Deep Learning Credit Risk Modeling

Gerardo Manzo and Xiao Qiao

Gerardo Manzo
is a quantitative researcher and portfolio manager at Kepos Capital in New York, NY.
gm.gerardomanzo@gmail.com

Xiao Qiao
is an assistant professor at the City University of Hong Kong and a member of the Hong Kong Institute for Data Science in Kowloon Tong, Hong Kong.
xiaoqiao@cityu.edu.hk

KEY FINDINGS

- Neural networks can approximate solutions to credit risk models, precisely capturing the relationship between model inputs and credit spreads.
- Compared to standard techniques, the approximate solutions are more computationally efficient.
- Neural networks can be used to accurately calibrate structural and reduced-form models of credit risk.

ABSTRACT

This article demonstrates how deep learning can be used to price and calibrate models of credit risk. Deep neural networks can learn structural and reduced-form models with high degrees of accuracy. For complex credit risk models with no closed-form solutions available, deep learning offers a conceptually simple and more efficient alternative solution. This article proposes an approach that combines deep learning with the unscented Kalman filter to calibrate credit risk models based on historical data; this strategy attains an in-sample R-squared of 98.5% for the reduced-form model and 95% for the structural model.

The increasing size and complexity of credit derivatives markets[1] pose a serious challenge for researchers seeking to accurately quantify credit risk (i.e., the risk that an issuer of a debt obligation defaults). The recent proliferation of complex credit risk models reflects the sophistication of the markets as well as the need for accurately capturing the risk of default. This complexity leads to more computationally intensive solutions, often involving numerical methods. In this article, we apply deep learning, or deep neural networks, to credit risk modeling. Deep learning models can accurately learn sophisticated credit risk models and then can be used for calibration to historical data.

Deep neural networks represent an important machine learning technique that is widely known for its strong predictive power and broad applications. Such deep learning has been successfully applied to speech recognition, natural language processing, computer vision, and other areas. Its use in economics and finance is attracting more interest among researchers. Our main contribution in this article is introducing deep learning to the credit risk literature.

[1] As of September 2019, the International Swap and Derivatives Association (ISDA) reports that market activity in the credit default swap (CDS) market averaged about $700 billion per quarter over the past 13 quarters (to June 2019). More important, the activity in the CDS indexes, mainly the CDX and iTraxx indexes, jumped from $4 trillion in 2017 to $5.8 trillion in 2019.

We start with an overview of the main credit risk models, which can be categorized into structural models and reduced-form models. Structural models provide a direct link between the default event and the capital structure of a firm. The seminal work of Merton (1974) sets the foundation for this category. In Merton (1974), the asset growth of a firm follows a stochastic process with normally distributed shocks. A firm defaults when its asset value falls below the debt face value at maturity. We then review Merton (1976) and Kou (2002), two models designed to capture the empirical observation that asset returns are fat-tailed. These models add stochastic jump risk to the Merton (1974) model: Merton (1976) includes log-normal jumps whereas Kou (2002) institutes double stochastic jumps. These three models are all characterized by static parameters governing the asset growth process. Stochastic state variables could be introduced for further generalization. We review a more general structural model that accommodates multiple stochastic volatilities and stochastic jumps, building on the findings of Duan (1999), Yan (2011), and Du, Elkamhi, and Ericsson (2019), among others. Our general credit risk model subsumes several popular models, including Heston (1993), one stochastic jump, two stochastic volatilities, one stochastic volatility and one stochastic jump, and two stochastic volatilities and one stochastic jump.

In reduced-form models, default risk is modeled as a statistical process. These models allow the researcher to price credit risk for entities with capital structures that cannot be easily defined, such as a country's default risk. In this article, we focus on the reduced-form model of Pan and Singleton (2008), with default intensity that follows a log-normal stochastic process.

After a review of credit risk models, we overview neural networks. We first describe a simple neural network with a single hidden layer and discuss its relationship to the familiar ordinary least squares regression. We then explain deep learning, which extends the neural networks approach to incorporate multiple hidden layers. We discuss the empirical choices that researchers must make when working with neural networks, including the model architecture, training and validation sets, activation function, batch size, and number of epochs. By using a simple but detailed explanation of practical implementation issues, we hope to facilitate the application of deep learning to economics and finance research.

How does deep learning relate to credit risk modeling? A credit risk model captures the relationship between model parameters and credit spreads in a pricing function. If we view model parameters as the inputs and credit spreads as the outputs, the pricing function computes the outputs given a set of inputs. Deep learning can be used to accurately approximate the function that maps inputs to outputs.

Thus, we can teach deep learning models the relationship between model parameters and credit spreads, replicating the complex credit risk models. We use simulations to generate artificial data for each credit risk model. We draw 50,000 combinations of model parameters. For each parameter combination, we consider credit instruments at five maturities: 1 year, 3 years, 5 years, 7 years, and 10 years. We then compute the credit spreads associated with each set of parameters and maturity. Our simulated data are composed of 250,000 pairs of parameters and credit spreads for each model. We employ these artificial observations to train and evaluate deep learning models.

For each model, we randomly split the simulated data into observations used for training and testing: 95% of the 250,000 observations for the training set and the remaining 5% for the test set. The deep learning models have either two or three hidden layers of 100 nodes, depending on the complexity of the credit risk models. We use the rectified linear unit (ReLU) as the activation function, a batch size of 1,024 observations, and 500 epochs to train. We provide a more detailed discussion of those practical choices later in this article.

Deep learning can accurately capture the pricing relationship between model parameters and credit spreads. We evaluate the performance of deep learning models on the test set, finding that the predicted spreads from deep learning approximate the actual spreads very closely; R-squared for the test set is close to 100% for all models. For example, for the Heston (1993) model, R-squared is 99.98% for the 1-year maturity and 99.99% for the 10-year maturity. Similarly, for the Pan and Singleton (2008) model, R-squared is 99.97% for both the 1-year and 10-year maturities. A high R-squared indicates that the deep learning models are approximating the relationship between model parameters and credit spreads with a high degree of accuracy.

Solutions to credit risk models can be computationally intensive. Aside from the Merton (1974) model, the more complex structural models do not have closed-form solutions. Therefore, pricing credit risk in a structural model requires numerical integration. Furthermore, the assumption of a log-normal default intensity in the reduced-form model of Pan and Singleton (2008) makes default probability calculations computationally expensive. Our proposed approach—using deep learning to price credit risk—offers an alternative to these numerically expensive methods. After learning the relationship embedded in each credit risk model, the deep learning models can quickly calculate credit spreads for an arbitrary set of model parameters.

To compare the speed of deep learning models against the pricing functions, we measure the time required to build a full term structure of credit spreads. We observe a tradeoff between precision and speed for the pricing functions: for more precise calculations, the numerical procedures take longer. For high precision, the pricing functions can take three to four seconds to build a full term structure, but for lower precision, the pricing functions need only 0.02 seconds. In comparison, our deep learning approach produces a full term structure in just 0.001 seconds, independent of the complexity of the underlying model, effectively resolving the tradeoff between precision and speed. In relative terms, our deep learning approach generates spreads about 100 to 240 times faster than the actual pricing function.

Deep learning can accurately and efficiently produce credit spreads for specified model parameters. An important goal of credit risk modeling is to capture and explain the observed time series and cross-maturity variations in credit spreads. In the final part of the article, we investigate whether we can apply deep learning models to calibration. In a typical calibration, the researcher chooses a set of parameters that minimize the pricing errors between historical spreads and model spreads. We propose combining deep neural networks with the unscented Kalman filter (UKF), an approach we call NN-UKF. We conclude that this approach is effective in recovering optimized model parameters. We calibrate the structural Heston (1993) model with book value and with market value of leverage and the reduced-form model of Pan and Singleton (2008); the in-sample R-squared values are 89%, 95%, and 98.5%, respectively.

Our results demonstrate that deep neural networks can accurately learn pricing functions and efficiently calibrate models to historical data for both structural and reduced-form models of credit risk. Importantly, the deep learning approach does not rely on the numerically intensive pricing techniques commonly used in the credit risk literature. Once the deep neural network is trained, it can repeatedly and quickly generate new term structures of credit spreads for different input parameters. In this sense, our deep learning approach can save significant time and computing resources compared to traditional pricing functions, which becomes particularly relevant in real-time calibrations to historical data.

LITERATURE REVIEW AND CONTRIBUTION

Our primary contribution is introducing deep learning to the credit risk literature. By proposing deep neural networks as highly accurate approximations for complex credit risk models, we give academics and practitioners a new tool that simplifies the testing and calibration of sophisticated models of default risk.

Our application of deep learning to structural credit models follows the recent growing literature that applies neural networks to options pricing. Over the past decade, an extensive body of literature developed around the application of machine learning to options pricing. Ruf and Wang (2019) comprehensively review nonparametric methods for options pricing. Liu, Oosterlee, and Bohte (2019) propose an artificial neural network to approximate the Black and Scholes (1973) model and the stochastic volatility model of Heston (1993). Our approach is also based on artificial neural networks, but we consider a much broader set of models in the credit risk space, with different levels of complexity. Liu et al. (2019) and Horvath, Muguruza, and Tomas (2019) introduce a method for performing calibration by using neural networks, and they calibrate the Heston model, the Bates model, and the Bergomi model. We advance their approach, substituting the complex pricing functions with trained neural networks, but we also propose a calibration of the full panel of historical data rather than fitting the model each day to the cross-section. Of course, fitting a model of credit risk cross-sectionally is not the same as fitting the options surface. Usually, only a few data points in the term structure are available because of liquidity, and credit spreads do not include a moneyness dimension. Compared to options prices, the cross-sectional dimension of credit risk is much smaller. Our approach holds the model parameters static while allowing stochastic state variables—and thus also serves as an additional test of the capability of deep learning pricing models to calibrate the time variation as well as the cross-maturity variation. We are the first to apply deep learning to both structural models and reduced-form models of credit risk.

