, 2015), we consider families of curves defined on the same maturity structure but differing among them in credit quality or because they are associated with different government bonds. For this task, we use neural network models trained jointly on a significant amount of historical yield curve data across geographies to forecast yield curves on an expected (best-estimate) basis, as well as to forecast the quantiles of the yield curves; the latter can be used directly for risk management purposes, for example, calculating the value-at-risk of the insurer.

1.1. Literature review

Several different approaches have appeared in the literature to model the uncertain future evolution of the yield curve (Venter, 2004; Teichmann and Wüthrich, 2016).

One class of models – focused on the risk-neutral evolution of the yield curve – consists of arbitrage-free models, which impose restrictions on the evolution of the yield curve to avoid risk-free profit opportunities. Prominent examples are the models developed in Hull and White (1990) and Heath et al. (1992). Although these models are widely used in option pricing, without further adaptation, they are often found to forecast poorly compared with a simple random walk model (see Duffee, 2002). We refer to Yasuoka (2018), and the citations therein, for interesting discussions of adapting risk-neutral interest rate models for real-world purposes by estimating the market price of interest rate risk.

A well-known technique is to apply principal components analysis (PCA) to vectors of the changes in the yield curve at each term and then to use the simulated changes in the yield curve to derive a distribution of yield curves (for an overview, see Redfern and McLean, 2014). This approach was used, for example, to calibrate the yield curve stresses in the interest rate risk module of Solvency II. This is an example of the unconditional approach to yield curve modeling, since the PCA analysis was performed at a point in time in the past, and is assumed to still be relevant in current market conditions. Other authors follow a purely statistical approach. This class of models has evolved from univariate (Fama and Bliss, 1987) to multivariate time series models and recent advances in dynamic factor models. Among them, the dynamic extension of the well-known Nelson–Siegel (NS) model (Nelson and Siegel, 1987) proposed in Diebold and Li (2006) (from now on referred to as DNS) has become very popular among practitioners thanks to its simplicity and discrete forecasting ability. In addition to being a relatively simple and parsimonious model, the DNS approach is also appealing since the NS model (and its extensions) underlying the DNS approach is often used by central banks and other institutions for calibrating the yield curve. Other notable examples of factor models can be found in Bowsher and Meeks (2008), Härdle and Majer (2016).

Numerous extensions of the DNS approach have been developed in the literature. Some research investigated using more flexible versions of the NS model, for example, the model proposed in Bliss (1996), the four-factor extension suggested in Svensson (1994), and the five-factor model investigated

in De Rezende and Ferreira (2013). Other authors try to improve the forecasting performance of the DNS model, including some macroeconomics variables in the models (see Diebold et al., 2006). A nice overview of the NS and DNS models is in Diebold and Rudebusch (2013), who also provide economic intuitions for the factors used in the NS model.

1.1.1. Multiple yield curve modeling

Globalization has intensified the connection among the financial markets, inducing a dependence structure among different yield curves, which renders the process of modeling these jointly complex; moreover, above we have discussed other reasons for the need for joint modeling of multiple yield curves, which is a challenging task. Despite the relevance of the topic for financial markets, there is relatively little discussion of this topic in the literature on a real-world basis; on the other hand, more literature is available in the risk-neutral setting, see Cuchiero et al. (2016) for an overview. Within the real-work setting, here we mention Gerhart and Lütkebohmert (2020), who introduced a multiple-curves PCA method where the dynamics of multiple yield curves captured through principal component analysis (PCA) are modeled as autoregressive processes and Atkins and Cummins (2023), who proposed a two-step method to jointly capture the risk-factor relationships within each curve and the risk-factor relationships between the curves. In the first stage, the authors use the PCA to derive components describing the dynamics of each curve, and then, secondly, combine these to describe the dynamics across all the curves. Notably, the joint forecasting of best-estimates and quantiles is not done in these works, and this is a novel aspect of the model presented here.

1.1.2. Neural networks for yield curve modeling

Recently, deep learning models have become popular for general machine learning (ML) tasks, due to their ability to model massive volumes of data in a flexible manner, and within finance and actuarial science. Deep neural networks have been successfully applied to several tasks such as pricing (Noll et al., 2020; Barigou and Delong, 2022), reserving (Gabrielli, 2020), and mortality forecasting (Perla et al., 2021; Scognamiglio, 2022). A detailed overview of the application of artificial intelligence (AI) and ML techniques in actuarial science can be found in Richman (2021a,b). Focusing on yield curves modeling and interest rate risk management, the literature is relatively sparse. Aljinović and Poklepović (2013) shows that feed-forward NN can be used to replace time series models to extrapolate the future values of the NS parameters. Kauffmann et al. (2022) proposes to improve the flexibility of the DNS model using NN for deriving the factor loadings. Nunes et al. (2019) directly employ feed-forward NN to forecast future yields, and Gerhart et al. (2019) use modern recurrent neural networks (RNNs) such as the long short-term memory (LSTM) which are specifically designed to analyze sequential data, such as the time series of the yields.

1.1.3. Contributions

In this work, we develop deep learning models that simultaneously model and forecast the dynamics of the multiple yield curves, which could be related to different countries, credit qualities, or liquidity characteristics; here, we focus on the first two of these. With their ability to describe high-dimensional time series data and model the nonlinearity often present in the data, deep learning techniques are promising tools for multiple yield curve modeling and forecasting. The idea is to exploit the dependence structure among the different yield curves induced by the globalization of financial markets or relationships between asset classes to improve the forecasting performance of our models by jointly training these on historical datasets of yield curves. Importantly, we focus both on producing best-estimate forecasts of the yield curve, as we as employ deep learning techniques to quantify the uncertainty around the predictions by forecasting quantiles. Although this latter aspect is relevant both from a practical risk management

and a theoretical point of view, it has not been deeply investigated in the literature, and creating joint forecasts of these is, to our knowledge, novel. For uncertainty quantification we use modern deep learning approaches, such as nonparametric quantile regression and deep ensembles model (Lakshminarayanan et al., 2017). This augmentation of best-estimate forecast models with forecast quantiles is particularly useful for risk management purposes since the quantiles correspond to the VaR, which, in practice, underlies many quantitative risk management systems in practice. A comparison among these methods for measuring the uncertainty in the forecasts is also of interest, since it could provide additional insights into how well these methods describe the evolution of the yield curves.

We utilize recent advances in deep learning methodology into our selected yield curve model, specifically, the self-attention mechanism, which has been used to great success in natural language processing (Vaswani et al., 2017) and has recently been applied for severity modeling of flood insurance claims (Kuo and Richman, 2021). Here, we show how the features derived using a convolutional neural network (CNN) can be enhanced using the self-attention mechanism for greater forecasting accuracy.

Finally, we investigate the use of transfer learning methods, which aim to transfer as much knowledge as possible from an existing model to a new model designed for a similar task. These methods are already intensively used in computer vision and natural language processing tasks, where models are trained on large general datasets and then fine-tuned on more specific tasks. In our context, the transfer learning mechanism is applied to exploit knowledge learned by parameterizing models on a database of yield curves from multiple jurisdictions and then transferring the learned model to a smaller dataset of yield curves derived for assets of various credit quality.

Organization of the manuscript: The rest of the manuscript is structured as follows. Section 2 introduces two of the most popular factor models for yield curve modeling and forecasting, Section 3 describes neural network building blocks used in Section 4, where we present the proposed yield curve model. Section 5 illustrates some numerical experiments on a large cross-geography dataset of yield curves, Section 6 discusses some possible ways to extend and enhance the proposed model, and Section 7 concludes.

2. Dynamic Nelson–Siegel and Nelson–Siegel–Svensson models

We consider a scenario where the objective is to model the dynamics of yield curves belonging to different families. These curve families might pertain to yield curves that vary in credit rating quality; for example, families could be labeled as ‘A’, ‘AA’, ‘AAA’, and so forth. Alternatively, the yield curves could be associated with government bonds from different countries, with families labeled as ‘Euro’, ‘United Kingdom’, and so on.

Let \mathcal{I} represent the set of considered curve families, \mathcal{M} denote the set of time-to-maturities for which the curve is defined, \mathcal{T} denote a set of dates, and $y_t^{(i)}(\tau)$ denote the continuously compounded zero-coupon nominal yield at time $t \in \mathcal{T}$ on a τ -month bond (i.e., at tenor $\tau \in \mathcal{M}$) for the i -th bond in the set \mathcal{I} . Importantly, we note that here we work with spot rates, whereas, for example, PCA analysis of yield curves is often performed on forward rates. Here, we focus on describing more traditional models which will be used as a benchmark for the neural network models introduced later.

In their influential work, Nelson and Siegel (NS) (Nelson and Siegel, 1987) introduce a three-factor model that, at a given date t , describes the relationship between the yield and maturity τ . Given that the classical NS model is static, Diebold and Li (2006) introduces a dynamic version where the model’s parameters can vary over time. In this case, $y_t^{(i)}(\tau)$ can be expressed as follows:

$$y_t^{(i)}(\tau) = \beta_{0,t}^{(i)} + \beta_{1,t}^{(i)} \left(\frac{1 - e^{-\lambda_t^{(i)} \tau}}{\lambda_t^{(i)} \tau} \right) + \beta_{2,t}^{(i)} \left(\frac{1 - e^{-\lambda_t^{(i)} \tau}}{\lambda_t^{(i)} \tau} - e^{-\lambda_t^{(i)} \tau} \right) + \epsilon_t^{(i)}(\tau), \quad \epsilon_t^{(i)}(\tau) \sim \mathcal{N}(0, \sigma_{\epsilon_t^{(i)}}^2),$$

where $\beta_{0,t}^{(i)}, \beta_{1,t}^{(i)}, \beta_{2,t}^{(i)}, \lambda_t^{(i)} \in \mathbb{R}$ are model parameters governing the shape of the curve that are estimated for each date t and each family curve i by using market data. More specifically, the parameters $\beta_{j,t}^{(i)}$ with

$j \in \{0, 1, 2\}$ can be interpreted as three latent factors defining the level, slope, and curvature of the yield curve, respectively, while $\lambda_i^{(i)}$ indicates where the loading achieves its maximum; here, we follow the interpretation of these factors given in Diebold and Li (2006) Different calibration procedures for the dynamic NS model exist in the literature. In the following sections, we adopt the methodology suggested in the original paper (Diebold and Li, 2006) For further details, please refer to Appendix A.1.

To make forecasts, a dynamic model for the latent factors $\beta_{j,t}^{(i)}$ with $j \in \{0, 1, 2\}$ has to be specified. The simplest choice consists of using a set of individual first-order autoregressive (AR) models:

$$\beta_{j,t}^{(i)} = \psi_{0j}^{(i)} + \psi_{1j}^{(i)}\beta_{j,t-1}^{(i)} + \zeta_{j,t}^{(i)}, \quad \zeta_{j,t}^{(i)} \sim \mathcal{N}(0, \sigma_{\zeta_j}^2) \tag{2.1}$$

where $\psi_{0j}^{(i)}, \psi_{1j}^{(i)} \in \mathbb{R}, i \in \mathcal{I}, j \in \{0, 1, 2\}$ are the time series model parameters and $\zeta_{j,t}^{(i)}$ are normally distributed error terms. Alternatively, one could also model the vector $\beta_t^{(i)} = (\beta_{0,t}^{(i)}, \beta_{1,t}^{(i)}, \beta_{2,t}^{(i)}) \in \mathbb{R}^3$ using single first-order multivariate vector autoregressive (VAR) model:

$$\beta_t^{(i)} = \mathbf{a}_0^{(i)} + A^{(i)}\beta_{t-1}^{(i)} + \boldsymbol{\eta}_t^{(i)}, \quad \boldsymbol{\eta}_t^{(i)} \sim \mathcal{N}(0, E^{(i)}) \tag{2.2}$$

with $\mathbf{a}_0^{(i)} \in \mathbb{R}^3, A^{(i)} \in \mathbb{R}^{3 \times 3}$, and $\boldsymbol{\eta}_t^{(i)} \sim \mathcal{N}(0, E^{(i)})$ is the normally distributed error term with matrix $E^{(i)} \in \mathbb{R}^{3 \times 3}$.

Numerous extensions of the NS model and its dynamic version have been proposed in the literature. One of the most popular enhancements, due to Svensson (Svensson, 1994), introduces an additional term to augment flexibility. This extension enhances the model’s capacity to capture various shapes of yield curves by incorporating additional curvature components. The Svensson extension allows for a more general representation of the term structure of interest rates, establishing it as a valuable and often used tool in fixed-income and financial modeling. The Nelson–Siegel–Svensson (NSS) model is defined as follows:

$$y_t^{(i)}(\tau) = \beta_{0,t}^{(i)} + \beta_{1,t}^{(i)}\left(\frac{1 - e^{-\lambda_{1,t}^{(i)}\tau}}{\lambda_{1,t}^{(i)}\tau}\right) + \beta_{2,t}^{(i)}\left(\frac{1 - e^{-\lambda_{1,t}^{(i)}\tau}}{\lambda_{1,t}^{(i)}\tau} - e^{-\lambda_{1,t}^{(i)}\tau}\right) + \beta_{3,t}^{(i)}\left(\frac{1 - e^{-\lambda_{2,t}^{(i)}\tau}}{\lambda_{2,t}^{(i)}\tau} - e^{-\lambda_{2,t}^{(i)}\tau}\right) + \epsilon_t^{(i)}(\tau),$$

where $\beta_{3,t}^{(i)} \in \mathbb{R}$ is a second curvature parameter and $\lambda_{1,t}^{(i)}, \lambda_{2,t}^{(i)} \in \mathbb{R}$ are two decay factors. Also for the NSS extension, we consider the calibration scheme adopted in Diebold and Li (2006). In the upcoming sections, we will present comparisons involving both the NS model and its NSS extension. We will use the notation NS_AR (NSS_AR) to refer to cases where the latent factors of the NS models are described as independent AR(1) models and NS_VAR (NSS_VAR) to refer to cases where a single multivariate VAR(1) model is utilized.

3. Neural Networks

Neural networks (NNs) represent nonlinear statistical models originally inspired by the functioning of the human brain, as implied by their name, and subsequently extensively developed for multiple applications in ML, see Goodfellow et al. (2016) for a review. A feed-forward neural network comprises interconnected computational units, or neurons, organized in multiple layers. These neurons “learn” from data through training algorithms. The fundamental concept involves mapping input data to a new multidimensional space, extracting derived features. The output (target) is then modeled as a nonlinear function of these derived features; this process is called representation learning (Bengio et al., 2013). Implementing multiple feed-forward network layers is called deep learning in the literature and has proved to be particularly promising when dealing with high-dimensional problems requiring the identification of nonlinear dependencies. The arrangement of connections among the units delineates various types of neural networks. Our neural network model is constructed on the principles of both feed-forward and RNNs. An overview of the neural network blocks employed in the best-performing model presented in this paper is provided below, whereas we summarize briefly the network blocks that are used in models that perform less well. For more detail on neural networks in an actuarial context, we refer to Wüthrich and Merz (2021), whose notation we follow.