Our calibration exercise provides an additional contribution to the literature that centers not only on the modeling aspect but also on the application of the models to historical data.

Researchers often try to understand historical credit risk dynamics and then extrapolate default predictions as part of a trading or risk management system. The proposed NN-UKF approach facilitates such applications. After training the appropriate deep learning model, a researcher can perform multiple calibrations, using the trained model in a faster and more computationally efficient manner. When analyzing complex credit risk models without closed-form solutions, our approach can reduce both the time and computing resources expended for calibration.

More generally, our work fits into the literature on applying machine learning to empirical finance. Early work by Hutchinson, Lo, and Poggio (1994) employs a nonparametric method to estimate the pricing formula for derivatives. More recently, Gu, Kelly, and Xiu (2020) explore a comprehensive set of machine learning techniques for cross-sectional asset pricing. Feng, Polson, and Xu (2019) describe a deep learning approach to build portfolios from firm-specific characteristics. Along the same lines, Gu, Kelly, and Xiu (2019) model factor exposures as a flexible nonlinear function of characteristics, using autoencoder neural networks. Feng, Giglio, and Xiu (2020) propose a model selection method to evaluate new factors. We contribute to this broadening body of literature by applying deep learning to credit risk models.

The article is structured as follows. In the next section, we summarize some of the important credit risk models. We then briefly review neural networks and discuss our empirical choices. We demonstrate that deep learning can accurately predict credit

spreads and show how deep learning can be used in calibrating credit risk models. In the last section, we offer several conclusions.

CREDIT RISK MODELING

A credit risk model captures the probability of default over a prespecified time horizon. Consider a firm i that issues a \$1 risky zero-coupon bond due in T years. The bond price at time t, $P_{i,t}(T)$, therefore is the probability-weighted discounted cash-flow at maturity. The bondholder either receives the full face value if the issuer does not default or collects the recovery amount RR, $\$0 \leq RR \leq \$1$ if the issuer defaults before maturity, as expressed by

$$\begin{aligned} P_{i,t}(T) &= D \times [1\$ \times (1 - CQDF_{i,t}) + RR \times CQDF_{i,t}] \\ &= D \times [1 - LGD \times CQDF_{i,t}] \end{aligned} \tag{1}$$

where $D = e^{-r_t T}$ is the continuous discount function with risk-free rate r_t, $LGD = (1 - RR)$ is the loss given default, and $CQDF_{i,t}$ is the cumulative default probability in T years. Assuming a constant probability of default $PD_{i,t}$, the cumulative default probability is then computed as $CQDF_{i,t} = 1 - (1 - PD_{i,t})^T$.[2]

Many researchers and practitioners commonly assume a constant LGD between 45% and 75%, depending on issuer (e.g., a company or a country) and on issue-specific characteristics such as firm size, seniority, duration, sector, and rating.[3] In our analysis, we set the LGD at 55%. Our approach can easily accommodate other LGD values or a more complex setting where the LGD is directly modeled.

If we rewrite $P_{i,t}(T) = e^{-y_{i,t}T}$ where y_t represents the risky-bond yield, the credit spread $s_{i,t} = y_{i,t} - r_t$ can be obtained by rearranging Equation 1 as

$$s_{i,t} = -\frac{1}{T} \ln[1 - LGD \times CQDF_{i,t}] \tag{2}$$

The credit risk literature addresses two main types of models: structural and reduced-form. Structural models, first introduced by Merton (1974), link the probability of default to the capital structure of a firm. This probability is modeled as a function of the issuing firm's economic fundamentals and is derived from the insight that a firm's equity value is functionally a call option on its asset value. Therefore, the risky debt embeds a (short) put option.

In reduced-form models, the default event is modeled as a statistical process. The most common specification includes a Poisson process defining the intensity and timing of the default. Structural models are primarily designed for a single firm (more recently by Du, Elkamhi, and Ericsson 2019, among others) or adapted to a panel of firms (as by Kelly, Manzo, and Palhares 2020). In contrast, reduced-form models prove particularly useful when defining the capital structure of an entity is difficult, as in the case of sovereign default risk (Pan and Singleton 2008, Longstaff et al. 2011, and Manzo and Veronesi 2016).

[2] If $PD_{i,t}$ represents the 1-year probability of default, then T is expressed in years. For example, if $PD_{i,t} = 2.5\%$, the probability of defaulting over the next five years is $CQDF_{i,t} = 1 - (1 - 2.5\%)^5 = 11.89\%$.

[3] Some evidence exists that LGD is not constant as recovery rates tend to correlate negatively with default rates over the business cycles (e.g., Altman et al. 2005). This effect is mainly relevant for corporate defaults; it is less applicable to sovereign defaults where the expected recovery is a function of the size of the country and the size and distribution of its external debt. However, separately identifying PD and LGD is difficult in practice because the identification depends on the parameters governing PD (Pan and Singleton 2008).

STRUCTURAL MODELS OF CREDIT RISK

Merton (1974) provides the foundational work for structural models by introducing a link between the credit and options markets. In structural models of credit risk, the credit spread is computed from the put option price. This link will help us construct more complex models by exploiting the modeling techniques developed in the options literature (Carr and Madan 1999).

In the following sections, we review the Merton (1974) model, discuss the Merton (1976) and Kou (2002) models that introduce jump risk, and then present a general model with three stochastic state variables that subsumes several others. We subsequently present a more detailed overview of the options pricing based on the characteristic function.

Merton Model (1974)

Consider a company with outstanding debt at face value D and maturity T. Under the assumption that default can only occur at T, the shareholder faces two possible scenarios. First, if the company's asset value is sufficiently high to repay the debt, $A_T > D$, then the creditor is paid in full, and the shareholder can claim the difference $A_T - D$. Second, if the company's asset value lies below the debt face value, $A_T < D$, the creditor claims the full value of the company's asset—that is, the firm goes bankrupt, and the shareholder receives nothing. The shareholder payoff is basically the same as a call option on the asset value of the firm with strike price D, that is, $E_T = max\,(A_T - D, 0) := (A_T - D)^+$.

To formalize this concept, Merton (1974) assumes that the asset value of the company evolves as a geometric Brown motion described by

$$\frac{dA_t}{A_t} = (r - q)dt + \sigma dW_t \tag{3}$$

where r is the risk-free rate, q is the dividend yield, and σ is the asset volatility.[4] Leverage is defined as the present value of the debt divided by the asset, $L = De^{-rT}/A_0$. We can express the present value of equity as the current price of the call option

$$E_0 = A_0[N(d_1) + L \times N(d_2)] \tag{4}$$

$$d_1 = \frac{-\log L}{\sigma\sqrt{T}} + \frac{1}{2}\sigma\sqrt{T};\; d_2 = d_1 - \sigma\sqrt{T} \tag{5}$$

where $N(\cdot)$ represents the cumulative distribution function of a standard normal distribution.

The bondholder's payoff at maturity T thus is expressed as

$$\begin{aligned} min(A_T, D) &= A_T - (A_T - D)^+ \\ &= D - (D - A_T)^+ \end{aligned} \tag{6}$$

$$= D - E_T \tag{7}$$

[4] This process is defined under the risk-neutral $\mathbb{Q}$ probability measure, so the drift is the risk-free rate. The market price of risk is $\phi = (\mu - r)/\sigma$ where μ is the mean rate of asset return.

The price of a defaultable bond can then be computed as

$$D_0 \equiv e^{-yT} = e^{-rT} - Put_0 \tag{8}$$

where y is the yield of the risky bond and Put_0 is the price of a put option with strike price D. The credit spread s can be calculated as

$$\begin{aligned} s \equiv y - r &= -\frac{1}{T}\log(1 - Put_0 \,/\, L) \\ &= -\frac{1}{T}\log(N(-d_1) \,/\, L + N(d_2)) \end{aligned} \tag{9}$$

Equation 8 highlights an important and general feature of structural models: the credit spread s is a function of a put price.

Adding Jump Risk

Despite the elegance of the Merton (1974) model, its assumption of Gaussian shocks in the asset growth dynamics does not make it a good candidate model to explain observed credit spreads, especially of highly rated companies,[5] in part because Gaussian shocks do not generate enough tail variation in the asset growth distribution. To capture this empirical behavior, Merton (1976) and Kou (2002) introduce tail risk in the form of a jump process in the asset dynamics. They modify the asset growth process in Equation 3 to produce

$$\begin{aligned} \frac{dA_t}{A_t} &= (r - q - \lambda\xi)dt + \sigma dW_t + (J - 1)dp \\ dp &= \begin{cases} 0 & \text{with prob. } 1 - \lambda dt \\ 1 & \text{with prob. } \lambda dt \end{cases} \end{aligned} \tag{10}$$

where dp is a Poisson process, independent of dW_t, with constant jump intensity λ; J is a random variable that captures the jump size; and $\xi = E[J - 1]$ is the Poisson compensator that adjusts the drift of the asset growth to guarantee risk neutrality. When $dp = 1$, the asset value jumps by JA.

Merton (1976) and Kou (2002) differ in their definition of the jump process. Merton (1976) specifies log-normal distributed jumps, $J \sim N(-\mu_J, \sigma_j^2)$, with constant Poisson intensity λ and expected jump size of $\xi = e^{-\mu_J + .5\sigma_J^2} - 1$. Kou (2002) proposes an asymmetric double-exponential jump with density, expressed as

$$v(\xi) = \lambda\left\{p\alpha_+ e^{-\alpha_+\xi}1_{\xi \geq 0} + (1 - p)\alpha_- e^{-\alpha_-\xi}1_{\xi < 0}\right\} \tag{11}$$

where p is the probability of a positive jump and α_+ and α_- are the positive and negative jump sizes, respectively. The idea behind asymmetric jumps is motivated by the empirical observation that financial asset returns are left-skewed.

To compute the credit spreads for these models, we must solve for the options prices. We describe each model's characteristic function associated with the probability of the option finishing in the money (ITM). This probability determines the price of

[5] While empirical specifics are outside of the scope of this article, it is worth noticing that mixed evidence exists on the ability of the classical Merton (1974) model to match historical spreads. Huang, Shi, and Zhou (2020) conduct a specification analysis of five structural models and strongly reject the Merton (1974) model and two diffusion-based models with a flat default boundary; however, they do find that models with jump risk improve the fit of observed credit spreads. In contrast, Feldhütter and Schaefer (2018) argue that a proper calibration to individual firms leans in favor of the Merton (1974) model.

an option. Carr and Madan (1999), among others, show that when this characteristic function is available in closed form, the option price can be obtained by applying the fast Fourier transform. We use this approach to price more complex models.

The closed-form characteristic function of the Merton (1976) model is stated as

$$\phi_T(u) = \exp\left\{-\frac{1}{2}u(u+i)\sigma^2 - \lambda\left[(e^{-iu\mu_J - .5u^2\sigma_J^2} - 1) + iu(e^{-\mu_J + .5\sigma_J^2} - 1)\right]\right\} \tag{12}$$

The characteristic function of the Kou (2002) model is expressed as

$$\phi_T(u) = \exp\left\{iu\omega T - \frac{1}{2}u^2\sigma^2 T + \lambda T\left(\frac{p}{\alpha_+ - iu} - \frac{1-p}{\alpha_- + iu}\right)\right\}$$
$$\omega = -\frac{1}{2}\alpha^2 - \lambda\left(\frac{p}{\alpha_+ + 1} - \frac{1-p}{\alpha_- - 1}\right) \tag{13}$$

In the next section, we introduce models with time-varying volatility and jump risks. For these models, the advantage of the characteristic function approach becomes more obvious.

Stochastic Volatility and Stochastic Jumps

Constant volatility and constant jump risks characterize all of the credit risk models discussed so far. However, these models exhibit a pricing bias (Duan 1999). In fact, the empirical findings of higher implied volatility in the out-of-the-money region (called the volatility smile) is a direct violation of constant volatility and tail risk. Du, Elkamhi, and Ericsson (2019) conclude that the inclusion of time-varying volatility and jump risk can improve the ability of the model to capture time variation in volatility as well as variation in the term structure of default spreads. Similarly, Kelly, Manzo, and Palhares (2020) demonstrate that systematic volatility and jump risk are needed to match a panel of credit spreads.

Motivated by the empirical findings, we consider a credit risk model that incorporates more complex time-varying dynamics. We allow two volatility processes and a jump process, with intensity driven by both volatility shocks and an independent stochastic component.[6] We modify the asset growth dynamics in Equation 3 to include additional stochastic processes so that

$$\begin{aligned}
\frac{dA_t}{A_t} &= (r - q - \lambda_t\xi)dt + \sqrt{v_{1,t}}dW_{1,t} + \sqrt{v_{2,t}}dW_{2,t} + (e^{-q_J} - 1)dJ(\lambda_t) \\
dv_{1,t} &= k_{v_1}(\theta_{v_1} - v_{1,t}) + \sigma_{v_1}\sqrt{v_{1,t}}dW_t^{v_1} \\
dv_{2,t} &= k_{v_2}(\theta_{v_2} - v_{2,t}) + \sigma_{v_2}\sqrt{v_{2,t}}dW_t^{v_2} \\
\lambda_t &= a \times (v_{1,t} + v_{2,t}) + z_t \\
dz_t &= k_z(\theta_z - z_t) + \sigma_z\sqrt{z_t}dW_t^z \\
E[Av_1] &= \rho_1 \\
E[Av_2] &= \rho_2
\end{aligned} \tag{14}$$

[6] Carr and Wu (2007) document an empirical relation between the currency option-implied volatility and the sovereign CDS spreads, and they model the default intensity as a function of this stochastic volatility. Kelly, Manzo, and Palhares (2020) employ a similar approach to tie the systematic stochastic volatility of the aggregate asset growth to the jump intensity for pricing the risk of safe companies at short maturities.

where the volatilities $v_{1,t}$ and $v_{2,t}$ and the independent component of jump intensity z_t all follow mean reverting processes with mean reversion speed k_i, mean reversion level θ_i, and volatility σ_i with $i \in \{v_1, v_2, z\}$. The correlation parameters ρ_1 and ρ_2 capture the interaction between volatility and asset growth. We allow volatility variation to drive the default intensity through the loading a and the independent variation governed by mean reverting process z_t.

The general specification of the model in Equation 14 incorporates the following models, with $v_{1,t}$, $v_{2,t}$, λ_t, and ξ defined in Equation 14:

- Heston (1993) model (SV), noted as

$$\frac{dA_t}{A_t} = (r-q)dt + \sqrt{v_{1,t}}dW_{1,t} \tag{15}$$

- Stochastic volatility and stochastic jump (1SV1SJ), expressed as

$$\frac{dA_t}{A_t} = (r-q-\lambda_t\xi)dt + \sqrt{v_{1,t}}dW_{1,t} + (e^{-q_J}-1)dJ(\lambda_t) \tag{16}$$

- Two stochastic volatilities and stochastic jump (2SV1SJ), specified as

$$\frac{dA_t}{A_t} = (r-q-\lambda_t\xi)dt + \sqrt{v_{1,t}}dW_{1,t} + \sqrt{v_{2,t}}dW_{2,t} + (e^{-q_J}-1)dJ(\lambda_t) \tag{17}$$

- Stochastic jump only (0SV1SJ), expressed as

$$\frac{dA_t}{A_t} = (r-q-\lambda_t\xi)dt + (e^{-q_J}-1)dJ(\lambda_t) \tag{18}$$

- Two stochastic volatilities (2SV), noted as

$$\frac{dA_t}{A_t} = (r-q)dt + \sqrt{v_{1,t}}dW_{1,t} + \sqrt{v_{2,t}}dW_{2,t} \tag{19}$$

Empirical research indicates that the credit term structure of single-name entities is primarily driven by one factor that captures parallel shifts in the term structure—a level factor (Pan and Singleton 2008, Manzo 2013). The more complex multifactor structure introduced in this section enables researchers to better understand richer dynamics, such as systematic components in credit risk.

Option Pricing Based on the Characteristic Function

In this section, we review a common options pricing technique. To price an option, we must compute the probability of the option finishing in the money. For a European call option on a firm's asset A with debt maturity T and face value D, the current price is expressed as

$$c = A\Pi_1 - De^{-rT}\Pi_2 \tag{20}$$

with

$$\Pi_2 = \Pr(A_T > D) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{e^{-iu\ln(D)}\phi_T(u)}{iu}\right)du$$
$$\Pi_1 = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{e^{-iu\ln(D)}\phi_T(u-i)}{iu\phi_T(-i)}\right)du \quad (21)$$

where Π_2 is the risk-neutral probability of finishing in the money and Π_1 is the delta, both defined on the set of real number $\Re(\cdot)$. $\phi_T(u)$ is the characteristic function of $a_T = \ln(A_T)$, that is, $\phi_T(u) = E[e^{iuaT}]$.

Given a closed-form characteristic function $\phi_T(u)$, Carr and Madan (1999) propose using the fast Fourier transform to convert the characteristic function into a probability. Carr and Madan (1999) provide a numerical tool to calculate the integral in Π_1 and Π_2, and they introduce a modified call price that makes the pricing function square-integrable. The integral requires a user-defined upper bound that is typically a power of the number 2. The modified call price is based on an additional parameter α that can be either optimized or set at a rule-of-thumb value of 0.75. Both the upper bound and α govern the approximation error of the numerical pricing function. The higher the precision (e.g., an upper bound of 2^{14}), the slower the pricing function. If we pair high precision with an optimization routine for α, we can further slow the computation of options prices. If we want to accelerate the pricing function, we have to accept lower precision.

This tradeoff between speed and precision constitutes a central feature in numerical methods. We demonstrate that our deep learning approach offers an alternative technique for computing credit spreads, potentially bypassing this tradeoff. Once the deep learning models are trained, they can quickly and accurately calculate credit spreads for new parameters under models of varying complexity.

Reduced-Form Models of Credit Risk

Although the link between the default probability and the capital structure offers an attractive economic intuition, structural models may not easily apply when the capital structure is complex (e.g., when the issuer is a country or a municipal government). Reduced-form models overcome this issue by defining a statistical process for the default event, with arrival governed by an intensity-based or hazard rate process.

The seminal work of Jarrow and Turnbull (1995) introduces the first reduced-form model of credit risk, and an extensive body of literature develops around this approach. In this article, we focus on the Pan and Singleton (2008) model, which describes default risk by a stochastic log-normal intensity that can generate fatter tails than a Gaussian process. The default probability does not have a closed-form solution. Calculating this probability requires a numerical method that can be computationally intensive.

Unlike the structural models that explicitly price the payoff of a bond, the Pan and Singleton (2008) model is designed for credit default swap (CDS) spreads. Given its swap structure, a CDS contract is priced differently than a bond. For a CDS, two parties are involved: the seller and the buyer. The buyer commits to quarterly payments of the premium $CDS_{i,t}(T)$ to hedge the default risk of entity i over T years (i could be a single firm, a portfolio of firms, or a country). The discounted value of these payments is then

$$\text{Premium Leg} = 4 \times \sum_{j=1}^{4T} E_t^{\mathbb{Q}} \left[e^{-\int_t^{t+.25j} (r_s + \lambda_s) ds} \right] = 4 \times \sum_{j=1}^{4T} D_{t,j} E_t^{\mathbb{Q}} \left[e^{-\int_t^{t+.25j} \lambda_s ds} \right] \tag{22}$$

where the second row follows from the assumed independence between the interest rate r_t and the default rate λ_t and where $D_{t,j}$ is the risk-free discount function.