3.1. Fully connected layer

A fully connected network (FCN) layer, also commonly known as a dense layer due to the dense connections between units, is a type of layer in a neural network where each neuron or unit is connected to every neuron in the previous layer. In other words, each neuron in a fully connected layer receives input from all the neurons in the preceding layer; if the layer is the first in a network, then each neuron receives inputs from all of the covariates input into the network.

Let $\mathbf{x} \in \mathbb{R}^{q_0}$ be the input vector; a FCN layer with $q_1 \in \mathbb{N}$ units is a vector function that maps \mathbf{x} into a q_1 -dimensional real-valued space:

$$\mathbf{z}^{(1)} : \mathbb{R}^{q_0} \rightarrow \mathbb{R}^{q_1}, \quad \mathbf{x} \mapsto \mathbf{z}^{(1)}(\mathbf{x}) = (z_1^{(1)}(\mathbf{x}), z_2^{(1)}(\mathbf{x}), \dots, z_{q_1}^{(1)}(\mathbf{x}))'$$

The output of each unit is a new feature $z_j^{(1)}(\mathbf{x})$, which is a nonlinear function of \mathbf{x} :

$$z_j^{(1)}(\mathbf{x}) = \phi^{(1)}\left(w_{j,0}^{(1)} + \sum_{l=1}^{q_1} w_{j,l}^{(1)}x_l\right) \quad j = 1, 2, \dots, q_1,$$

where $\phi^{(1)} : \mathbb{R} \mapsto \mathbb{R}$ is the activation function and $w_{j,l}^{(1)} \in \mathbb{R}$ represent the weights. In matrix form, the output $\mathbf{z}^{(1)}(\mathbf{x})$ of the FCN layer can be written as:

$$\mathbf{z}^{(1)}(\mathbf{x}) = \phi^{(1)}(\mathbf{w}_0^{(1)} + W^{(1)}\mathbf{x}). \tag{3.1}$$

Shallow neural networks are those networks with a single dense layer and directly use the features derived in the layer for computing the (output) quantity of interest $y \in \mathcal{Y}$. In the case of $\mathcal{Y} \subseteq \mathbb{R}$, the output of shallow NN reads

$$y = \phi^{(o)}\left(w_0^{(o)} + \langle \mathbf{w}^{(o)}, \mathbf{z}^{(1)}(\mathbf{x}) \rangle\right),$$

where $w_0^{(o)} \in \mathbb{R}$, $\mathbf{w}^{(o)} \in \mathbb{R}^{q_1}$, $\phi^{(o)} : \mathbb{R} \mapsto \mathbb{R}$, $\langle \cdot, \cdot \rangle$ denotes the scalar product in \mathbb{R}^{q_1} .

If, on the other hand, the network is deep, the vector $\mathbf{z}^{(1)}(\mathbf{x})$ is used as input in the next layer for computing new features and so on for the following layers. Let $h \in \mathbb{N}$ be the number of hidden layers (depth of network), and $q_k \in \mathbb{N}$, for $1 \leq k \leq h$, be a sequence of integers that indicates the dimension of each FCN layer (widths of layers). A deep FCN can be described as follows:

$$\mathbf{x} \mapsto \mathbf{z}^{(h:1)}(\mathbf{x}) = (\mathbf{z}^{(h)} \circ \dots \circ \mathbf{z}^{(1)})(\mathbf{x}) \in \mathbb{R}^{q_h},$$

where the vector functions $\mathbf{z}^{(k)} : \mathbb{R}^{q_{k-1}} \rightarrow \mathbb{R}^{q_k}$ have the same structure of (3.1), and $W^{(k)} = (\mathbf{w}_j^{(k)})_{1 \leq j \leq q_k} \in \mathbb{R}^{q_k \times q_{k-1}}$, $\mathbf{w}_0^{(k)} \in \mathbb{R}^{q_k}$, for $1 \leq k \leq h$ are the network weights. In the case of deep NN, the output layer uses the features extracted by the last hidden layer $\mathbf{z}^{(h:1)}(\mathbf{x})$ instead of those $\mathbf{z}^{(1)}(\mathbf{x})$. FCN layers are used to process numerical features or the outputs from other layers, such as recurrent, embedding, or attention layers.

Finally, we mention that if the inputs to the FCN have a sequential structure, one way of processing these is to apply the same FCN to each input in turn, producing learned features for each entry in the sequence. This is called a point-wise neural network in Vaswani et al. (2017) and a time-distributed network in the Keras library.

3.2. Embedding layer

An embedding layer is designed to acquire a low-dimensional representation of categorical variable levels. Let $q_{\mathcal{L}} \in \mathbb{N}$ denote the hyperparameter determining the size of the embedding. The categorical variable levels are transformed into a real-valued $q_{\mathcal{L}}$ -dimensional space, and the coordinates of each level in this new space serve as learned parameters of the neural network, requiring training; see Guo and Berkahn (2016) who introduced this technique for deep learning models.

The distances between levels in this learned space reflect the similarity of levels concerning the target variable: closely related levels exhibit small Euclidean distances, while significantly different categories display larger distances.

Formally, let $l_1, l_2, \dots, l_{n_{\mathcal{L}}}$ represent the set of categories for the qualitative variable. The embedding layer functions as a mapping:

$$e : \{l_1, \dots, l_{n_{\mathcal{L}}}\} \mapsto \mathbb{R}^{q_{\mathcal{L}}}, \quad l_k \mapsto e(l_k) \stackrel{\text{set}}{=} e^{(k)}.$$

The total number of embedding weights to be learned during training is $n_{\mathcal{L}}q_{\mathcal{L}}$. Generally, embedding layers are applied to the original categorical covariates to learn a numerical representation of the levels. These representations are then passed to subsequent layers, which can be of various types, that utilize the information from these vectors to compute the quantity of interest.

3.3. Attention layer

An attention layer is a component in neural network architectures that implements an attention mechanism, which, in some sense “focuses” on certain aspects of the input data that are deemed to be relevant for the problem at hand. These mechanisms allow the flexibility and performance of models to be enhanced and have produced excellent results, especially in tasks involving sequences like natural language processing, as well as within actuarial tasks, see, for example, Kuo and Richman (2021). The main idea of attention mechanisms is their ability to enable deep neural networks to reweight the significance of input data entering the model dynamically.

Several attention mechanisms have been proposed in the literature. We focus on the most popular form of attention, which is the *scaled dot-product* attention proposed in Vaswani et al. (2017) for as a component of the Transformer model proposed there.

Let $Q \in \mathbb{R}^{q \times d}$ be a matrix of query vectors, $K \in \mathbb{R}^{q \times d}$ be a matrix of key vectors, $V \in \mathbb{R}^{q \times d}$ is a matrix of value vectors. The scaled dot-product attention mechanism is a mapping:

$$A : \mathbb{R}^{(q \times d) \times (q \times d) \times (q \times d)} \rightarrow \mathbb{R}^{(q \times d)}, \quad (Q, K, V) \mapsto A = \text{attn}(Q, K, V).$$

The attention mechanism is applied to the matrix V , and the resulting output is calculated as a weighted sum of its elements. These attention coefficients depend on the matrices Q and K . They undergo scalar-dot multiplication first, followed by the application of the softmax function to normalize the scores. Formally, the attention mapping has the following structure:

$$A = \text{softmax}(B)V = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where $d \in [0, +\infty)$ is a scalar coefficient, and the matrix of the scores B^* is derived from the matrix B :

$$B^* = \text{softmax}(B) \quad \text{where} \quad b_{i,j}^* = \frac{\exp(b_{i,j})}{\sum_{k=1}^q \exp(b_{i,k})} \in (0, 1).$$

d can be set equal to the dimension of the value vectors in the attention calculation or can alternatively be a learned parameter. The scalar dot attention mechanism is computationally efficient, mainly because it does not require recursive computation and is thus easily implemented on graphics processing units (GPU), and has been widely adopted in Transformer-based models due to its simplicity and effectiveness in capturing relationships between elements in a sequence.

We provide a simple intuition for the learned attention scores in B^* , which are multiplied by the value vectors in V . Each row of the new matrix B^*V is comprised of a weighted average of the vectors in V , where the weights – adding to unity – determine the “importance” of each row vector i of V . Attention layers are typically used for processing numerical input. They can be applied directly to original data in numerical form or to the learned representations generated by embedding layers for categorical features. The output of the attention layer can then be further processed by various types of layers, including FCN layers.

3.4. Other network layers

Here, we briefly mention other neural network layers that were tested when designing the attention network that is the main result presented in Section 4.

The network components presented to this point process inputs to the network without any reference to previous inputs to the network. RNNs modify neural networks to maintain an internal state that is calibrated from previous inputs of the network. An example of a RNN is the LSTM network introduced in Hochreiter and Schmidhuber (1997). The key innovation of LSTM networks lies in their architecture, which incorporates specialized memory cells capable of selectively retaining or discarding information over time. This is achieved through a set of gating mechanisms that regulate the flow of information within the network. Specifically, LSTM units contain three gates: the input gate, the forget gate, and the output gate. The input gate determines which information from the current input should be stored in the memory cell. The forget gate decides which information from the previous internal state should be discarded. The output gate controls the information that will be output the current memory cell. By dynamically adjusting the operations of these gates based on the input data and the network's internal state, LSTMs can effectively learn and remember patterns across varying timescales in sequential data. This capability makes them particularly well suited to effectively capture long-term dependencies in sequential data.

CNNs differ from FCNs by connecting the units of this network only to a small patch of the inputs, whereas the units in FCNs are connected to all of the inputs. They are especially promising when processing spatial data, such as images or sequences. CNN layers employ a sliding window called a kernel or filter, which moves across the input data, scanning small patches at a time. This localized connectivity allows CNNs to effectively capture spatial hierarchies and patterns present in the input data. Moreover, the same set of learnable weights is applied across all patches, facilitating parameter sharing and reducing the overall number of parameters in the network, which is crucial for scalability and generalization.

Finally, a Transformer model (Vaswani et al., 2017) builds on the self-attention layer in two main ways. First, instead of employing self-attention in a single head, Transformer models utilize multi-head attention mechanisms. This involves performing self-attention multiple times in parallel, each with its own set of learnable parameters. By allowing the model to attend to different parts of the input sequence simultaneously, multi-head attention enhances the model's ability to capture diverse and complex relationships within the data. Secondly, Transformers employ feed-forward neural networks, also known as point-wise fully connected layers, to process the outputs of the multi-head attention mechanism.

We refer to Goodfellow et al. (2016) and Wüthrich and Merz (2021) for more detail on these network layers.

3.5. Network calibration

The network's performance hinges on appropriately calibrating the weights in different layers, denoted as $w_{l,j}^{(k)}$. In the case of FCN layers, these weights manifest as matrices $W^{(k)}$ and a bias term $w_0^{(k)}$ for each k ranging from 1 to m . Meanwhile, for embedding layers, the weights correspond to the coordinates of levels in the new embedding space, represented as $e^{(l)}$ for all l in \mathcal{L} . These weights need to be optimized not only to fit the available data well but also to produce accurate predictions on new examples. To assess this, the standard procedure entails dividing the available data into two parts: the *learning* sample, utilized for model calibration, and the *testing* sample, used to evaluate out-of-sample performance. The training process arise an unconstrained optimization, where a suitable loss function $L(w_{l,j}^{(k)}, \cdot)$ is chosen, and the objective is to find its minimum. The NN training employs the back-propagation (BP) algorithm, wherein weight updates are determined by the gradient of the loss function measured on the learning sample. The iterative adjustment of weights aims to minimize the error between the network outputs and reference values. The training complexity increases with the number of layers and units per layer in the network architecture; these are hyperparameters that should be suitably chosen. Indeed, a too deep NN

would lead to *overfitting* producing a model unable to generalize to new data points, or, alternatively, lead to the *vanishing gradient* problem, which prevents the BP algorithm from updating the weights successfully. One remedy for the *overfitting* problem is the application of regularization methods such as dropout. Dropout Srivastava et al. (2014) is a *stochastic* technique that ignores, that is, sets to zero, some randomly chosen units during the network fitting. This is generally achieved by multiplying the output of the different layers by independent realizations of a Bernoulli random variable with parameter $p \in [0, 1]$. Mathematically, the introduction of the dropout in a FCN layer, for example, in the k -th layer, induces the following structure:

$$\begin{aligned} r_i^{(k)} &\sim \text{Bernoulli}(p) \\ \mathbf{z}^{(k-1)}(\mathbf{x}) &= \mathbf{r}^{(k)} * \mathbf{z}^{(k-1)}(\mathbf{x}) \\ \mathbf{z}^{(k)}(\mathbf{x}) &= \phi(\mathbf{w}_0^k + W^{(k)}\mathbf{z}^{(k-1)}(\mathbf{x})), \end{aligned}$$

where $*$ denotes the element-wise product and $\mathbf{r}^{(k)} = (r_1^{(k)}, \dots, r_{q_{k-1}}^{(k)})$ is a vector of independent Bernoulli random variables, each of which has a probability p of being 1. This mechanism leads to zero the value of some elements and encourages more robust learning.

For a comprehensive understanding of neural networks and back-propagation, a detailed discussion can be found in Goodfellow et al. (2016).

4. YC_ATT – a neural network model for multiple yield curve modeling and forecasting

This section formally outlines the proposed deep learning-based multiple yield curve model, which we call YC_ATT model. We make the assumption that the different yield curve observations over time are members of families of yield curves that are defined on the same tenor structure \mathcal{M} . For example, we will have multiple observations of the US yield curve, which is one family of yield curves among other families of yield curves defined by geography. The name YC_ATT emphasizes that the architecture exploits attention mechanism, for enabling efficient processing of a (multivariate) time series of yield time series, implicit information selection, and formulation of accurate predictions. Our goal is to derive precise point forecasts and effectively measure the uncertainty of the future yield curve. To achieve this, we have designed a model that allows us to jointly estimate a measure of the central tendency of the distribution of future spot rates (mean or median) and two quantiles at a specified tail level, forming confidence intervals for the predictions. We note that, despite this section focusing on the attention-based architecture, nonetheless in Section 5 we will present a comparative study with some variants of this architecture based on different deep learning models. Denoting as $\mathbf{y}_t^{(i)} = (y_t^{(i)}(\tau))_{\tau \in \mathcal{M}} \in \mathbb{R}^M$ the vector (yield curve) containing the spot rates for different maturities related to the curve family i , at time t , where $M = |\mathcal{M}|$.