The seller commits to pay the loss given default. The default leg is expressed as

$$\begin{aligned} \text{Default Leg} &= LGD \times \int_t^{t+T} E_t^{\mathbb{Q}} \left[\lambda_u e^{-\int_t^u (r_s + \lambda_s) ds} \right] du \\ &= LGD \times \int_t^{t+T} D_{t,u} E_t^{\mathbb{Q}} \left[\lambda_u e^{-\int_t^u \lambda_s ds} \right] du \end{aligned} \tag{23}$$

where *LGD* is the loss given default and $E_t^{\mathbb{Q}} \left[\lambda_u e^{-\int_t^u \lambda_s ds} \right]$ represents the default probability.

The par spread is defined as the spread where the default leg equals the premium leg at the inception of the contract, and it can be obtained by making the calculation[7]

$$CDS_{i,t}(T) = \frac{LGD \times \int_t^{t+T} E_t^{\mathbb{Q}} \left[\lambda_u e^{-\int_t^u (r_s + \lambda_s) ds} \right] du}{4 \times \sum_{j=1}^{4T} E_t^{\mathbb{Q}} \left[e^{-\int_t^{t+.25j} (r_s + \lambda_s) ds} \right]} \tag{24}$$

Pan and Singleton (2008) assume a log-normal mean reverting process for the default intensity, as noted in

$$d \ln \lambda_t = k(\theta - \ln \lambda_t) dt + \sigma_\lambda dB_t \tag{25}$$

where k is the mean reversion speed, θ is the long-term average intensity, and σ_λ is the volatility. Although the log-normal assumption offers the advantage of generating fatter tails compared to a normal distribution—thus making the default event more likely—the probability of default is not available in a closed-form solution. In fact, Pan and Singleton (2008) employ a fully implicit finite-difference method to approximate this probability on a grid. Like the options pricing based on the characteristic function (described in the previous section), this method can be computationally intensive.

DEEP NEURAL NETWORKS

We use neural networks to approximate the nonlinear relationship between model parameters and credit spreads. In this section, we provide an overview of neural networks and discuss our empirical choices.

Neural networks developed separately in statistics and artificial intelligence (Hastie, Tibshirani, and Friedman 2009). Within statistics, neural networks research grew from the areas of semiparametric statistics and smoothing. In artificial intelligence, neural networks seek to model biological neural networks with individual artificial neurons (nodes). Neural networks have evolved into one of the most important

[7] In 2009, the ISDA issued the "Big Bang" protocol that incorporated fixed coupons in the CDS contract. For North American contracts, this coupon is either 100 basis points or 500 basis points. In this article, we do not take into account the coupon structure and the pricing of the so-called par spread.

machine learning algorithms, with broad applications in many areas, including speech recognition, machine translation, and computer vision.[8]

Single-Layer Neural Network

A neural network is a system of nodes that can learn the relationship between any sets of inputs and outputs. A simple neural network contains only a single hidden layer. Suppose that we are trying to predict a continuous variable Y with N-dimensional input X. In an ordinary least squares regression, the model predicts Y as a linear function of X. A single-layer neural network introduces an intermediate step that involves a nonlinear transformation $\sigma(\cdot)$ of a linear function of X, $\alpha_{0m} + \alpha_m^\top X$. The final prediction for Y is then a linear function of this transformation, expressed as

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^\top X), \quad m = 1, \ldots, M \tag{26}$$

$$Y = \beta_0 + \beta_k^\top Z \tag{27}$$

where Z_m represents the M nodes in the hidden layer. The nonlinear transformation $\sigma(\cdot)$ is called the activation function. Commonly used nonlinear activation functions include the hyperbolic tangent function, sigmoid functions, and rectified linear unit (ReLU).[9] If the activation function $\sigma(\cdot)$ were linear, we recover a linear relationship between X and Y that can be captured by an ordinary least squares regression. Although each node in the neural network embeds a simple nonlinear transformation, the network combines many nodes to accommodate a much more complex relationship between X and Y.

When modeling credit risk, input X represents the set of model parameters while output Y specifies the credit spread for a single maturity. For example, in the Heston (1993) model, X includes the parameters associated with the stochastic volatility process:

(1) mean reversion speed k,
(2) relative volatility of volatility σ,
(3) mean reversion level θ,
(4) correlation between asset growth and volatility ρ,
(5) initial level of stochastic volatility v_0,
(6) leverage, and
(7) maturity.

The model parameters in a neural network are called weights. Each hidden node Z_m depends on all the inputs X and weights α, and the output is a function of Z_m and weights β. To operationalize the neural network, we estimate the model's weights in a training phase. During training, the optimal weights result from an optimization procedure that minimizes a cost function. We use the common mean squared error approach as the cost function for predicting continuous variables. This choice also aligns with the way pricing models are typically calibrated to actual data.

Deep Learning

Neural networks represent a flexible methodology for function approximation. Cybenko (1989) demonstrates that a single hidden layer containing a finite number of nodes can approximate arbitrary continuous functions—a result known as the

[8] Goodfellow, Bengio, and Courville (2016) offer a more in-depth treatment.

[9] The hyperbolic tangent function is $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$; the sigmoid function is $\sigma(x) = 1/(1 + e^{-x})$; and the ReLU is $\sigma(x) = \max(0, x)$.

universal approximation theorem. In practice, with one hidden layer, the number of nodes needed to approximate complex functions could be quite large.

We can generalize single-layer neural networks by increasing the number of hidden layers to further enlarge the set of possible models, allowing for more complex nonlinearities. Rather than using one hidden layer with a large number of nodes, we could incorporate multiple hidden layers, each with fewer nodes, and achieve the same level of complexity.

In the field of artificial intelligence, the history of neural networks extends at least back to the 1950s, when Rosenblatt (1958) introduces a single-layer neural network to learn a linear decision boundary. For decades, scientists believe that sophisticated neural networks are very difficult to train. As the computer infrastructure for deep learning (both hardware and software) improves over time, interest in neural networks resurges over the past two decades. In addition, neural networks also become more powerful as the quantity of training data expands. Researchers begin to train deeper neural networks than ever before, and the term "deep learning" emerges to refer to neural networks with multiple hidden layers (Goodfellow, Bengio, and Courville 2016). Deep learning now constitutes a highly active research area, and some researchers have trained models with more than 1,000 hidden layers.

In our application to credit risk modeling, we consider multilayer neural networks to approximate the pricing function (i.e., the relationship between the model parameters and the credit spreads). In this sense, our deep learning models replace complex solutions to credit risk models.

Empirical Choices

In the practical implementation of neural networks, the researcher must make a number of empirical choices that can greatly impact the efficacy of neural networks for approximating functions. In this section, we note the main considerations in the practical implementation of a deep learning model. In particular, we discuss choices relevant to the model architecture, training and test sets, activation function, batch size, and number of epochs. We also address predictive accuracy and the software that we use to implement deep learning.

The architecture of the deep learning model refers to the arrangement of nodes into layers. Although the universal approximation theorem states that a single-layer neural network can approximate any continuous function, the theorem does not specify the number of nodes needed to do so. For complex functions, the number of nodes required in a single hidden layer could be very large. Researchers often employ multiple hidden layers, so the number of nodes per layer can be greatly reduced. In our implementation, we vary the number of hidden layers, depending on the complexity of the credit risk model, and we fix the number of nodes at 100.[10] The activation function governs the nonlinearity embedded in each node; we use the ReLU.

Deep learning models must be trained to learn the relationship between inputs and outputs. The training set, a collection of observations, is used to fit and optimize the deep learning model. Learning is achieved by minimizing the cost function on the training set. The fit on the training set is defined as an in-sample fit, similar to an in-sample R-squared in a linear regression. Because the model uses the training set to optimize weights, the value of the cost function on the training set is likely to exhibit a downward bias relative to an out-of-sample fit. In our setting, we try to approximate known model structures, so well-trained deep learning models should capture the

[10] In unreported results, we compute the out-of-sample R-squared of trained neural networks with 10, 20, and as many as 150 nodes per hidden layer for one of the structural models (2SV0SJ) and for the reduced-form model. We observe an increase in the out-of-sample R-squared for neural networks with more than 60 nodes. However, after the network exceeds 100 nodes, we do not see significant improvement.

complete relationship between model parameters and credit spreads. Therefore, we expect the deep learning models to achieve a high R-squared, close to 100%.

We train the deep learning models on simulated data, with 95% of the simulated observations used for training and the remaining 5% reserved as the test set for evaluating model performance. The test set is a collection of observations that the neural network weights are not optimized on, so it serves as an out-of-sample test for the model. A strong model shows accurate predictions on the test set as well as the training set.

In the training phase, we need to learn the optimal weights of the neural network. We use the Adam algorithm, a stochastic gradient descent method that employs discrete steps to search for optimal parameters.[11] Given the prediction error, weights are updated by using a back-propagation algorithm. The researcher can choose the number of samples to use at each iteration (i.e., the batch size), which we set to 1,024 observations. The batch size regulates the noise in the update. If the batch size is too small, then the gradient calculation is subject to noisy sampling variation, but frequent optimization updates could lead to faster learning. If the batch size is too large, then computing the gradient entails less noise, but the model updates could be very slow for a large data set.

The number of epochs defines how many times the deep learning model will execute the entire training set to optimize model weights. For one epoch, the deep learning model uses each observation once to update the model weights. We follow the typical practice, which allows many epochs so that the deep learning algorithm can run until the training error is sufficiently small. We set the number of epochs at 500 when training the deep learning models.

For the practical construction and implementation of deep learning models, we use the Keras package in Python. The high-level Keras package implements deep learning models without handling low-level operations such as tensor products or numerical optimization schemes, enabling researchers to focus on modeling choices rather than computational options.