Choosing a look-back period $L \in \mathbb{N}$ and a confidence level $\alpha \in (0, 1)$, we design a model that takes as input the matrix of yield curves related to the previous L dates denoted as $Y_{t-L,t}^{(i)} = (y_{t-l}^{(i)})_{0 \leq l \leq L} \in \mathbb{R}^{(L+1) \times M}$ and the label related to the curve family $i \in \mathcal{I}$, and produces three output vector $\widehat{\mathbf{y}}_{\alpha/2,t+1}^{(i)}, \widehat{\mathbf{y}}_{t+1}^{(i)}, \widehat{\mathbf{y}}_{1-\alpha/2,t+1}^{(i)} \in \mathbb{R}^M$. More specifically, $\widehat{\mathbf{y}}_{\alpha/2,t+1}^{(i)}$ is the vector of quantiles at levels of confidence $\alpha/2$, $\widehat{\mathbf{y}}_{t+1}^{(i)}$ is the vector of a such measure of the central tendency of the distribution (in what follows, we consider both the mean or the median), and $\widehat{\mathbf{y}}_{1-\alpha/2,t+1}^{(i)}$ is the vector of quantiles at level $(1 - \alpha/2)$. The quantile statistics will be calibrated to quantify the uncertainty in future yields. We desire to learn the mapping:

$$\begin{aligned} f : \mathbb{R}^{(L+1) \times M} \times \mathcal{I} &\rightarrow \mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^M \\ (Y_{t-L,t}^{(i)}, i) &\mapsto (\widehat{\mathbf{y}}_{\alpha/2,t+1}^{(i)}, \widehat{\mathbf{y}}_{t+1}^{(i)}, \widehat{\mathbf{y}}_{1-\alpha/2,t+1}^{(i)}) = f(Y_{t-L,t}^{(i)}, i). \end{aligned}$$

We approximate $f(\cdot)$ with a DNN that combines embedding, FCN, and attention layers. More specifically, we process the label i using an embedding layer of size $q_{\mathcal{I}} \in \mathbb{N}$. It is a mapping with the following structure:

$$\mathbf{e} : \{\text{family}_1, \dots, \text{family}_{n_T}\} \mapsto \mathbb{R}^{q_T}, \quad \text{family}_i \mapsto \mathbf{e}(\text{family}_i) \stackrel{\text{set}}{=} \mathbf{e}^{(i)}$$

where $\mathbf{e}^{(i)}$ is a vector that encoded the information related to the family of curves i . It can be seen as a new representation of i in a new q_T -dimensional real-valued space that is optimal with respect to the response variable. On the other hand, we process the matrix of the past yield curves with three time-distributed FCN layers aiming to derive, respectively, the query, key, and value vectors for input into the attention component. The time-distributed mechanism consists of applying the same transformation (the same layer) to each row of the matrix $Y_{t-L,t}^{(i)}$ containing the yield curves of the different dates. We apply three FCN layers that can be formalized as:

$$\mathbf{z}^{(j)} : \mathbb{R}^M \rightarrow \mathbb{R}^{q_A}, \quad \mathbf{y}_t^{(i)} \mapsto \mathbf{z}^{(j)}(\mathbf{y}_t^{(i)})$$

where q_A is the number of units of the three layers, and $j \in \{Q, K, V\}$. They produce the following matrices:

$$\begin{aligned} Q_t^{(i)} &= (\mathbf{q}_{t-l}^{(i)})_{0 \leq l \leq L} \in \mathbb{R}^{(L+1) \times q_A}, & \mathbf{q}_{t-l}^{(i)} &= \mathbf{z}^{(Q)}(\mathbf{y}_{t-l}^{(i)}) = \phi^{(Q)}(\mathbf{w}_0^{(Q)} + W^{(Q)}\mathbf{y}_{t-l}^{(i)}) \in \mathbb{R}^{q_A} \\ K_t^{(i)} &= (\mathbf{k}_{t-l}^{(i)})_{0 \leq l \leq L} \in \mathbb{R}^{(L+1) \times q_A}, & \mathbf{k}_{t-l}^{(i)} &= \mathbf{z}^{(K)}(\mathbf{y}_{t-l}^{(i)}) = \phi^{(K)}(\mathbf{w}_0^{(K)} + W^{(K)}\mathbf{y}_{t-l}^{(i)}) \in \mathbb{R}^{q_A} \\ V_t^{(i)} &= (\mathbf{v}_{t-l}^{(i)})_{0 \leq l \leq L} \in \mathbb{R}^{(L+1) \times q_A}, & \mathbf{v}_{t-l}^{(i)} &= \mathbf{z}^{(V)}(\mathbf{y}_{t-l}^{(i)}) = \phi^{(V)}(\mathbf{w}_0^{(V)} + W^{(V)}\mathbf{y}_{t-l}^{(i)}) \in \mathbb{R}^{q_A}, \end{aligned}$$

where $\mathbf{w}_0^{(j)} \in \mathbb{R}^{q_A}$, $W^{(j)} \in \mathbb{R}^{q_A \times M}$ are network parameters, and $\phi^{(j)} : \mathbb{R} \mapsto \mathbb{R}$ are activation functions, $j \in \{Q, K, V\}$. The three matrices $Q_t^{(i)}, K_t^{(i)}, V_t^{(i)}$ are processed by an attention layer that combine them according to the following transformation:

$$X_t^{(i)} = \text{softmax}\left(\frac{Q_t^{(i)}(K_t^{(i)})^\top}{\sqrt{d_k}}\right)V_t^{(i)} \in \mathbb{R}^{(L+1) \times q_A}$$

The output of this layer is arranged in a vector $\mathbf{x}_t^{(i)} = \text{vec}(X_t^{(i)})$ that can be interpreted as a set of features that summarizes the information contained in the matrix $Y_{t-L,t}^{(i)}$. We remark that other deep learning models could be used to derive the vector of features from the matrix of the past data; here, we use the attention mechanism, having found that this produces the best out-of-sample forecasting performance.

Finally, we apply three different $|M|$ -dimensional FCN layers to the set of features $\mathbf{x}_t^{(i)}$ for deriving the predictions related to the three vectors of output statistics of interest. The first layer aims to derive the lower quantiles of the yields, the second one aims to compute the mean (or the median), while the third one is related to the upper quantiles. Each unit of these layers is dedicated to a specific maturity, and we have in each layer as many units as the maturities considered. Before inputting the outputs of the attention layer $\mathbf{x}_t^{(i)}$ to these FCNs, we apply dropout to $\mathbf{x}_t^{(i)}$ for regularization and reducing overfitting. In this setting, the predictions are obtained according to the following set of equations:

$$\begin{aligned} r_i^{(\text{att})} &\sim \text{Bernoulli}(p_1^{(\text{att})}) \\ \dot{\mathbf{x}}_t^{(i)} &= \mathbf{r}^{(\text{att})} * \mathbf{x}_t^{(i)} \\ \hat{\mathbf{y}}_{t+1}^{(i)} &= g\left(\mathbf{b}_c + U_c \mathbf{e}^{(i)} + W_c \dot{\mathbf{x}}_t^{(i)}\right) \end{aligned} \tag{4.1}$$

$$\hat{\mathbf{y}}_{\alpha/2,t+1}^{(i)} = \hat{\mathbf{y}}_{t+1}^{(i)} - \phi_+ \left(\mathbf{b}_{lb} + U_{lb} \mathbf{e}^{(i)} + W_{lb} \dot{\mathbf{x}}_t^{(i)}\right) \tag{4.2}$$

$$\hat{\mathbf{y}}_{1-\alpha/2,t+1}^{(i)} = \hat{\mathbf{y}}_{t+1}^{(i)} + \phi_+ \left(\mathbf{b}_{ub} + U_{ub} \mathbf{e}^{(i)} + W_{ub} \dot{\mathbf{x}}_t^{(i)}\right) \tag{4.3}$$

where $\mathbf{r}^{(k)} = (r_1^{(k)}, \dots, r_{(L+1) \times q_A}^{(k)})$, $p_1^{(\text{att})} \in [0, 1]$ is the dropout rate, $g : \mathbb{R} \rightarrow \mathbb{R}$ and $\phi_+ : \mathbb{R} \rightarrow]0, +\infty)$ are strictly monotone functions, and $\mathbf{b}_j, U_j, W_j, j \in \{c, lb, ub\}$ are network parameters. We remark that the number of units in these three output layers corresponds to the number of maturities considered, denoted

as M . This means that if we modify the set of maturities, we only need to adjust the size of the input and output layers. This flexibility ensures that the model can be applied across diverse financial and insurance contexts, accommodating scenarios where yield curves may exhibit varying numbers of maturities. Looking at this set of equations, some remarks can be made:

- (1) The model presents some connections with the affine models¹ discussed in Piazzesi (2010). Considering a single maturity τ , Equation (4.1) can be formulated as follows:

$$g^{-1}(\widehat{\mathbf{y}}_{t+1}^{(i)}(\tau)) = b_{c,\tau} + \langle \mathbf{u}_{c,\tau}, \mathbf{e}^{(i)} \rangle + \langle \mathbf{w}_{c,\tau}, \mathbf{x}_t^{(i)} \rangle.$$

Indeed, it has the constant-plus-linear structure and depends on the vector of variables $\mathbf{x}_t^{(i)}$ derived by the past observed data. Furthermore, the following additional arguments can be provided:

- b_c can be interpreted as a sort of global intercept;
 - $\langle \mathbf{u}_{c,\tau}, \mathbf{e}^{(i)} \rangle$ is an intercept correction related to the curve family i and maturity τ ;
 - $\mathbf{w}_{c,\tau}$ is the vector of maturity-specific weights associated with $\mathbf{x}_t^{(i)}$. These coefficients are shared among all the curve families since these do not depend on i .
- (2) The formulation we propose avoids potential quantile crossing, which refers to the scenario in which the estimated quantiles of a probability distribution do not respect the expected order. Inaccurate or inconsistent quantile estimates have the potential to affect the reliability and interpretability of predictions from the model. Indeed, the use of the activation function $\phi_+(\cdot)$ that assumes only positive values ensures that:

$$\widehat{\mathbf{y}}_{\alpha/2,t+1}^{(i)} < \widehat{\mathbf{y}}_{t+1}^{(i)} < \widehat{\mathbf{y}}_{\alpha/2,t+1}^{(i)}.$$

- (3) Rewriting Equations (4.2) for a single maturity, we have

$$\phi_+^{-1}(\widehat{\mathbf{y}}_{t+1}^{(i)}(\tau) - \widehat{\mathbf{y}}_{\alpha/2,t+1}^{(i)}(\tau)) = b_{lb,\tau} + \langle \mathbf{u}_{lb,\tau}, \mathbf{e}^{(i)} \rangle + \langle \mathbf{w}_{lb,\tau}, \mathbf{x}_t^{(i)} \rangle$$

emphasizing that we model, on the ϕ_+^{-1} scale, the difference between the central measure and lower quantile at a given maturity τ is an affine model. Similar comments can be made for the difference between the upper quantile and the central measure.

We illustrate the YC_ATT model in Figures 1 and 2.

The calibration of the multi-output network is carried out accordingly, with a loss function specifically designed for our aim. It is the sum of three components:

$$\begin{aligned} \mathcal{L}_{\alpha,\gamma}(\boldsymbol{\theta}) &= \mathcal{L}_{\alpha/2}^{(1)}(\boldsymbol{\theta}) + \mathcal{L}_{\gamma}^{(2)}(\boldsymbol{\theta}) + \mathcal{L}_{1-\alpha/2}^{(3)}(\boldsymbol{\theta}) \\ &= \sum_{i,t,\tau} \ell_{\alpha/2}(y_t^{(i)}(\tau) - \widehat{y}_{\alpha/2,t}^{(i)}(\tau)) + \sum_{i,t,\tau} h_{\gamma}(y_t^{(i)}(\tau) - \widehat{y}_t^{(i)}(\tau)) + \sum_{i,t,\tau} \ell_{1-\alpha/2}(y_t^{(i)}(\tau) - \widehat{y}_{1-\alpha/2,t}^{(i)}(\tau)) \end{aligned} \quad (4.4)$$

where $\ell_{\alpha}(u)$, $\alpha \in (0, 1)$ is the asymmetric loss function introduced in Koenker and Bassett (1978) also known in ML literature as pinball function:

$$\ell_{\alpha}(u) = \begin{cases} (1 - \alpha)|u| & u \leq 0 \\ \alpha|u| & u > 0, \end{cases}$$

and $h_{\gamma}(u)$, $\gamma \in \{1, 2\}$ is

$$h_{\gamma}(u) = \begin{cases} |u| & \gamma = 1 \\ u^2 & \gamma = 2, \end{cases}$$

¹The term ‘‘affine term structure model’’ is used in different ways by the literature, we refer to the definition given in Chapter 12 of [Piazzesi (2010)].

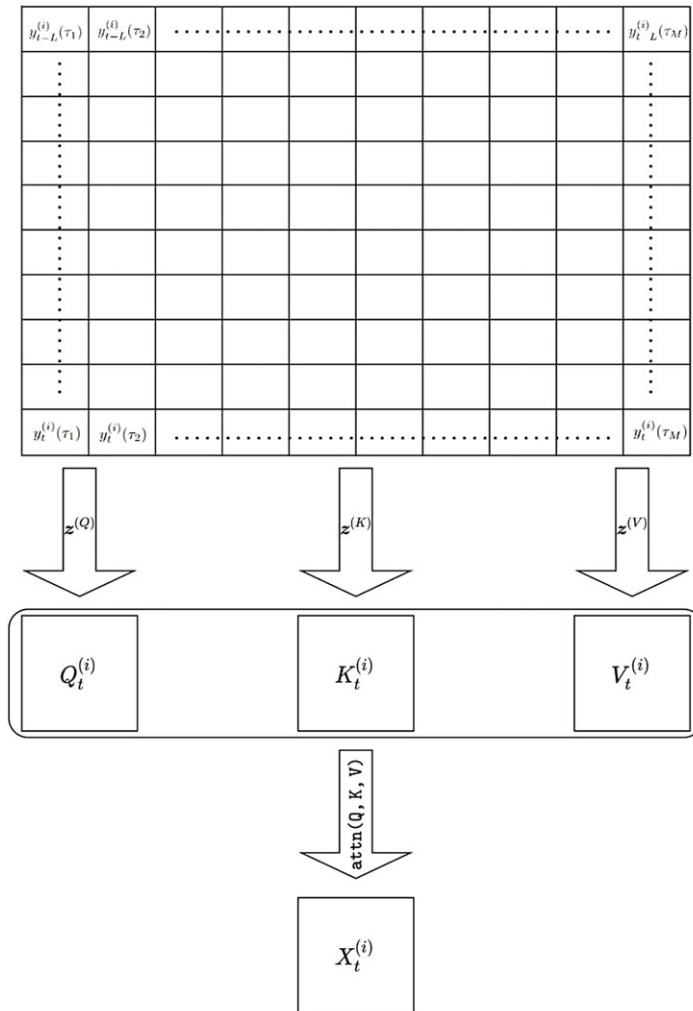


Figure 1. Diagram of the feature processing components of the YC_ATT model. A matrix of spot rates is processed by three FCN layers in a time-distributed manner, to derive the key, query, and value matrices which are then input into a self-attention operation.

We emphasize that the first term of the loss function is the pinball function with parameter $\alpha/2$ associated with the estimation of the lower quantile. The second term represents a generic function linked to the estimation of the central tendency, while the last term is the pinball function with parameters $\alpha/2$ associated with the estimation of the upper quantile.