USE OF NEURAL NETWORKS TO PREDICT CREDIT SPREADS

A credit risk model can be used as a pricing function: for a given set of parameters, the model generates a credit spread. In this section, we demonstrate that deep learning can approximate this relationship between model parameters and credit spreads. Similar to a pricing function, a deep learning model can produce credit spreads for a given set of parameters, thus capturing the complexity of the credit risk model.

We train deep learning models in two main steps. First, we simulate various combinations of model parameters and credit spreads. Second, we train the deep learning models on the simulated data so that they learn the pricing relationships between model inputs and credit spreads. We thereby demonstrate that deep learning models can accurately capture these relationships.

Simulated Data

Let $f_M(\Theta_M)$ be the pricing function for model M and model parameters Θ_M where M is one of the previously reviewed credit risk models. Our goal is using deep learning to approximate the pricing function f_M. We generate simulated data of the form $\{\Theta_M, f_M(\Theta_M)\}$ that we then use to train the deep learning models. We consider five

[11] According to Kingma and Ba (2014), the Adam algorithm is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters."

EXHIBIT 1

Simulated Set of Model Parameters

	Structural Models							
	σ^a	λ	μ_J	σ_J	p^{up}	μ^{up}	μ^{down}	
Merton (1974)	[0.001, 2]	–	–	–	–	–	–	
Merton (1976)	[0.001, 2]	[0.001, 1]	[−2, 0]	[0.001, 1]	–	–	–	
Kou (2002)	[0.001, 2]	[0.001, 1]	–	–	[0.001, 1]	[0.001, 1]	[0.001, 1]	
	v_0	σ_{var}	κ_{var}	θ_{var}	ρ_{Av}	λ	μ_J	σ_J
Heston (SV)	[0.001, 1]	[0.001, 1]	[0.001, 2]	[0.001, 1]	[−0.99, −0.05]	–	–	–
Heston with jumps	[0.001, 1]	[0.001, 1]	[0.001, 2]	[0.001, 1]	[−0.99, −0.05]	[0.001, 1]	[−2, 0]	[0.001, 1]
	$v_{0,i}$	$\sigma_{var,i}$	$\kappa_{var,i}$	$\theta_{var,i}$	$\rho_{Av,i}$			
	[0.001, 1]	[0.001, 1]	[0.001, 2]	[0.001, 1]	[−0.99,−0.05]			
2SV1SJ	z_0	σ_z	k_z	θ_z	μ_J	σ_J	α_{var}	
	[0.001, 1]	[0.001, 1]	[0.001, 2]	[0.001, 1]	[−2, 0]	[0.001, 1]	[0.001, 1]	
	Reduced-Form Model							
	λ	σ_λ	κ_λ	θ_λ				
Pan and Singleton (2008)	[0.001, 1]	[0.001, 2]	[0.001, 2]	[−7, 1]				

NOTES: This exhibit reports the parameter ranges used in our Latin hypercube sampling algorithm to draw 50,000 combinations. For each model, the exhibit reports the range of parameter values as [lower bound, upper bound]. The 2SV1SJ model subsumes the 1SV1SJ, 0SV1SJ, and 2SV0SJ models. For the structural models, the leverage range is set to [0.01, 0.99]. Each set of simulated parameter values is paired with five maturities: 1Y, 3Y, 5Y, 7Y, and 10Y.

[a]Please refer to the models section for details on model specification and parameters.

maturities: 1 year (1Y), 3 years (3Y), 5 years (5Y), 7 years (7Y), and 10 years (10Y). For each model, we simulate 50,000 different combinations of model parameters Θ_M, using the Latin hypercube sampling (LHS) approach, which generates random samples on a grid in a multivariate distribution, resulting in a more representative sample from the parameter space than that derived from the same number of samples on a grid (Liu, Oosterlee, and Bohte 2019). As the dimension of the parameter space increases, the number of data points required in a grid to cover the parameter space expands exponentially, and the approach quickly becomes computationally intensive. Latin hypercube sampling generates near-random samples of parameter values even when the dimension of the parameter space is high. For each set of parameter combinations, we use one of the identified credit risk models to compute the credit spreads. For the structural models, credit spreads in the Merton (1974) model are calculated in a closed-form solution; all other models employ the characteristic function approach to compute the credit spreads. We follow the Pan and Singleton (2008) model approach, applying a fully implicit finite-difference method to numerically calculate the probability of default. For each model, with five different maturities and 50,000 sets of parameter combinations, we produce an artificial data set of 250,000 observations that we use to train the deep neural networks.

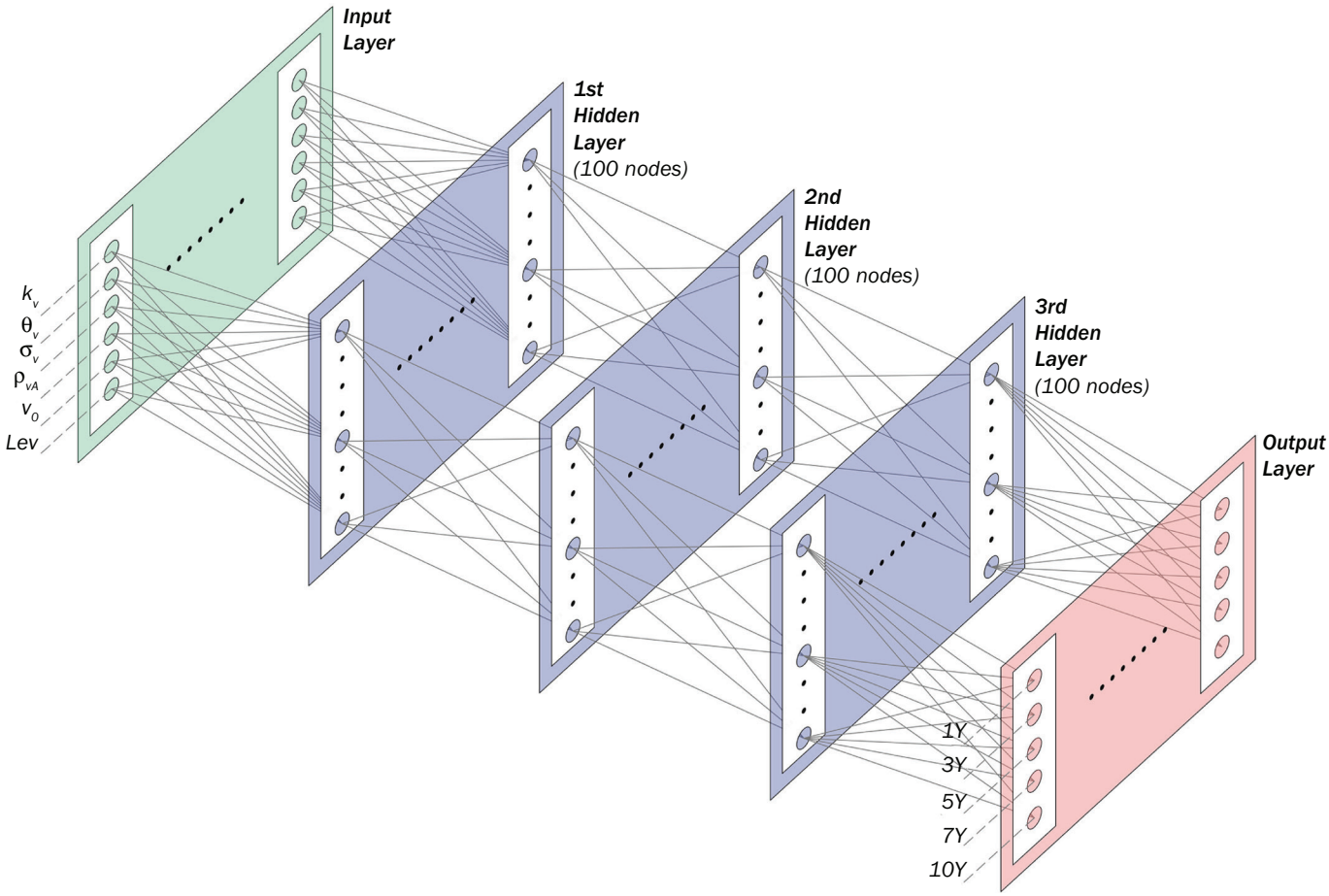
Exhibit 1 reports the upper and lower bounds for our parameter space. We set the parameter bounds based on the empirical calibration exercises of, among others, Pan and Singleton (2008) and Du, Elhamhi, and Ericsson (2019). For all of the structural models, we set leverage between 0.01 and 0.99, thus including a range of companies (from low-leverage to high-leverage firms).

Training and Evaluation

Using the simulated datasets, we train the deep neural networks. For each model, we randomly divide the 250,000 simulated observations into training and

EXHIBIT 2

Deep Neural Network Used for the Heston Model



NOTES: This exhibit illustrates the deep learning architecture used to learn the pricing function of the Heston model. (1) The green layer represents the inputs: four parameters describing the stochastic volatility process (k_v, θ_v, σ_v, and ρ_{vA}), the initial volatility, v_0, and the leverage *Lev*. (2) The middle blue panels represent three fully connected hidden layers with 100 nodes each. (3) The final red layer represents the outputs—a term structure of credit spreads for five maturities.

test samples, assigning 95% of the observations to training and 5% to the test set to evaluate the deep learning models. We standardize the simulated credit spreads so that they have zero mean and unit standard deviation (by subtracting the mean and dividing by the standard deviation). We follow the example of Horvath, Muguruza, and Tomas (2019) and normalize the input parameters in the range –1 to 1, using the upper and lower bounds in Exhibit 1, as defined in

$$\frac{2\theta - (\theta_{max} + \theta_{min})}{\theta_{max} - \theta_{min}} \in [-1,1] \tag{28}$$

where θ_{max} is the upper bound, θ_{min} is the lower bound, and θ is the raw parameter value. Normalizing inputs and outputs is common in deep learning because it accelerates learning and enables faster convergence while potentially avoiding local optima.

Exhibit 2 depicts a schematic of a deep neural network that we build to learn the pricing function of the Heston (1993) model. The Heston model is described by four parameters governing the stochastic volatility (k_v, θ_v, σ_v, and ρ_{vA}), by the initial value

EXHIBIT 3

Model Architectures and Test Set Performance

Panel A: Common Specification

Simulated Data	250,000
Train Size	95%
Test Size	5%
Batch Size	1,024
Epochs	500
Units (#nodes)	100
Act. Fun	ReLU
Output Fun	Linear

Panel B: DNN Architecture

Models	Hidden Layers
Merton (1974)	2
Merton (1976)	2
Kou (2002)	2
Heston (SV)	3
Heston with jumps	3
1SV1SJ	3
2SV1SJ	3
0SV1SJ	3
2SV0SJ	3
Pan and Singleton (2008)	3

Panel C: Test Set *R*-Squared

1Y	3Y	5Y	7Y	10Y
0.999988	0.999985	0.999988	0.999991	0.999992
0.999981	0.999978	0.999980	0.999981	0.999981
1	1	1	1	1
0.999831	0.999930	0.999921	0.999891	0.999850
0.999733	0.999807	0.999804	0.999798	0.999751
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0.999763	0.999608	0.999581	0.999500	0.999426
0.999651	0.999652	0.999649	0.999652	0.999653

NOTES: This exhibit notes the architecture of the deep neural networks. Panel 3A reports the size of the simulated data, training set, test set, and batch size. For each model, Panel 3B lists the number of hidden layers and the units per layer. Panel 3C notes the R-squared on the test set for each time period.

of stochastic volatility v_0, and by leverage *Lev*. We use three hidden layers, each with 100 nodes. Each set of parameter combinations is passed to the first hidden layer and then transformed using the ReLU activation function, which feeds sequentially into the second and third layers. The output layer provides a term structure of credit spreads for each combination of parameters. In the training phase, we obtain the optimal weights that minimize the error between the simulated spreads and the spreads calculated by the deep neural network.

In Exhibit 3, Panel 3A reports the common specification that applies to all the deep neural networks. In particular, we split 250,000 simulated credit spreads into 95% training set and 5% test set, with a batch size of 1,024 observations, 500 epochs, and 100 nodes per hidden layer (Liu, Oosterlee, and Bohte 2019). ReLU serves as the activation function.

Panel 3B (Exhibit 3) lists the specific architecture used for each model. We employ three hidden layers for most of the models, except for the simplest structural models of Merton (1974, 1976) and Kou (2002); for these, we utilize two hidden layers. This choice reflects differences in model complexity. Unlike the SVSJ models, the three simplest structural models are all characterized by static parameters and no time-varying state variables.

In addition, Panel 3C (Exhibit 3) notes the R-squared for each model and each maturity computed on the test set. We compare the predicted credit spreads from the deep learning models against the actual spreads. Across models and maturities, we observe a high level of accuracy, close to 100% R-squared. The deep learning

EXHIBIT 4

Efficiency of Deep Neural Networks

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Integral Upper Bound	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	**DNN**	**Efficiency (3)/(6)**
0SV1SJ	0.025	0.047	0.15	0.78	3.19	0.000998	150
1SV1SJ	0.024	0.060	0.19	1.04	4.04	0.000995	192
2SV0SJ	0.023	0.057	0.19	1.06	4.12	0.000995	196
2SV1SJ	0.027	0.064	0.24	1.25	4.94	0.000996	242
Heston (SV)	0.021	0.045	0.18	0.78	2.99	0.000996	177
Heston with Jumps	0.021	0.052	0.17	0.88	3.47	0.000994	172
Intensity Step	**1/100**	**1/200**	**1/300**	**1/400**	**1/500**		
Pan and Singleton (2008)	0.10	0.11	0.11	0.12	0.12	0.000993	113

NOTES: This exhibit reports the average time in seconds that a model takes to generate a full term structure of credit spreads. Columns 1 through 5 list the average speed of the pricing functions, from the least numerically precise to the most numerically precise. The precision of structural models is given by the upper bound of the integral in Equation 21 that defines the probability of the option finishing in the money. For the reduced-form model, precision is defined by the number of grid points of the default intensity. Column 6 notes the average speed for the deep neural networks. Column 7 lists the ratio of Column 3 to Column 6 as the relative efficiency.

models can accurately capture the relationship between the model parameters and the credit spreads for the 10 models that we consider.

Therein lies the first main contribution of our article: deep neural networks can learn both structural models and reduced-form models of credit risk, even when the underlying design is complex (e.g., the 2SV1SJ model).

Computational Efficiency of Deep Neural Networks

How efficient are deep neural networks in pricing credit risk? We measure efficiency as the speed of the model in generating credit spreads. This metric is particularly relevant when (1) a complex model is used for calibration to historical data and (2) the researcher must execute the pricing function many times to calculate the spreads. We compute the average time, across five simulations, that each model requires to produce a term structure of credit spreads. We then compare that time metric with the time it takes for the trained deep neural networks to perform the same task.

Exhibit 4 reports the average time in seconds. Columns 1 through 5 note the time recorded by each pricing function to build a term structure of spreads from a given set of model parameters. Because numerical integrals are required, we must select the degree of precision for computing the integrals. For structural models, we vary the upper bound for the integral of the probability of finishing in the money in Equation 21 from 2^{10} to 2^{18} (2^{10} discretizations give a less precise integral than 2^{18}). For 2^{10} discretizations, the pricing function computes the credit spreads in about 0.025 seconds. However, when we aim for higher precision, the time to generate spreads can range up to three seconds to four seconds. These few seconds may not seem like a lot of time, but when credit term structures are computed for many different input parameter combinations (e.g., during a calibration), thousands of iterations may be needed to identify the optimal model parameters. The total time required would differ greatly for 2^{18} and 2^{10}.

For the reduced-form model, we vary the number of grid points (from 100 to 500) to approximate the intensity of default in Equation 22. More grid points would lead to improved precision. For this model, we observe small differences across precision levels, with the speed ranging from 0.10 seconds with 100 steps to 0.12

seconds with 500 steps.[12] However, when compared to the time that a deep neural network takes to compute the term structure of credit spreads (Exhibit 4, Column 6), we observe significant differences. Deep neural networks generate the full term structure in approximately 0.001 seconds, regardless of the complexity of the model and regardless of whether the model is structural or reduced-form.

In Column 7 of Exhibit 4, we report the relative speed of deep neural networks and traditional pricing functions. We compare the middle value for numerical precision (Column 3) to the deep neural networks (Column 6). For the structural models, the trained neural networks are about 150 to 200 times faster—the more complex the underlying model, the greater the time savings from using a deep neural network. For the reduced-form Pan and Singleton (2008) model, the deep neural network is about 100 times faster.[13]

In our comparison, we must also take into account the training time for the deep learning models. However, the training time for our deep neural networks is not prohibitively long; on a standard personal computer with 16 gigabytes of RAM and four cores, the most complex models require just over one hour to train. Once the deep learning models are trained, they can save many hours of the researcher's time in calibration or other applications (when compared to numerical methods).

Deep neural networks can efficiently and accurately approximate the pricing function of a credit risk model. The natural follow-up question is whether trained neural networks can replace the pricing function to calibrate the model to historical data. We provide our answer in the next section.

DEEP LEARNING CALIBRATION

We demonstrate in the previous section how deep learning models can be used to approximate the pricing function of credit risk models, accurately and efficiently mapping the set of model parameters into credit spreads. A natural next step is assessing whether we can use the trained deep neural networks to calibrate the model to historical data.

For calibration, we select the structural (Heston 1993) stochastic volatility model and the reduced-form model of Pan and Singleton (2008). In these one-factor models, a single stochastic state variable drives the model dynamics. Empirically, the first principal component on the term structure of single-name entities explains more than 95% of the panel variation (Manzo and Veronesi 2016). Therefore, the one-factor structure in the Heston (1993) and Pan and Singleton (2008) models should be sufficient to capture time series and cross-maturity variation in credit spreads.[14]

The goal of calibration is to choose a set of parameters that optimizes a specific objective function. The most common objective function is derived from the assumption that the credit risk model can price observed credit spreads, with a pricing error that is assumed to have a known distribution. If we let $CDS_t(T)$ be the observed credit spread at time t for the maturity T, then we can write it as

$$CDS_t(T) = f_M(\Theta_M, T, S_t) + \epsilon_t \tag{29}$$

[12] The step, $1/m$, defines how refined the grid is. As an example, if we assume a range for the default intensity λ between 0.000001 and 1, a step of 100 (500) will space λ by 0.0099 (0.0019).

[13] We run our models on a commercial Windows 10 machine with an Intel Xeon Silver 4110 CPU (2.10GHz) with 32GB of memory.

[14] We do not calibrate nonstochastic models because their static parameters are known violations of time-varying volatility and jump risk.

where $f_M(\cdot)$ is the pricing function, Θ_M is the set of static model parameters, T is the time to maturity, and S_t is the vector of state variables. As an example, for the Heston (1993) model, Θ_M is the set of parameters governing the stochastic volatility process $\Theta_M = \{k_v, \theta_v, \sigma_v, \rho_{Av}\}$, and the stochastic volatility is the sole state variable $S_t = v_t$. The pricing error term ε_t is assumed to be normally distributed with a mean of 0 and a constant variance σ_ϵ^2 (homoskedastic errors). Although a simplifying assumption, homoskedastic errors help contain the number of parameters to estimate $\Theta_{M+} = \left\{\Theta_M, \sigma_\epsilon^2\right\}$. If we consider five maturities, for each time t, Equation 29 fits the full term structure—it relates the 5×1 vector of observed spreads $CDS_t(T)$ to the 5×1 vector of model spreads $f_M(\Theta_M, T, S_t)$, for a given set of parameters and a state variable. Note that for structural models, Equation 29 is also a function of the "observed" leverage L_t, $f_M(\Theta_M, T, L_t, S_t)$.