Regarding the function $h_\gamma(\cdot)$, it is a general function that encompasses two different criteria commonly used in the literature to fit models describing the central tendency of the distribution of the response variable. If one wishes to estimate the mean, γ should be set to 2, in which case $h_\gamma(\cdot)$ becomes the mean squared error (MSE). Conversely, if one wishes to estimate the median, γ should be set to 1, in which case $h_\gamma(\cdot)$ becomes the mean absolute error (MAE).

5. Numerical experiments

We present some numerical experiments conducted on data provided by the European Insurance and Occupational Pensions Authority (EIOPA). The authority publishes risk-free interest rate term structures

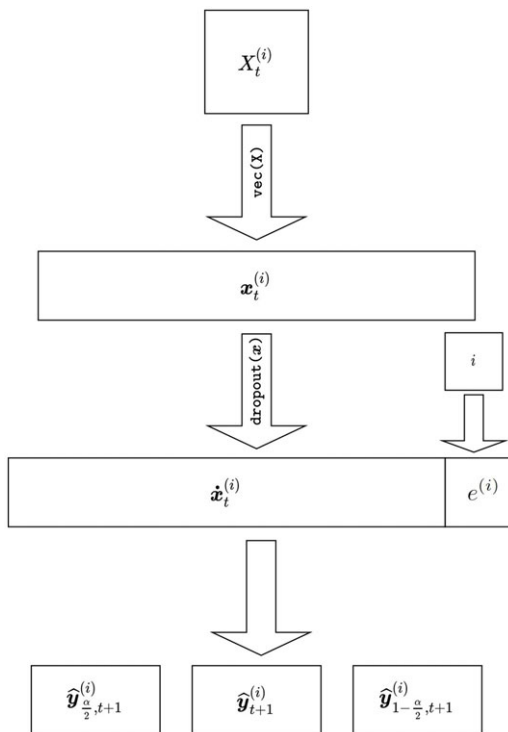


Figure 2. Diagram of the output components of the YC_ATT model. The matrix of features produced by the first part of the model are flattened into a vector and then dropout is applied. We add a categorical embedding to this vector, and, finally, then the best-estimate and quantile predictions are produced.

derived from government bonds of various countries; these are published on a monthly basis as spot curves, which are the default option for use in the Solvency II regime for discount rates. We define \mathcal{I} as the set of all available countries and \mathcal{M} as the set of maturities expressed in years, $\mathcal{M} = \{\tau \in \mathbb{N} : \tau \leq 150\}$, such that $|\mathcal{M}| = 150$. We note that we are in a setting where the set of maturities \mathcal{M} is the same for all the families considered, as is the case with the EIOPA data.² Our sample data covers the period from December 2015 to December 2021, and Figure A2 provides a graphical representation of the EIOPA data.

Selecting an observation time t_0 , we partition the full dataset into two parts. The first, containing yields before time t_0 (referred to as the *learning sample*), is used for model calibration. The second, encompassing data after time t_0 (referred to as the *testing sample*), is employed to evaluate the out-of-sample accuracy of the models. Interval forecasts are constructed by considering the case in which we desire a coverage probability $\alpha = 0.95$. As suggested in the application of ML methods, we preprocess the data by applying min-max scaling.

To benchmark our model, we consider the dynamic NS proposed in Diebold and Li (2006) and its related Svensson (NSS) extension. We examine both cases where the latent factors follow individual AR(1) models and the case of a single multivariate VAR(1) model, denoted respectively as NS_AR (NSS_AR) and NS_VAR (NSS_VAR).

Since our focus is on measuring forecasting accuracy in terms of both point and interval forecasts, we employ several metrics to compare the models. Concerning point forecast accuracy, we use the global

²More information about EIOPA data are available at https://www.eiopa.europa.eu/tools-and-data/risk-free-interest-rate-term-structures_en.

Table 1. Performance of the NS and NSS models in terms of MSE, MAE, PICP, and MPIW; The MSE values are scaled by a factor of 10^5 , while the MAE values are scaled by a factor of 10^2 .

Model	MSE	MAE	PICP	MPIW
NS_AR	0.7433	0.4496	0.9984	0.0540
NS_VAR	0.4977	0.3492	0.7288	0.0080
NSS_AR	0.5379	0.3709	0.9987	0.4253
NSS_VAR	0.4626	0.3226	0.7462	0.0307

(i.e., evaluated for all countries in the dataset) MSE and the MAE defined as follows:

$$MSE = \frac{1}{n} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{\tau \in \mathcal{M}} (y_t^{(i)}(\tau) - \hat{y}_t^{(i)}(\tau))^2, \tag{5.1}$$

$$MAE = \frac{1}{n} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{\tau \in \mathcal{M}} |y_t^{(i)}(\tau) - \hat{y}_t^{(i)}(\tau)|, \tag{5.2}$$

where $n \in \mathbb{N}$ is the number of instances. We emphasize that MSE is based on the l_2 -norm of the errors and penalizes a larger deviation from the observed values with respect to the MAE which is based on the l_1 -norm of the errors. For measuring the interval prediction accuracy, we consider the prediction interval coverage probability (PICP):

$$PICP = \frac{1}{n} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{\tau \in \mathcal{M}} \mathbb{1}_{[y_t^{(i)}(\tau) \in [\hat{y}_{t, LB}^{(i)}(\tau), \hat{y}_{t, UB}^{(i)}(\tau)]]} \tag{5.3}$$

Furthermore, we also analyse the mean prediction interval width (MPIW) to take into account the width of the confidence interval:

$$MPIW = \frac{1}{n} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{\tau \in \mathcal{M}} (\hat{y}_{t, UB}^{(i)}(\tau) - \hat{y}_{t, LB}^{(i)}(\tau)). \tag{5.4}$$

Intuitively, a good model should provide a PICP close to the value of α , indicating that the model is well calibrated, while having as small an MPIW, as possible. Table 1 presents the performance related to the four measures considered for the NS and NSS benchmark models.

Regarding the accuracy of point forecasts, it is worth noting that employing a VAR(1) model instead of independent AR(1) models for modeling the latent factors dynamics enhances the performance of both the NS and NSS models. This improvement is evident in terms of both MSE and MAE. Furthermore, it is noteworthy that NSS models consistently demonstrate superior accuracy compared to NS models. Turning our attention to interval forecasts, it becomes apparent that models incorporating AR processes tend to exhibit over coverage, as indicated by excessively high PICP and impractically large interval widths providing very limited information content. In contrast, versions based on VAR models for NS and NSS tend to yield more reasonable interval widths. However, given their relatively low coverage probability, they fall short of adequately capturing the uncertainty in future yields. In summary, we designate the NSS_VAR model as the optimal choice, as it yields the lowest MSE and MAE, along with the highest PICP and a reasonable MPIW. Consequently, we will employ this model for subsequent comparisons throughout the remainder of the paper.

Now, we can focus on the YC_ATT model. We examine two variants distinguished by the gamma parameter in the loss function employed for calibration. The first variant is calibrated by setting $\gamma = 1$ (denoted as YC_ATT $_{\gamma=1}$) and, in that case, the second component of the loss function is the MAE. On the other hand, the second variant is calibrated with $\gamma = 2$ and the component of the loss related to the central tendency of the distribution then uses the MSE. Regarding the other hyperparameters, we have configured the number of units in the dense intermediate layers comprising the attention component

Table 2. Out-of-sample performance of the different deep learning models in terms of MSE, MAE, PICP, and MPIW; the MSE values are scaled by a factor of 10^5 , while the MAE values are scaled by a factor of 10^2 . Bold indicates the smallest value, or, for the PICP, the value closest to $\alpha = 0.95$.

Model	MSE		MAE		PICP		MPIW	
	Average	Ensemble	Average	Ensemble	Average	Ensemble	Average	Ensemble
YC_ATT $_{\gamma=1}$	0.2947	0.2887	0.2667	0.2616	0.9154	0.9191	0.0106	0.0106
YC_ATT $_{\gamma=2}$	0.3663	0.3638	0.3463	0.3451	0.8528	0.8573	0.0105	0.0105
YC_CONV $_{\gamma=1}$	0.3778	0.3642	0.2975	0.2850	0.9035	0.9235	0.0115	0.115
YC_CONV $_{\gamma=2}$	0.4258	0.4244	0.3890	0.3884	0.8509	0.8530	0.0110	0.0110
YC_LSTM $_{\gamma=1}$	0.4272	0.4111	0.3164	0.2970	0.7757	0.8147	0.0093	0.0093
YC_LSTM $_{\gamma=2}$	0.3898	0.3697	0.3352	0.3198	0.6911	0.7081	0.0084	0.0084
YC_TRAN $_{\gamma=1}$	0.4308	0.4167	0.3313	0.3168	0.8371	0.8645	0.0113	0.0113
YC_TRANS $_{\gamma=2}$	0.4232	0.4124	0.4042	0.3987	0.5771	0.5760	0.0091	0.0091

of the model as $q_A = 8$, while the dropout rate is set $p_1^{(att)} = 0.5$. As previously indicated, the part of the network architecture that processes the past yields can be constructed using various neural network blocks. The attention layer, discussed earlier, is just one of the available choices. We further explore the utilization of other well-established deep learning models that have demonstrated success in modeling sequential data. Specifically, we investigate other three variants:

- YC_LSTM: based on the LSTM network of Hochreiter and Schmidhuber (1997) that is a popular kind of RNN;
- A simplification of the YC_ATT model, that removes the attention mechanism and relies only on processing the yield curves using time-distributed FCNs; since these are also called one-dimensional CNNs, we call this variant YC_CONV;
- YC_TRANS which is a more complex Transformer-based model (Vaswani et al., 2017), which adds extra FCNs to the YC_ATT model.

For all these models, we set $g(x)$ and $\phi_+(x)$ equal to the sigmoid function, while employing hyperbolic tangent activation for all intermediate layers. In order to make the comparison fair, also for these variants we set the number of units in the layers equal to 8. Furthermore, for each one of these architectures, we respectively test the variants with $\gamma = 1$ and $\gamma = 2$.

Table 2 presents the performance metrics, including MSE, MAE, PICP, and MPIW, for various deep learning models.

To account for randomness in batch sampling and the random initial parameters for each network for the optimization, multiple training attempts were conducted, and the results represent the average performance over 10 training attempts. The boxplots related to the 10 training attempts are shown in Figure A3 (Appendix A). Additionally, we also consider the use of averaging ensemble mechanisms. In that case, predictions are generated by averaging the predictions from the 10 trained models. More specifically, let $\hat{y}_t^{(i,k)}(\tau)$ be the predictions generated by the k -th training attempt of a given model, for $k = 1, \dots, 10$, according to the use of the averaging ensemble mechanism, predictions are calculated as:

$$\hat{y}_t^{(i,ENS)}(\tau) = \frac{1}{10} \sum_{k=1}^{10} \hat{y}_t^{(i,k)}(\tau).$$

We can then calculate the performance metrics (5.1)–(5.4) using the average ensemble predictions $\hat{y}_t^{(i,ENS)}(\tau)$. The idea is that the averaging mechanism allows the errors from different models to counter-balance each other, potentially resulting in the ensemble’s performance being superior to the average performance of individual models. Additionally, this technique can be applied to the lower and upper

quantile predictions $\hat{y}_{i, LB}^{(i,k)}(\tau)$ and $\hat{y}_{i, UB}^{(i,k)}(\tau)$, enhancing the interval predictions. We remark that, generally, the average performance across different runs differs from the performance of the ensemble predictions obtained by averaging the predictions of the different models. However, we note that the average MPIW model performance across different training attempts coincides with the MPIW related to the ensemble predictions. The proof of this statement is provided in Appendix A.2.

Several noteworthy findings emerge. First, the consistent adoption of ensemble mechanisms leads to superior performance compared to the average performance across individual models. This trend holds across all examined models and metrics, aligning with prevailing findings in the literature on predictive modeling with deep learning. Moreover, a notable trend is highlighted: models configured with $\gamma = 1$ consistently outperform their counterparts with $\gamma = 2$ in both point forecasts and interval forecast accuracy. This outcome can be attributed to the robust nature of MAE as a measure, providing increased resilience to outliers and contributing to a more stable model calibration. Surprisingly, this is even the case when evaluating the models using the MSE metric. Finally, among the deep learning models, the YC_ATT model with $\gamma = 1$ demonstrates the best performance, suggesting that this architecture is particularly well suited for the regression task at hand. Also notable is that the ensemble predictions of the YC_ATT model with $\gamma = 1$ are almost the best calibrated, as measured by the PICP metric, while the bounds are relatively narrow, as measured by the MPIW metric. Nonetheless, the best performance on the PICP metric on a standalone basis is the YC_CONV with $\gamma = 1$. Regarding ensemble predictions, it is noteworthy that the average MPIW model across various training attempts coincides with the MPIW associated with ensemble predictions. The proof of this assertion can be found in the Appendix. In summary, evaluating the models on all of the metrics, the YC_ATT performs the best overall.

Figure 3 graphically compares the point and interval forecasts generated by the NSS_VAR and YC_ATT models for yield curves across various countries. The figure refers to the central date within the forecasting horizon, specifically June 2021, as the forecasting period spans monthly observations from January to December 2021. The figure shows the actual observed yield curve, and an out-of-sample forecast of the best-estimate and quantiles using the previous 10 months of data; model parameters are those calibrated using data up to the end of 2020. Upon examination, it becomes evident that the NSS_VAR model falls short in accurately depicting the shape of yield curves in certain countries. Notably, the realized yields deviate significantly from the projected forecasts and extend beyond the confidence intervals. This discrepancy is particularly pronounced in the cases of Brazil, Colombia, Mexico, India, Russia, South Africa, and others. In contrast, the YC_ATT model exhibits more flexibility and demonstrates the ability of effectively capturing the uncertainties associated with future yields.

Figure 4 offers a more in-depth analysis of the performance exhibited by the YC_ATT and NSS_VAR models across various countries of the EIOPA yield curves. It illustrates the MSE, MAE, and PICP produced by the different models for the different yield curve families. From the standpoint of point forecasts, it is noted that in certain instances, the YC_ATT and NSS_VAR models produce comparable results. However, for specific cases, the YC_ATT model demonstrates a notable enhancement, particularly evident in developing countries such as Brazil, Chile, Malaysia, Mexico, South Africa, and Turkey. This finding suggests that yields shows that when the yield curve evolution can be adequately described by the NSS_VAR model, our YC_ATT tends to replicate the same point predictions. On the other hand, for the yield curve families for which the NSS_VAR model is not optimal, the YC_ATT improve the results. In terms of interval forecasts, there is a noticeable improvement attributable to the YC_ATT model, impacting all the EIOPA yield curves under consideration.