The calibration provides us with the set of parameters $\hat{\Theta}_{M+}$ associated with the smallest pricing errors. This goal can be achieved by either minimizing the (weighted) sum of squared errors or by maximizing the model's log-likelihood.

We propose replacing the pricing function $f_M(\cdot)$ with the trained deep neural networks $NN_M(\cdot)$ and assuming the same linear error term as in Equation 29 to yield

$$CDS_t(T) = NN_M(\Theta_M, T, S_t) + \epsilon_t \tag{30}$$

We previously conclude that deep neural networks can accurately approximate the pricing functions. By replacing pricing functions with deep learning models, we can take advantage of the computational efficiency of the deep neural network without sacrificing pricing accuracy.

We calibrate the models using the unscented Kalman filter. We cast the model into a state-space representation, with the state equation represented by the stochastic state variable. For the Heston (1993) model, the stochastic volatility is the state variable. Its first and second moments are expressed as

$$\begin{aligned} E[v_t^{Heston}] &= v_{t-1}e^{-k\Delta t} + \theta_v(1 - e^{-k\Delta t}) \\ Var[v_t^{Heston}] &= v_{t-1}\frac{\sigma_v^2}{k_v}(e^{-k\Delta t} - e^{-2k\Delta t}) + \frac{\theta_v\sigma_v^2}{2k_v}(1 - e^{-2k\Delta t}) \end{aligned} \tag{31}$$

For the Pan and Singleton (2008) model, the stochastic default intensity is the state variable, with first and second moments defined as

$$\begin{aligned} E[\lambda_t^{P\&S}] &= \exp(\ln(\lambda_{t-1})e^{-k\Delta t} + \theta_v(1 - e^{-k\Delta t})) \\ Var[\lambda_t^{P\&S}] &= \frac{\sigma_v^2}{2k_v}(1 - e^{-2k\Delta t}) \end{aligned} \tag{32}$$

where $\Delta t = 1/52$. The measurement equation is specified by Equation 30. Because the measurement equation is nonlinear, the unscented Kalman filter accrues an advantage over the extended Kalman filter: no need to derive analytical approximations because the unscented Kalman filter approximates any equations in the distribution space via sigma points (Wan and Van Der Merwe 2000). We call our approach, which combines the unscented Kalman filter with deep neural networks, NN-UKF (the appendix includes further technical details).

The deep neural networks are trained on standardized data. Therefore, in the practical implementation, we standardize both the state equation and the parameters

EXHIBIT 5

Estimated Parameters and In-Sample R-Squared for Model Calibration

Model	Calibrated Parameters					In-Sample R^2 (%)
	k_v	θ_v	σ_v	ρ_{Av}	σ	
Heston SV	0.001	0.135	0.136	−0.977	0.0000	89.3
	k_λ	θ_λ	σ_λ	–	σ	
Pan and Singleton (2008)	0.037	0.375	0.393		0.0009	96.9

NOTES: This exhibit lists the calibrated parameters, with the numerical standard errors in parentheses, for the structural Heston (1993) model and the reduced-form Pan and Singleton (2008) model on the term structure of CDS spreads of American Express (AXP). The data are weekly, from January 2001 to September 2014. The Heston (1993) model is calibrated using leverage constructed with the book value as well as the market value of asset. Calibrations use the neural network–based unscented Kalman filter (NN-UKF) implemented under the risk-neutral probability measure. The parameter σ_ε is the standard deviation of the pricing error. R^2 is the in-sample goodness of fit. Please see the models section for model specification and parameters.

Θ_M by using Equation 28 before feeding them into the trained $NN_M(\Theta_M, T, S_t)$.[15] The deep learning models then produce standardized credit spreads, which we transform using the mean and variance of the observed spreads. The final NN-UKF output is the model's total log-likelihood, which we aim to maximize to find the optimal set of parameters Θ_{M+}.

We choose American Express (AXP) as a random name in the North American CDS universe, and we collect weekly data from January 2001 to September 2014 for the five maturities (1Y, 3Y, 5Y, 7Y, and 10Y). Our initial sample is limited by the availability of the CDS universe, which developed in the early 2000s. In the calibration exercise, our goal is to stress the approximating pricing function in the presence of relevant observable inputs, such as leverage for the structural model, as well as a multiyear period that spans normal and distressed periods, including the Great Recession. We collect annual accounting data from Google Finance and measure leverage as the ratio of total liabilities over the book value of asset. We assume that leverage is constant within each year and that it is consistently high, ranging between a maximum of 0.93 in 2009 to a minimum of 0.87 in 2014. We also calibrate a model where leverage is constructed using the asset, that is, the sum of total liabilities and market capitalization. Compared to the book value, the market value of asset adds time variation to the observed leverage. Consequently, we expect the market value to improve the model fit to historical data.

In our calibration, we assume a risk-free rate of zero, an assumption that does not materially affect our results because CDS spreads display low sensitivity to interest rates (Duffie 1999). We calibrate both models under the risk-neutral probability measure, but our proposed NN-UKF approach is flexible, accommodating the inclusion of a market price of risk in the volatility and intensity dynamics.

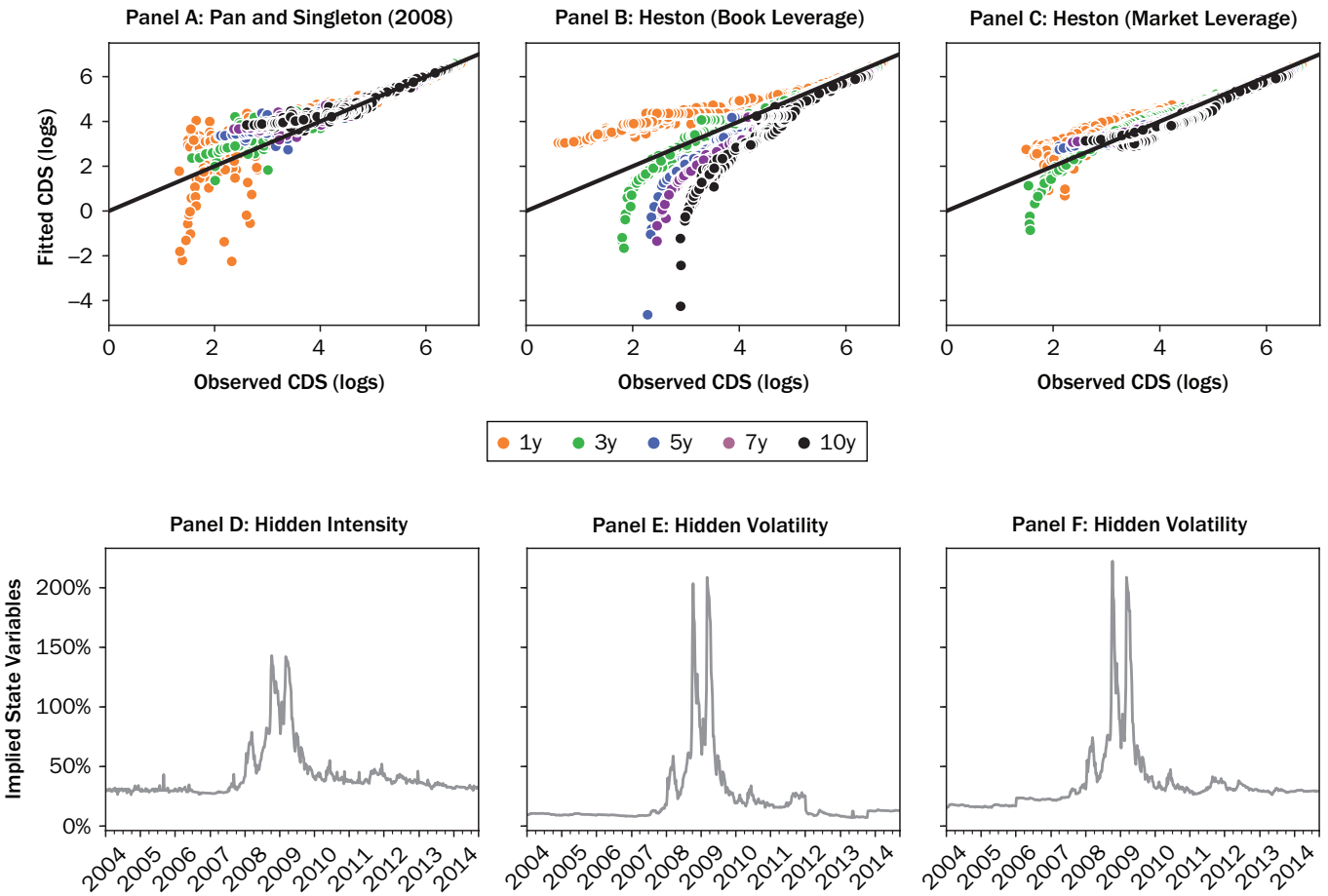
Exhibit 5 notes the calibrated parameters, with their numerical standard errors in parentheses. In both the structural models and the reduced-form models, the underlying state dynamics show (1) persistent variation in volatility, with low mean reverting speeds of 0.001; and (2) less persistence for the default intensity, with a coefficient of 0.048 for the reduced-form model. However, the large standard errors for both models highlight the empirical challenge in identifying the mean reversion speed. The long-run volatility is 13.4% for the Heston (1993) model with a leverage effect where the correlation between asset value and volatility is –93.6%. In the Pan and Singleton (2008) model, the volatility of the default process is about 36%, and the long-run default intensity is 56.6% (exp(–0.569)), implying an average default jump approximately every two years $(1 / \hat{\theta}_\lambda = 1.8)$.