To gain insights into the mechanism underlying the YC_ATT model, we investigate the features that the model extracts from the input data and that are used by the output layers to derive key statistics. We consider the time series of the feature vector $(\mathbf{e}^{(i)}, \mathbf{x}_t^{(i)}) \in \mathbb{R}^{q_X + q_A \times M}$, $t \in \mathcal{T}$, and conduct PCA to reduce the dimensionality. We extract the first four principal components (PCs) which explain the 97% of the variability of $(\mathbf{e}^{(i)}, \mathbf{x}_t^{(i)})$ such that we have a mapping with the structure $\mathbb{R}^{q_X + (L+1) \times q_A} \mapsto \mathbb{R}^4$ for each feature vector. To assess the similarity of information contained in the four PCs with the time series of $\beta_t^{(i)}$ factors of the NSS model, we calculate, for each β -coefficient, the average correlation (in absolute value) with

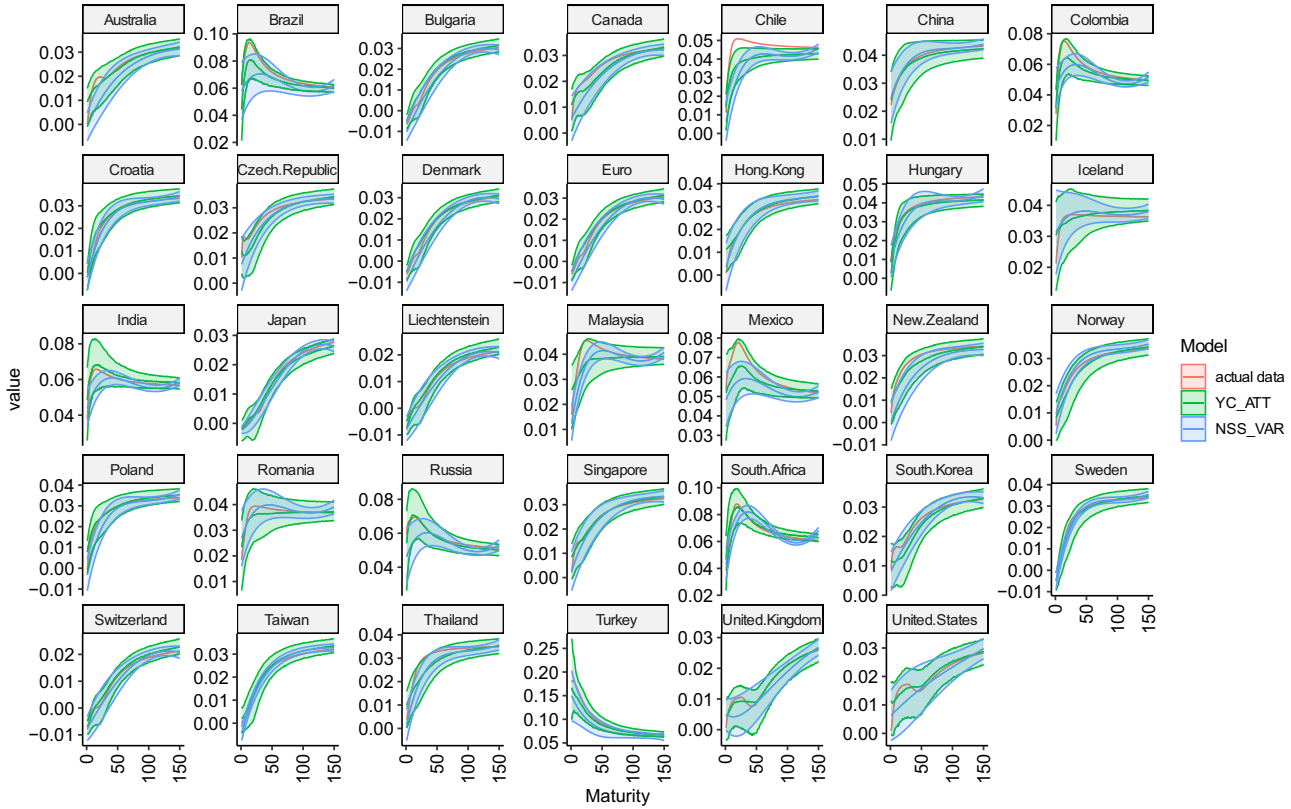


Figure 3. Point and interval forecasts for EIOPA yield curves generated by the NSS_VAR and YC_ATT models as of June 2021, the central date of the forecasting period.

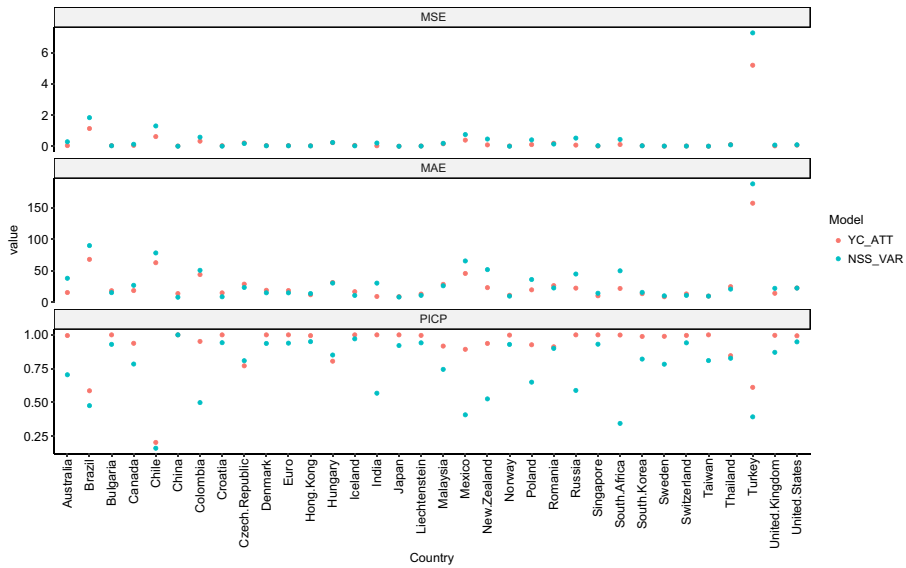


Figure 4. MSE, MAE, and PICP obtained by the YC_ATT and NSS_VAR models in the different countries.

the first four PCs. These quantities measure how much the PCs are linearly correlated with the $\beta_t^{(i)}$ factors that contain information about the level, slope, and curvature of the yield curves. The results for each yield curve family are presented in Figure 5. Notably, we observe that very high correlations are detected in some cases, while low correlations are obtained in the others. Examining this figure in conjunction with Figure 4, we note that in families exhibiting similar performances, such as Euro, Bulgaria, Denmark, and Iceland, the PCs are high correlated with the NSS latent factors. Conversely, instances of notable improvements by the YC_ATT model, as seen in Mexico, Turkey, Malaysia, and Brazil, are accompanied by smaller correlations between the PC components from the output of the attention layer and the beta parameters of NSS. In essence, this figure confirms that in cases where the yield curves follow a process adequately described by the NSS models, the YC_ATT model replicates this by extracting variables highly correlated with beta. However, when this is not the case, and the yields present more complex patterns, our attention model derives features that deviate from the NSS model, resulting in better outcomes.

6. Extensions and variants of the YC_ATT model

In this section, we explore potential extensions and variants of the YC_ATT, considering some modifications that aim to enhance the modeling of yield curves and their associated uncertainties.

6.1. Deep ensemble

An alternative approach for modeling uncertainty in future yields is the deep ensemble (DE) method discussed in Lakshminarayanan et al. (2017). In contrast to the quantile regression-based YC_ATT, this method relies on distributional assumptions for the response. The idea consists of formulating a heteroscedastic Gaussian regression model that provides joint estimates for both the mean and variance of the yields, denoted as $(y_t^{(i)}(\tau), (\sigma_t^{(i)}(\tau))^2)$. This technique not only facilitates the extraction of additional insights into future yields but also accommodates heteroscedasticity in the modeling process; this is different from the networks calibrated in the previous section which are equivalent to assuming that the

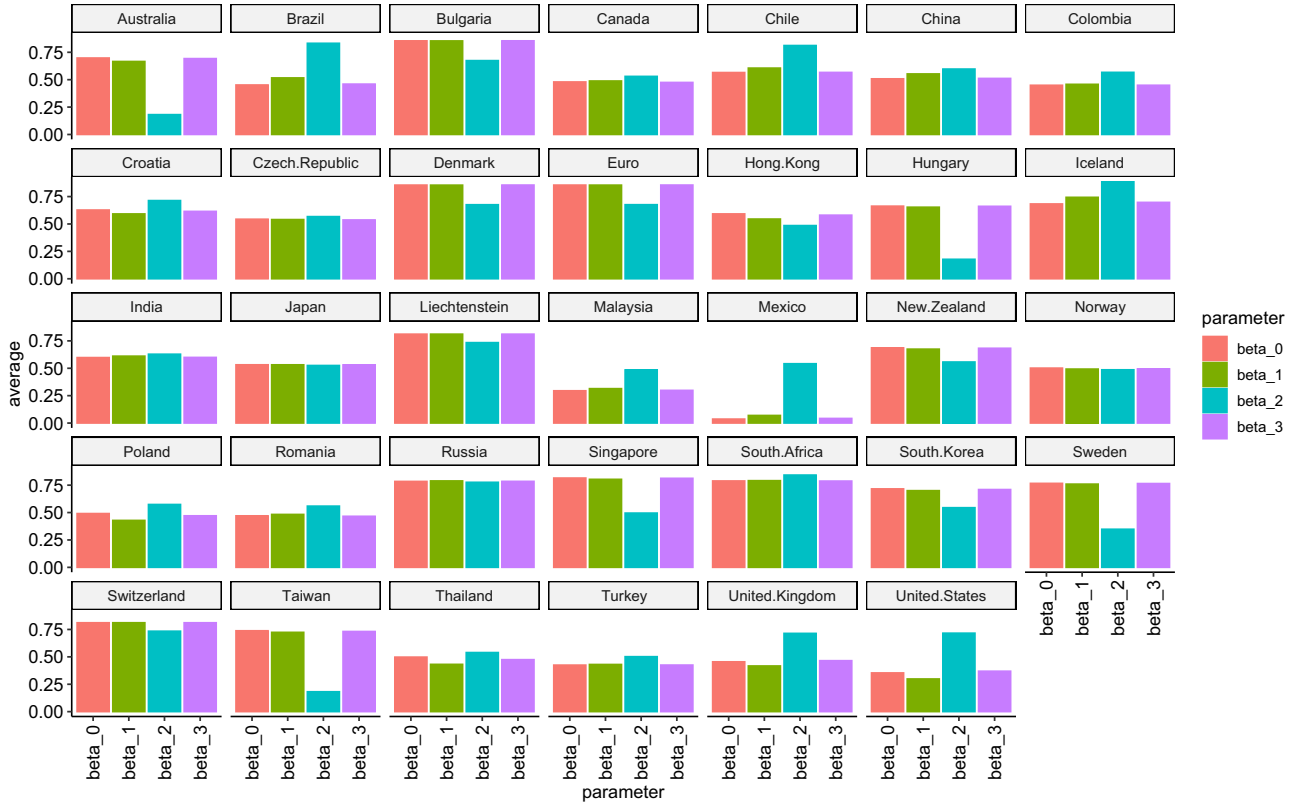


Figure 5. Linear correlation coefficients (in absolute value) of the four PCs derived from the learned features, represented as $(\mathbf{e}^{(i)}, \mathbf{x}_i^{(i)}) \in \mathbb{R}^{qZ+(L+1) \times q_A}$, with respect to the $\beta_i^{(i)}$ factors of the NSS model for the different yield curve families.

responses follow a homoscedastic Laplace or Gaussian distribution for choices of $\gamma \in \{0, 1\}$, respectively. Furthermore, confidence intervals can be derived using these estimates. In this vein, we design a network architecture with two output layers that produces predictions of the yield curves and their related variances. We refer to this model as YC_ATT_DE. As discussed in Nix and Weigend (1994), model calibration of the DE model can be performed by minimizing the following loss function:

$$\mathcal{L}(\theta) = \sum_{i,i,\tau} \left[\frac{y_t^{(i)}(\tau) - \hat{y}_t^{(i)}(\tau)}{(\sigma_t^{(i)}(\tau))^2} + \frac{\log((\sigma_t^{(i)}(\tau))^2)}{2} \right].$$

The first component represents the MSE between the prediction and the actual yields, scaled by the variance. Meanwhile, the second term acts as a penalization factor for observations with notably high estimated variances. We calibrate the YC_ATT_DE model on the EIOPA data in the same setting used above. Figure 6 shows the standard deviation estimates $(\hat{\sigma}_t^{(i)}(\tau))_{\tau \in \mathcal{M}}$ associated with the yield curves for the different countries obtained through the YC_ATT_DE model.

We notice that larger standard deviations are detected for the yields corresponding to short time-to-maturity in contrast to the yields associated with longer maturities. This observation aligns with intuition, as shorter-term yields are more susceptible to market fluctuations, rendering them more volatile. Additionally, we note that standard deviation estimates are notably higher for specific members of the EIOPA family of yield curves, specifically those linked to Turkey, Russia, Brazil, and Mexico. This finding is plausible in light of the economic instability experienced in these countries.

Figure 7 illustrates the PICP of the YC_ATT, YC_ATT_DE, and NSS_VAR models across various time-to-maturities. Notably, the NN-based models exhibit significantly superior performance compared to the NSS_VAR model, confirming once again that the NSS model is not sufficiently flexible to capture uncertainty in certain yield curve families. On the other hand, YC_ATT and YC_ATT_DE emerge as more promising candidates to address this task, producing higher PICP for all the maturities considered. Furthermore, upon comparing YC_ATT and YC_ATT_DE, we observe that the former tends to excel for short time-to-maturity, while the latter yields higher PICPs for longer times to maturity. This finding suggests that the Gaussian distribution assumption appears to be more suitable for yields with long maturities, since the yields related to short maturity are more susceptible to market fluctuations and may be affected by some asymmetry and fat tails.

We note that the forecast term structures of standard deviations from the YC_ATT_DE are a nice by-product of this method and can be used for other quantitative risk management applications.

6.2. Transfer learning

The calibrated YC_ATT models may also provide some benefits when used on smaller datasets through the mechanism of transfer learning. Transfer learning allows for leveraging knowledge gained from solving one task and applying it to improve the performance of a different but related task. In other words, we take a model trained on one task (the source task) is repurposed or fine-tuned for a different but related task (the target task).

For this particular application, we aim to exploit a model with experience acquired in modeling and forecasting EIOPA yield curves to construct forecasting models for different families of yield curves. Transfer learning is of particular interest when a dataset of experience is available, that is too small to calibrate reliable models on. For example, with the recent implementation of IFRS 17, companies will produce portfolio specific illiquidity-adjusted yield curves. It is likely that these curves comprise too small a dataset to model; in this case transfer learning can be used.

To explore this idea, we collected a new, smaller dataset of US spot curves relating to assets with different rating levels. A visual representation of this supplementary data is presented in Figure A4 in the appendix. In this context, the set of yield curve families is defined as $\hat{\mathcal{I}} = \{AAA, AA, A, BBB, BB, B\}$. The set of maturities is represented by

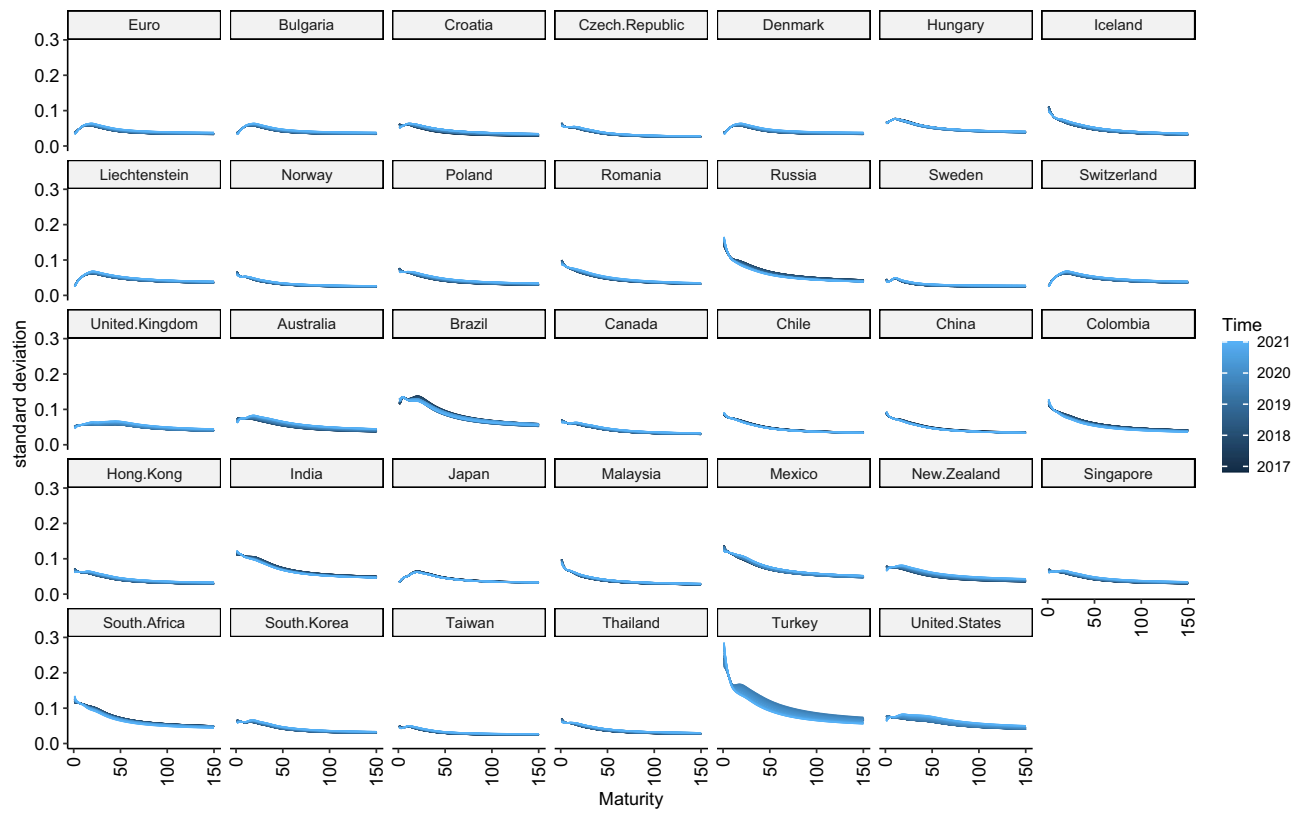


Figure 6. $(\hat{\sigma}_t^{(i)}(\tau))_{\tau \in \mathcal{M}}$ estimates associated to the yields related to the different countries.

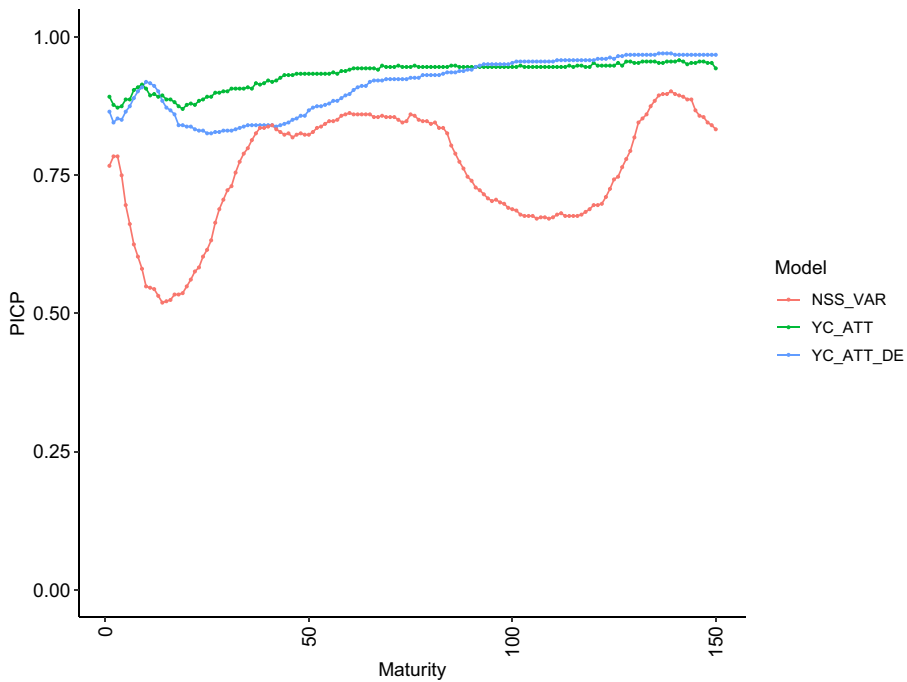


Figure 7. PICP of the NSS_VAR, YC_ATT, and YC_ATT_DE models for different time-to-maturities.

$\dot{\mathcal{M}} = \{0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30\}$, denoting $\dot{M} = |\dot{\mathcal{M}}|$ its cardinality, and the timespan \dot{T} covers monthly observations from January 2015 to October 2021. Importantly, we here have different inputs to the networks, both in terms of the number of maturities and the categorical input i . We divided the dataset into a *learning sample* and a *testing sample*, conducting a forecasting exercise for the most recent 12 months of experience in this dataset.

As noted, the two groups of curves present different number of maturities ($\dot{M} \neq M$), and directly applying the YC_ATT model to the new data becomes unfeasible. To address this issue, we equip YC_ATT model with an additional layer designed to align the US credit curves with the same dimension as the EIOPA curves. This adjustment enables us to process them using pre-calibrated attention layer of the YC_ATT model, facilitating the extraction of the relevant features $x_t^{(i)}$. Denoting as $\tilde{Y}_{t-L,t}^{(i)} \in \mathbb{R}^{(L+1) \times \dot{M}}$ the matrix of past yield curves related to the $L + 1$ previous date, we apply the (learned) mapping:

$$\dot{z} : \mathbb{R}^{(L+1) \times \dot{M}} \rightarrow \mathbb{R}^{(L+1) \times M}, \quad E_{t-L,t}^{(i)} = \dot{z}(\tilde{Y}_{t-L,t}^{(i)});$$

this used an FCN layer. Furthermore, since we are now considering a different set of yield curve families, we also introduce a new embedding layer aimed to learn a \mathbb{R} -valued representation of the elements in $\dot{\mathcal{I}}$ that is optimal with respect to the forecasting task. It is a mapping with the structure

$$\dot{e} : \{\text{AAA, AA, A, BBB, BB, B}\} \rightarrow \mathbb{R}^{q_{\dot{z}}}, \quad l_i \rightarrow \dot{e}(l_i) \stackrel{\text{set}}{=} \dot{e}^{(i)}.$$

where $q_{\dot{z}} \in \mathbb{N}$ is the hyperparameter defining the size of the embedding layer. In this case, the three output layers related to the calculation of the lower quantiles, the best estimates, and the upper quantiles have size equal to \dot{M} .

Letting $\dot{\theta}$ be the vector of NN parameters of these two new layers, the calibration process is carried out by minimizing the loss, as defined in Equation (4.4), which now also depends on the parameters $\dot{\theta}$. The objective is to learn an effective mapping that transforms the US credit curve data into the dimension of the EIOPA yield curve data in order to be processed by the pretrained attention layer of the YC_ATT model and simultaneously train the new embedding layer. In essence, we optimise the model with respect

Table 3. MSE, MAE, PICP, and MPIW of the different models considered. Bold indicates the smallest value, or, for the PICP, the value closest to $\alpha = 0.95$.

Model	MSE	MAE	PICP	MPIW
NS_AR	0.4605	0.5279	0.5243	0.9221
NS_VAR	0.3273	0.4412	0.43056	0.5796
NSS_AR	0.4833	0.5381	0.5512	1.0109
NSS_VAR	0.3416	0.4514	0.4323	0.8853
YC_ATT _{$\gamma=1$}	0.3178	0.4691	0.9003	2.3189
YC_ATT _{$\gamma=1$} (ensemble)	0.3113	0.4640	0.9019	2.3189
YC_transfer _{$\gamma=1$}	0.3152	0.4622	0.9285	2.1109
YC_transfer _{$\gamma=1$} (ensemble)	0.2262	0.3963	0.9852	2.1109

to $\hat{\theta}$ while keeping constant (or “frozen”) the parameters $\theta^{(ATT)}$ related to the key, query, and value FCNs and the attention layer:

$$\arg \min_{\hat{\theta}} \mathcal{L}(\hat{\theta}, \theta^{(ATT)}).$$

To benchmark our model with transfer learning – called YC_transfer in the below – we present the comparison against the NS and NSS models. Since we are considering a different set of data, we now extend again the comparison to all four versions of the NS and NSS models with both AR(1) and VAR(1) parameter forecasts. We also include in the comparison the model YC_ATT that is directly trained on the US credit curve data. For both NN-based models, we also investigate the use of the ensemble mechanism. In this application, we focus on the case $\gamma = 1$, that is, calibrating the best-estimates output of the model using the MAE.

Table 3 presents the performance metrics for all the models across the four measures. We note that NS_VAR and NSS_VAR models outperform their counterparts that are based on independent AR models, in terms of point forecasts. However, it is noteworthy that all four models exhibit poor performance in terms of PICP, indicating a limited ability to capture uncertainty surrounding future yields. When examining NN-based models, we also note the ensemble mechanism consistently enhances the results of both the YC_ATT _{$\gamma=1$} and YC_transfer _{$\gamma=1$} models. The most accurate outcomes are obtained with the YC_transfer _{$\gamma=1$} (ensemble), which produces superior performance in terms of MSE, MAE, and PICP. Figure 8 illustrates the point and interval forecasts of the YC_ATT _{$\gamma=1$} and the YC_transfer _{$\gamma=1$} models on three distinct dates: the starting date, the middle date, and the last date of the forecasting horizon. Notably, the width of the forecast interval expands as we transition from yield curves associated with high AAA ratings to those with lower B ratings. This evidence works for all three dates. This trend aligns with expectations, as greater uncertainty is logically anticipated in yields linked to lower-rated companies. Moreover, both models exhibit commendable performance in predicting yields for reliable ratings (BBB, A, AA, AAA). The transfer model, in particular, demonstrates enhanced coverage for the lower-rated categories (B and BB), where more uncertainty is expected. A plausible explanation for this observation is that the transfer model adeptly captures tension by leveraging insights gained from EIOPA data, featuring yield curves marked by substantial volatility.

7. Conclusions

The accurate modeling of the yield curves is crucial in insurance and finance for several reasons, playing a fundamental role in risk management, investment decision-making, and asset-liabilities evaluation. This paper has advanced the field by developing deep learning models to describe the dynamics of multiple yield curves associated with diverse credit qualities or countries simultaneously. We have confirmed the intrinsic ability of these models to effectively describe large-dimensional time series data and model nonlinearity inherent in yield curve dynamics. Our study shows that these models outperform other

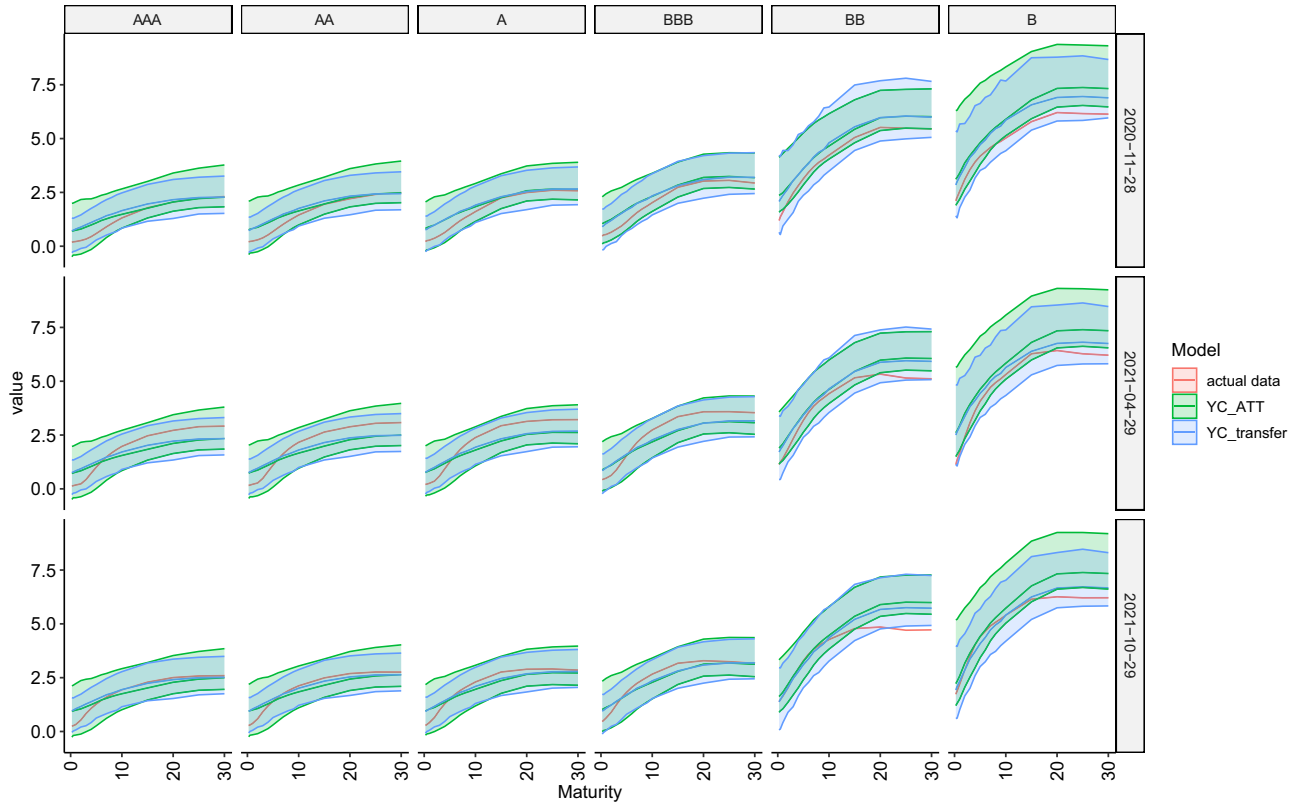


Figure 8. Point and interval forecasts for the US credit curves generated by the YC_ATT and YC_transfer models for the central, the middle and the final date of forecasting period.

well-known models such as the dynamic version of the Nelson and Siegel (Nelson and Siegel, 1987; Diebold and Li, 2006) and the related Svensson extension (Svensson, 1994) in the multiple yield curve modeling and forecasting tasks. We performed several numerical experiments on the data provided by the EIOPA. Although different kinds of neural network architecture have been investigated, we discover the most promising results have been obtained by using the self-attention mechanism, which has proven successful in natural language processing. Furthermore, we also investigate techniques for quantifying the uncertainty around predictions, a critical yet under-explored in the existing literature. We explored the use of nonparametric quantile regression and designed an architecture specifically designed to avoid quantile crossing issues. The numerical analysis of the performance, conducted in terms of PICP and MPIW, shows the effectiveness of the proposed approach. We finally discuss two possible extensions and variants of the proposed model. The first one considers using the DE method for measuring uncertainty in forecasts. This approach, which requires the assumption of heteroscedastic Gaussian distributions of the response, appears to be promising, especially in describing the dynamics of yields with long time-to-maturities; the DE approach also produces a term structure of forecast volatility, which is a useful by-product. The second extension uses a transfer learning mechanism that could be useful to exploit the experiences gained in modeling the EIOPA yield curves to improve the performance of a model related to a different set of yield curves. A numerical illustration of this approach is conducted considering the US credit curve data with different credit qualities (ratings). We show that this approach allows for improved forecasting performance, especially in terms of prediction intervals, when the data we are considering are subject to more uncertainty.

In future studies, we plan to explore the use of explainable deep learning techniques, in particular, the LocalGLMnet model introduced in Richman and Wüthrich (2023), to model effectively the uncertainty inherent in future yield predictions in an interpretable manner. Initially designed for expected values, an extension to quantile modeling is interesting but also challenging due to the intricate issue of quantile crossing. Moreover, we would like to model jointly interest rates and other market variables using a similar model. Finally, our future research agenda extends to the investigation of other potential applications of attention and transfer models within the insurance domain. Specifically, we aim to explore their efficacy in nonlife insurance fields, such as frequency-severity modeling.

Acknowledgments. The authors thank the two anonymous referees whose comments helped to improve the manuscript. The authors acknowledge the financial support for this project “Multiple Yield Curve modeling and Forecasting using Deep Learning” through a Research Grant of the Life Section (a Section of the International Actuarial Association). The authors are grateful to the staff members at Old Mutual who provided the US credit curves.

References

- Aljinović, Z. and Poklepović, T. (2013) Neural networks and vector autoregressive model in forecasting yield curve. In *The 6th International Conference on Information Technology (ICIT)*, pp. 1–8.
- Atkins, P.J. and Cummins, M. (2023) Improved scalability and risk factor proxying with a two-step principal component analysis for multi-curve modelling. *European Journal of Operational Research*, **304**(3), 1331–1348.
- Barigou, K. and Delong, L. (2022) Pricing equity-linked life insurance contracts with multiple risk factors by neural networks. *Journal of Computational and Applied Mathematics*, **404**, 113922.
- Bengio, Y., Courville, A. and Vincent, P. (2013) Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(8), 1798–1828.
- Bliss, R.R. (1996) Testing term structure estimation methods. Tech. rep., Working Paper.
- Bowsher, C.G. and Meeks, R. (2008) The dynamics of economic functions: Modeling and forecasting the yield curve. *Journal of the American Statistical Association*, **103**(484), 1419–1437.
- Cuchiero, C., Fontana, C. and Gnoatto, A. (2016) A general HJM framework for multiple yield curve modelling. *Finance and Stochastics*, **20**, 267–320.
- De Rezende, R.B. and Ferreira, M.S. (2013) Modeling and forecasting the yield curve by an extended Nelson-Siegel class of models: A quantile autoregression approach. *Journal of Forecasting*, **32**(2), 111–123.
- Diebold, F.X. and Li, C. (2006) Forecasting the term structure of government bond yields. *Journal of Econometrics*, **130**(2), 337–364.
- Diebold, F.X. and Rudebusch, G.D. (2013) *Yield Curve Modeling and Forecasting: The Dynamic Nelson-Siegel Approach*. Princeton, NJ: Princeton University Press.

- Diebold, F.X., Rudebusch, G.D. and Aruoba, S.B. (2006) The macroeconomy and the yield curve: A dynamic latent factor approach. *Journal of Econometrics*, **131**(1-2), 309–338.
- Duffee, G.R. (2002) Term premia and interest rate forecasts in affine models. *The Journal of Finance*, **57**(1), 405–443.
- Europeen, G.C.A. (2012) Market consistency.
- Fama, E.F. and Bliss, R.R. (1987) The information in long-maturity forward rates. *The American Economic Review*, **77**(4), 680–692.
- Gabrielli, A. (2020) A neural network boosted double overdispersed Poisson claims reserving model. *ASTIN Bulletin: The Journal of the IAA*, **50**(1), 25–60.
- Gerhart, C. and Lütkebohmert, E. (2020) Empirical analysis and forecasting of multiple yield curves. *Insurance: Mathematics and Economics*, **95**, 59–78.
- Gerhart, C., Lütkebohmert, E. and Weber, M. (2019) Robust forecasting of multiple yield curves. In *Theory and Applications of Time Series Analysis: Selected Contributions from ITISE 2018*, **5**, pp. 187–202. Springer.
- Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. Cambridge, MA: MIT Press.
- Grbac, Z. and Runggaldier, W.J. (2015) *Interest Rate Modeling: Post-crisis Challenges and Approaches*, Vol. **157**. Cham, Switzerland: Springer.
- Guo, C. and Berkahn, F. (2016) Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737.
- Härdle, W.K. and Majer, P. (2016) Yield curve modeling and forecasting using semiparametric factor dynamics. *The European Journal of Finance*, **22**(12), 1109–1129.
- Heath, D., Jarrow, R. and Morton, A. (1992) Bond pricing and the term structure of interest rates: A new methodology for contingent claims valuation. *Econometrica: Journal of the Econometric Society*, **60**(1), 77–105.
- Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.
- Hull, J. and White, A. (1990) Pricing interest-rate-derivative securities. *The Review of Financial Studies*, **3**(4), 573–592.
- Hyndman, R.J. and Khandakar, Y. (2008) Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, **27**, 1–22.
- Kauffmann, P.C., Takada, H.H., Terada, A.T. and Stern, J.M. (2022) Learning forecast-efficient yield curve factor decompositions with neural networks. *Econometrics*, **10**(2), 15.
- Kemp, M. (2009) *Market Consistency: Model Calibration in Imperfect Markets*. Hoboken, NJ: John Wiley & Sons.
- Koenker, R. and Bassett Jr, G. (1978) Regression quantiles. *Econometrica: Journal of the Econometric Society*, **46**(1), 33–50.
- Kuo, K. and Richman, R. (2021) Embeddings and attention in predictive modeling. arXiv preprint arXiv:2104.03545.
- Lakshminarayanan, B., Pritzel, A. and Blundell, C. (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30*.
- Nelson, C.R. and Siegel, A.F. (1987) Parsimonious modeling of yield curves. *Journal of Business*, **60**(4), 473–489.
- Nix, D.A. and Weigend, A.S. (1994) Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, Vol. 1, pp. 55–60. IEEE.
- Noll, A., Salzmann, R. and Wüthrich, M.V. (2020) Case study: French motor third-party liability claims. Available at SSRN 3164764.
- Nunes, M., Gerding, E., McGroarty, F. and Niranjan, M. (2019) A comparison of multitask and single task learning with artificial neural networks for yield curve forecasting. *Expert Systems with Applications*, **119**, 362–375.
- Perla, F., Richman, R., Scognamiglio, S. and Wüthrich, M.V. (2021) Time-series forecasting of mortality rates using deep learning. *Scandinavian Actuarial Journal*, **2021**(7), 572–598.
- Piazzesi, M. (2010) Affine term structure models. In *Handbook of Financial Econometrics: Tools and Techniques*, pp. 691–766. Elsevier.
- Redfern, D. and McLean, D. (2014) Principal component analysis for yield curve modelling. *Enterprise Risk Solutions*.
- Richman, R. (2021a) Ai in actuarial science—a review of recent advances—part 1. *Annals of Actuarial Science*, **15**(2), 207–229.
- Richman, R. (2021b) Ai in actuarial science—a review of recent advances—part 2. *Annals of Actuarial Science*, **15**(2), 230–258.
- Richman, R. and Wüthrich, M.V. (2023) LocalGLMnet: Interpretable deep learning for tabular data. *Scandinavian Actuarial Journal*, **2023**(1), 71–95.
- Scognamiglio, S. (2022) Calibrating the Lee-Carter and the Poisson Lee-Carter models via neural networks. *ASTIN Bulletin: The Journal of the IAA*, **52**(2), 519–561.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**(1), 1929–1958.
- Svensson, L.E. (1994) Estimating and interpreting forward interest rates: Sweden 1992–1994.
- Teichmann, J. and Wüthrich, M.V. (2016) Consistent yield curve prediction. *ASTIN Bulletin: The Journal of the IAA*, **46**(2), 191–224.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017) Attention is all you need. In *Advances in Neural Information Processing Systems 30*.
- Venter, G.G. (2004) Testing distributions of stochastically generated yield curves. *ASTIN Bulletin: The Journal of the IAA*, **34**(1), 229–247.
- Wüthrich, M.V. and Merz, M. (2023) *Statistical Foundations of Actuarial Learning and its Applications*. Cham, Switzerland: Springer Nature.
- Yasuoka, T. (2018) *Interest Rate Modeling for Risk Management: Market Price of Interest Rate Risk*, Vol. **1**. Bentham Science Publishers.

Appendix

A Appendix: Proof, data, and plots

A.1 Calibration of the Nelson–Siegel and Nelson–Siegel–Svensson models

The procedure introduced in Diebold and Li (2006) was originally designed for modeling the evolution over time of a single family of yield curves. Since we are considering multiple families, we apply the procedure to each family of curves individually. For a chosen observation date, we fit the NS model on the yields of the different dates. The method sets $\lambda_t^{(i)}$ to a constant value $\lambda^{(i)}$ for all $t \in \mathcal{T}$. In our experiments, we set this constant equal to the reciprocal of the average maturity. The remaining parameters are estimated date by date by solving a sequence of independent linear optimization problems:

$$\hat{\beta}_t^{(i)} = \arg \min_{\beta_{0,t}^{(i)}, \beta_{1,t}^{(i)}, \beta_{2,t}^{(i)}} \sum_{\tau \in \mathcal{M}} \left(y_t^{(i)}(\tau) - \beta_{0,t}^{(i)} - \beta_{1,t}^{(i)} \left(\frac{1 - e^{-\lambda^{(i)}\tau}}{\lambda^{(i)}\tau} \right) - \beta_{2,t}^{(i)} \left(\frac{1 - e^{-\lambda^{(i)}\tau}}{\lambda^{(i)}\tau} - e^{-\lambda^{(i)}\tau} \right) \right)^2, \quad \forall t \in \mathcal{T}.$$

This procedure results in the time series of the estimates $\hat{b}_t^{(i)} = (\hat{\beta}_{0,t}^{(i)}, \hat{\beta}_{1,t}^{(i)}, \hat{\beta}_{2,t}^{(i)})$ of the latent factors for each family of curves $i \in \mathcal{I}$. To formulate the forecasts of the future yield curves, we need to extrapolate the future values of these latent factors. Both the models in (2.1) and (2.2) can be considered for this purpose. The parameters of these time series models can be estimated individually for each family of yield curves using the ordinary least squares (OLS) estimator, leveraging standard statistical packages. We use the R package “forecast” (see Hyndman and Khandakar, 2008).

Figure A1 illustrates, for each family of curves, the estimates of the latent factors and their associated forecasts using both AR(1) and VAR(1) models, based on the EIOPA data analyzed in the numerical experiments outlined in Section 5.

For the NSS models, we adopt a similar methodology. Specifically, we set $\lambda_{1,t}^{(i)} = \lambda_1^{(i)}$ and $\lambda_{1,t}^{(i)} = \lambda_2^{(i)}$, respectively, equal to the first and third quartiles of the distribution of the maturities. The remaining parameters are optimized through the resolution of the following equations:

$$\arg \min_{\beta_{0,t}^{(i)}, \beta_{1,t}^{(i)}, \beta_{2,t}^{(i)}, \beta_{3,t}^{(i)}} \left(y_t^{(i)}(\tau) - \beta_{0,t}^{(i)} - \beta_{1,t}^{(i)} \left(\frac{1 - e^{-\lambda_{1,t}^{(i)}\tau}}{\lambda_{1,t}^{(i)}\tau} \right) - \beta_{2,t}^{(i)} \left(\frac{1 - e^{-\lambda_{1,t}^{(i)}\tau}}{\lambda_{1,t}^{(i)}\tau} - e^{-\lambda_{1,t}^{(i)}\tau} \right) - \beta_{3,t}^{(i)} \left(\frac{1 - e^{-\lambda_{2,t}^{(i)}\tau}}{\lambda_{2,t}^{(i)}\tau} - e^{-\lambda_{2,t}^{(i)}\tau} \right) \right).$$

Also in this case, the future values of these parameters can be extrapolated by using univariate or multivariate time series models.

A.2 Proof: average MPIW and MPIW of the averaging ensemble predictions

Here, we show that the average MPIW across different training attempts coincides with the MPIW of the averaging ensemble predictions of the K models.

We denote

- $y_{t,\alpha/2}^{(i,k)}(\tau)$ lower quantile estimate at level $\alpha/2$ of the k -th model,
- $y_{t,1-\alpha/2}^{(i,k)}(\tau)$ upper quantile estimate at level $1 - \alpha/2$ of the k -th model,
- $n = |\mathcal{M}| \times |\mathcal{I}| \times |\mathcal{T}|$.

The MPIW of the k -th model on the different data points is

$$\text{MPIW}_k = \frac{1}{n} \sum_t \sum_i \sum_{\tau} \left(y_{t,1-\alpha/2}^{(i,k)}(\tau) - y_{t,\alpha/2}^{(i,k)}(\tau) \right).$$

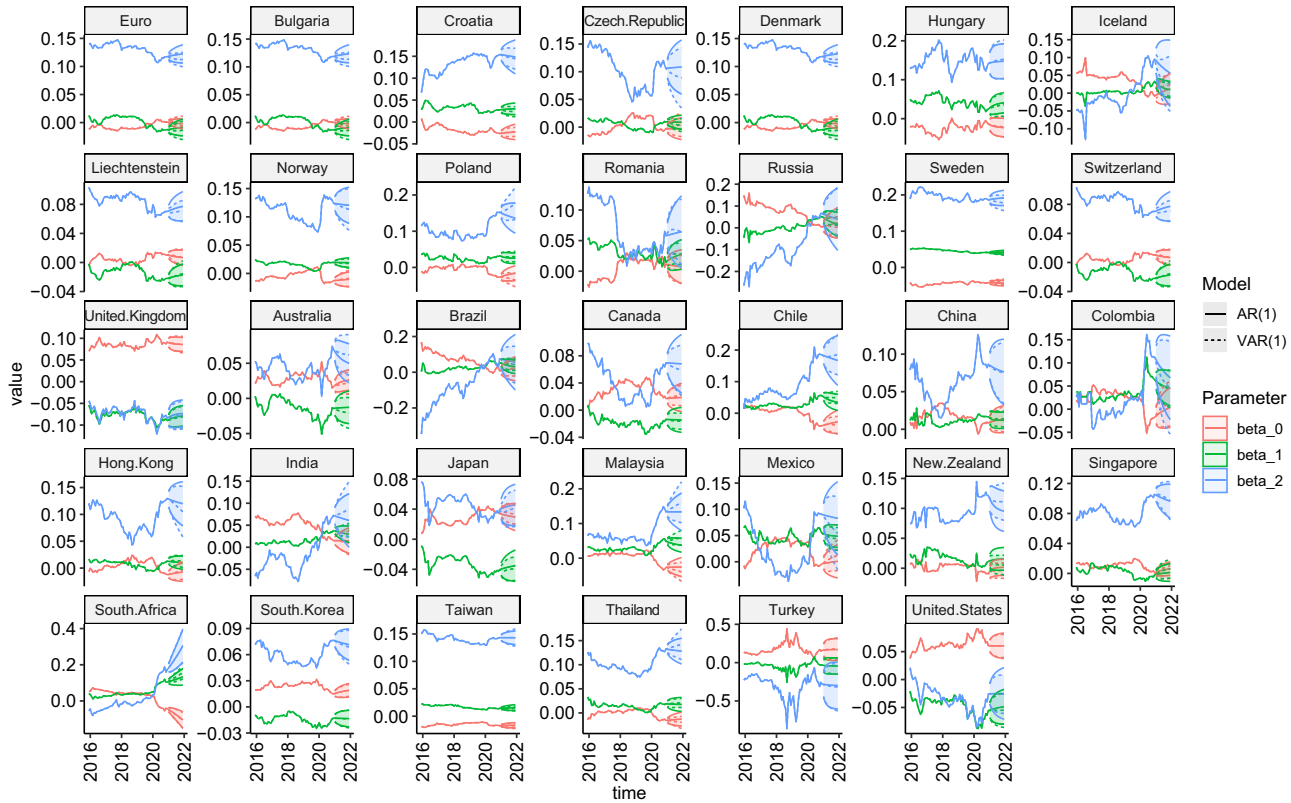


Figure A1. Time series of the estimates and the related forecast of the latent factors $\hat{\beta}_t^{(i)}$ for the different families of yield curves.

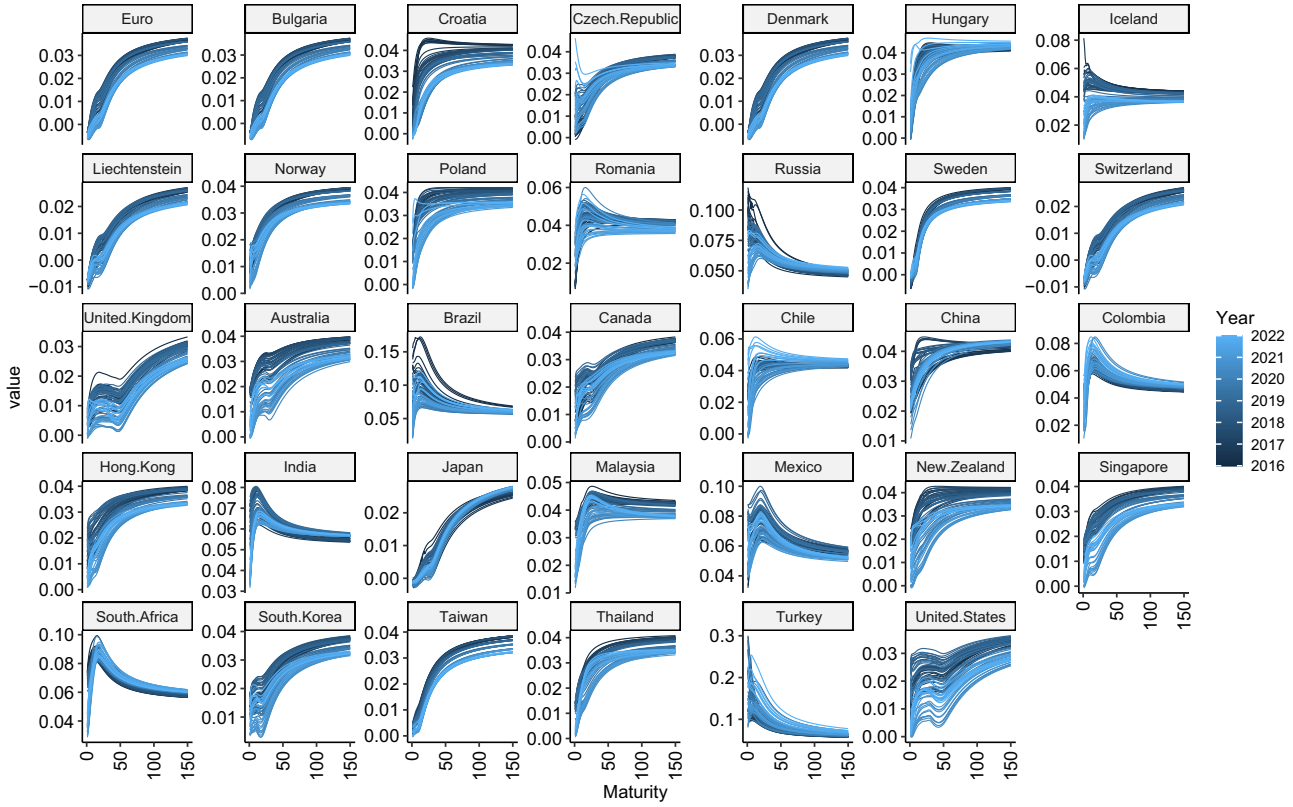


Figure A2. Risk-free interest rate term structures derived from government bonds of different countries; observation period spans from December 2015 to December 2021.

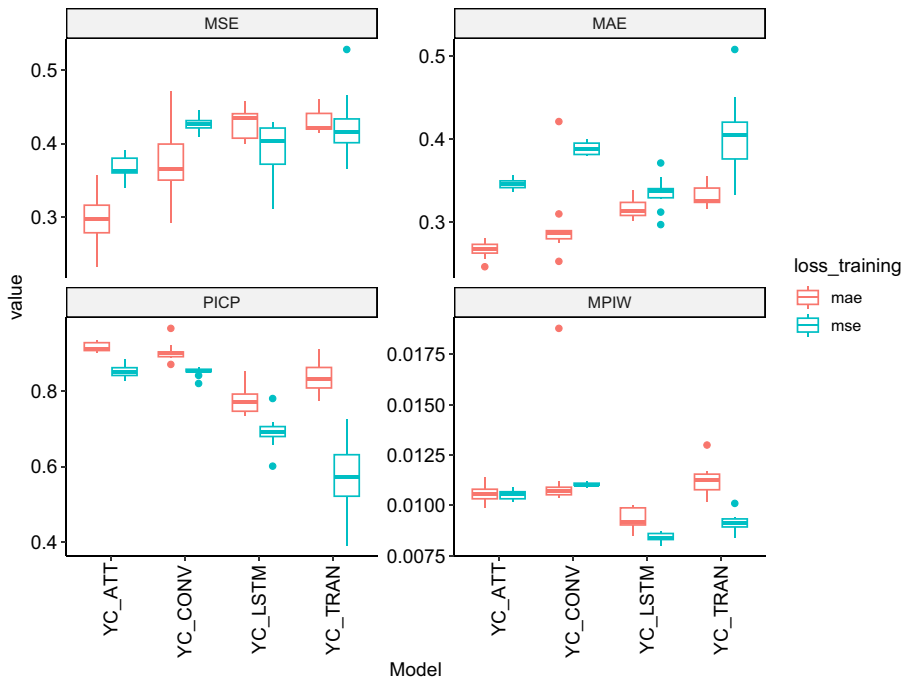


Figure A3. Boxplot of the out-of-sample MSE, MAE, PICP, and MPIW of the different models on ten runs; the MSE values are multiplied 10^5 , the MAE values are multiplied by 10^2 .

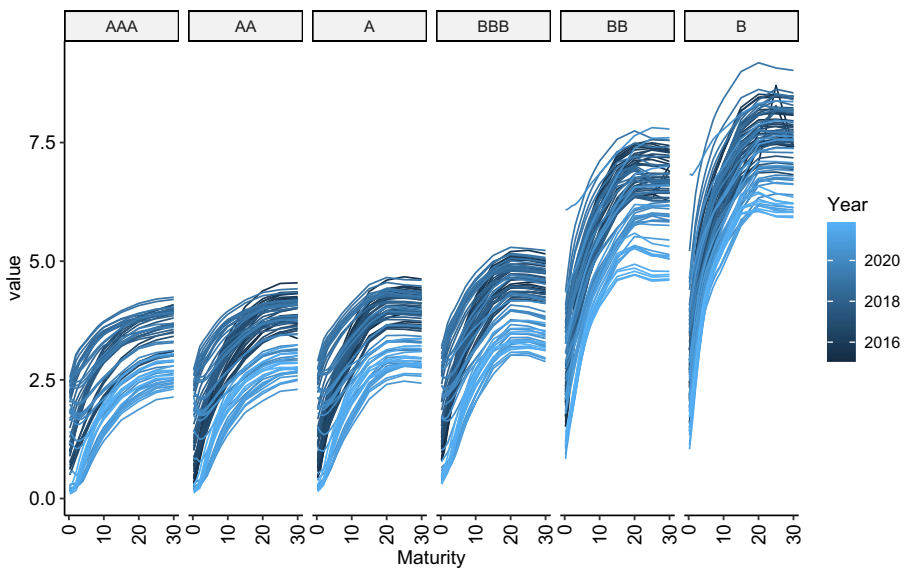


Figure A4. US credit curves related to different rating qualities.

The average MPIW over K training attempts is

$$MPIW_{average} = \frac{1}{K} \sum_{k=1}^K MPIW_k = \frac{1}{K * n} \sum_{k=1}^K \sum_t \sum_i \sum_{\tau} \left(y_{t,1-\alpha/2}^{(i,k)}(\tau) - y_{t,\alpha/2}^{(i,k)}(\tau) \right),$$

the MPIW of the ensemble predictions is

$$\begin{aligned} \text{MPIW}_{ensemble} &= \frac{1}{n} \sum_t \sum_i \sum_\tau \left(\frac{1}{K} \sum_{k=1}^K y_{t,1-\alpha/2}^{(i,k)}(\tau) - \frac{1}{K} \sum_{k=1}^K y_{t,\alpha/2}^{(i,k)}(\tau) \right) \\ &= \frac{1}{K * n} \sum_{k=1}^K \sum_t \sum_i \sum_\tau \left(y_{t,1-\alpha/2}^{(i,k)}(\tau) - y_{t,\alpha/2}^{(i,k)}(\tau) \right). \end{aligned}$$

Then, we can conclude that

$$\text{MPIW}_{average} = \text{MPIW}_{ensemble}.$$

A.3 R Code for the YC_ATT model and its variants

A.4 Data and other plots

Listing 1: Keras Code for defining the YC_ATT model and its variants.

```

1 # mod_type (string) is a variable defining the type of Neural Network layer that processes
2 # the past yield curves.
3 # u (integer) is an hyperparameter defining the size of the intermediate layers.
4
5 k_clear_session()
6 yields <- layer_input(shape = c(10,150), dtype = 'float32', name = 'yields')
7 Country <- layer_input(shape = c(1), dtype = 'int32', name = 'Country')
8 Country_embed=Country%>%
9   layer_embedding(input_dim = 34, output_dim = 5) %>%
10  layer_flatten(name= 'Country_embed')
11
12 if (mod_type == "YC_ATT"){
13   #Pure Self-Attention model
14   query = yields %>% time_distributed(layer_dense(units =u, activation = "tanh"))
15   key = yields %>% time_distributed(layer_dense(units =u, activation = "tanh"))
16   value = yields %>% time_distributed(layer_dense(units =u, activation = "tanh"))
17   features = list(query, key, value) %>% layer_attention(use_scale = T, trainable = T) %>%
18   layer_flatten( name = "features") %>%layer_dropout(0.5)
19 }else if (mod_type == "YC_CONV" ){
20   #Pure convolutional model
21   features = yields %>% layer_conv_1d(filter =u, kernel_size =1, activation = "tanh",
22   padding = "causal", input_shape =c(10,150))%>%
23   layer_flatten()%>%layer_dropout(0.5)
24 }else if (mod_type == "YC_LSTM" ){
25   #Pure LSTM model
26   features = yields %>% layer_lstm(units=u, activation = "tanh", return_sequences = T,
27   input_shape =c(10,d))%>%
28   layer_flatten()%>%layer_dropout(0.5)
29 }else if (mod_type == "YC_TRAN" ){
30   #Transformer model
31   intermediate = yields %>% layer_conv_1d(filter =u, kernel_size =3, activation = 'tanh',
32   padding = "causal", input_shape =c(10,150))%>%layer_dropout(0.2)
33   query = intermediate %>% time_distributed(layer_dense(units = u, activation = "tanh"))
34   key = intermediate %>% time_distributed(layer_dense(units = u, activation = "tanh"))
35   value = intermediate %>% time_distributed(layer_dense(units = u, activation = "tanh"))
36   attended = list(query, key, value) %>% layer_attention(use_scale = T, trainable = T)
37   for_ln = list(intermediate, attended) %>% layer_add()
38   attended2 = for_ln %>% time_distributed(layer_dense(units = u*4, activation = "tanh")) %>%
39   time_distributed(layer_dense(units = u, activation = "tanh"))
40   features = list(attended2, for_ln) %>% layer_add() %>% layer_flatten() %>%layer_dropout(0.5)
41 }
42
43 central_comp = features %>%list(Country_embed)%>%
44   layer_concatenate(name = "central_comp") %>%
45   layer_dense(units = d, activation = 'sigmoid') %>%
46   layer_reshape(c(1,d),name = 'forecast_yields')
47 lower_comp = features %>%list(Country_embed)%>%
48   layer_concatenate() %>%
49   layer_dense(units = d, activation = 'sigmoid') %>% layer_lambda(function(x) -x) %>%
50   layer_reshape(c(1,d),name = 'lower_comp')
51 upper_comp = features %>%list(Country_embed)%>%
52   layer_concatenate() %>%
53   layer_dense(units = d, activation = 'sigmoid') %>%
54   layer_reshape(c(1,d),name = 'upper_comp')
55
56
57 upper = central_comp %>% list(upper_comp) %>% layer_add()%>% layer_reshape(c(1,d),name = 'upper')
58 lower = central_comp %>% list(lower_comp) %>% layer_add()%>% layer_reshape(c(1,d),name = 'lower')
59
60
61 model <- keras_model(inputs = list(yields,Country), outputs = c(central_comp, lower, upper))

```