Exhibit 5 also reports the goodness of the fit as measured by the in-sample R-squared. The high R-squared reflects small pricing errors with low volatility. For the Heston (1993) model, the definition of leverage matters. Using the market value of leverage, we observe an in-sample R-squared of 95%, higher than the calibration under the book value of leverage (88.6%). The R-squared is even higher for the Pan and Singleton (2008) model, reaching 98.5%. Evidently, deep neural networks not

[15]For structural models, we also scale the observed leverage.

EXHIBIT 6

Graphical Representation of Model Calibration



NOTES: This exhibit plots the fit of the Pan and Singleton (2008) model, Heston (1993) model with the book value of leverage, and Heston (1993) model with the market value of leverage. Panels 6A, 6B, and 6C are scatter plots for the fit of the full term structure (in logarithms). The black 45-degree lines represent a perfect fit. Panels 6D, 6E, and 6F plot the endogenous state variables: the stochastic volatility in the Heston (1993) model and the stochastic default intensity in the Pan and Singleton (2008) model. Both models are calibrated to weekly data using the NN-UKF approach on the term structure of CDS spreads of American Express (AXP) from January 2004 to September 2014.

only can learn complex models of credit risk but also can be effectively employed for model calibration to historical data.

In Panels 6A, 6B, and 6C, we plot the relation between the fitted CDS and the actual CDS spreads. Because the R-squared measures are so high, we display the scatter plots in log-log form to better visualize any mispricing. For each scatter plot, we include a 45-degree line where data points would lie if the fitted spreads perfectly matched the actual spreads. Overall, few deviations from the 45-degree line occur across the five maturities for both the reduced-form model and the structural model with market leverage. Across models, the largest pricing errors are evident for short-term spreads. Furthermore, by combining a properly designed deep neural network with the appropriate inputs (such as market leverage instead of book leverage), the Heston (1993) model can achieve performance similar to that of the statistical reduced-form model of Pan and Singleton (2008).

In Panels 6D, 6E, and 6F (Exhibit 6), we plot the state variables. Our proposed NN-UKF approach generates endogenous state variables that capture relevant time variations. The state variables experience meaningful variation across peaceful and turbulent times; both the stochastic volatility of the Heston (1993) model and the

stochastic default intensity of the Pan and Singleton (2008) model are particularly high in the Great Recession of 2008.

CONCLUSION

In this article, we introduce deep learning into the credit risk modeling literature. We demonstrate that deep neural networks can learn complex models of credit risk with a high degree of accuracy. Using simulated data, we train deep learning models to capture the pricing relationship that maps model parameters into credit spreads. Applied on 10 different models of credit risk, our deep learning approach achieves predictive sample R-squared of more than 99%. Moreover, our deep learning approach provides a more computationally efficient alternative to traditional pricing functions: deep neural networks can build a full term structure of credit spreads about 100 to 240 times faster than the pricing functions.

We apply our deep learning approach to model calibration. We combine trained deep neural networks with the unscented Kalman filter to calibrate both structural and reduced-form credit risk models. Our NN-UKF approach attains in-sample R-squared of 88.6% and 95% for the structural model of Heston (1993) using book leverage and market leverage, and 98.5% for the reduced-form model of Pan and Singleton (2008).

Our findings represent an advancement in the credit risk modeling literature. Our results suggest that researchers can approximate complex models of credit risk in offline training and use them in an online calibration. This approximation can spare the researcher from computationally expensive numerical techniques when the model does not allow closed-form solutions. Although some initial computational cost is incurred in training the deep learning models, these trained models can save significant time and computing resources in subsequent applications.

Our deep learning approach can be extended in several directions. Interest rate models also employ (affine) reduced-form models to price, for example, the term structure of Treasury rates (Duffie and Kan 1996). Moreover, an extensive body of literature addresses modeling baskets of CDSs, or CDS indexes such as the CDX and iTraxx indexes (Brigo, Pallavicini, and Torresetti 2010). From a modeling perspective, one challenge is properly capturing the default correlation, usually described by a normal copula. Numerical precision becomes more relevant when the researcher is modeling the risk embedded in tranches of the index, such as collateral debt obligations. Precision is usually achieved at the expense of intensive numerical procedures, such as the systemic expected loss in Manzo and Picca (2020) that requires importance sampling to generate enough rare events. Future research can adapt our approach to approximate these complex functions.

Given the flexibility of deep learning models and their strong predictive power, we are confident that appropriately specified deep learning models can be applied to broader settings in economics and finance research. For example, the deep learning approach may be used to approximate other models with time-varying asset volatility (Du, Elkamhi, and Ericsson 2019), first passage time (Black and Cox 1976), optimal capital structure (Leland and Toft 1996), and multifactor approaches (e.g., Kelly, Manzo, and Palhares 2020).

APPENDIX

NEURAL NETWORK–BASED UNSCENTED KALMAN FILTER

The approach we propose for the model's calibration to historical data is the unscented Kalman filter (Wan and Van Der Merwe 2000) where the pricing equation is replaced by the

trained neural networks, an approach we call NN-UKF. We choose this approach because it is flexible to the inclusion of the market price of risk in the dynamics of the state variables. However, our calibration exercise is under the risk-neutral probability measure.

We cast the models with stochastic factors into a state-space representation where each process is a state variable and where the pricing function is the measurement equation. In particular, we assume that the measurement equation is

$$\mathbf{y}_t = NN_j\left(S_t \mid \Theta\right) + \eta_t \tag{A1}$$

where y_t is the vector of observed credit spreads; $NN_j(\cdot)$ is the trained neural network for model j, parametrized by the set Θ; S_t is the vector of state processes; and $\eta \sim N(0, \sigma_\eta)$ assumes homoskedastic pricing errors. For the Heston (1993) model (SV) and the Pan and Singleton (2008) model, we use discretized versions of the volatility and default intensity dynamics, respectively. The actual state equations are then their respective means and variances, conditional on the $(t-1)$ information set I_{t-1}.

We compute the log-likelihood using the unscented transformation (UT) that employs a set of sigma points to calculate the first two moments of the state and measurement equations. In formulas, given initial states $\boldsymbol{\mu}_{t-1} = S_{t-1}$ and covariance P_{t-1}, we form a matrix X_{t-1} of $2L + 1$ sigma vectors, $\chi_{i,t-1}$, with corresponding weights W_i, such that

$$\begin{aligned}
\mathcal{X}_{0,t-1} &= \mu_{t-1} \\
\mathcal{X}_{i,t-1} &= \mu_{t-1} + \left(\sqrt{(L+\lambda)\mathbf{P}_{V,t-1}}\right)_i \quad i = 1, \ldots, L \\
\mathcal{X}_{i,t-1} &= \mu_{t-1} - \left(\sqrt{(L+\lambda)\mathbf{P}_{V,t-1}}\right)_{i-L} \quad i = L+1, \ldots, 2L \\
W_0^{(m)} &= \lambda / (L+\lambda) \\
W_0^{(c)} &= \lambda / (L+\lambda) + \left(1 - \alpha^2 + \beta\right) \\
W_i^{(m)} &= W_i^{(c)} = 1 / \left\{2(L+\lambda)\right\} \quad i = 1, \ldots, 2L
\end{aligned} \tag{A2}$$

where W is a set of weights determined by the scaling parameters $\lambda = \alpha^2(L + k) - L$, α = 1e-3, $k = 0$, and $\beta = 2$ and where the latter captures information on the Gaussian distribution of the states.

By propagating these sigma points through the state equation, we obtain the time $(t-1)$ ex ante forecast of time t states and measurements, expressed as

$$\begin{aligned}
\mathbf{S}_{t|t-1} &= \mu\left(\mathcal{X}_{t-1}\right) \\
\mathbf{Y}_{t|t-1} &= NN\left(\mathcal{S}_{t|t-1} \mid \Theta\right)
\end{aligned} \tag{A3}$$

where the sample mean and covariance are the weighted sample mean and covariance of the posterior sigma points, that is

$$\begin{aligned}
\bar{\mathbf{y}}_t &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,t|t-1} \\
\mathbf{P}_{\mathbf{y},t} &= \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{Y}_{i,t|t-1} - \bar{\mathbf{y}}_t\right]\left[\mathcal{Y}_{i,t|t-1} - \bar{\mathbf{y}}_t\right]' \\
\bar{\mu}_t &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{S}_{i,t|t-1} \\
\bar{\mathbf{P}}_{S,t} &= \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{S}_{i,t|t-1} - \bar{\mu}_t\right]\left[\mathcal{S}_{i,t|t-1} - \bar{\mu}_t\right]'
\end{aligned} \tag{A4}$$

Given these estimated moments, the time t log-likelihood $llk_t(\Theta)$ is

$$llk_t(\Theta) = -\frac{1}{2}\ln|P_{y,t}| - \frac{1}{2}(CIV_t - \bar{\mathbf{y}}_t)' \mathbf{P}_{y,t}^{-1}(CIV_t - \bar{\mathbf{y}}_t) \tag{A5}$$

and the state update is

$$\begin{aligned} \mu_t &= \bar{\mu}_t + \mathcal{K}_t(CS_t^{obs} - \bar{\mathbf{y}}_t) \\ \mathbf{P}_{S,t} &= \bar{\mathbf{P}}_{S,t} + \mathcal{K}_t \mathbf{P}_{yS,t} \mathcal{K}_t^{'} \end{aligned} \tag{A6}$$

where CS_t^{obs} is the time t observed term structure (1, 3, 5, 7, and 10 years), $\mathcal{K}_t = \mathbf{P}_{yS,t} \times \mathbf{P}_{y,t}^{-1}$ is the Kalman gain, and $\mathbf{P}_{yS,t} = \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{S}_{i,t|t-1} - \bar{\mu}_t\right]\left[\mathcal{Y}_{i,t|t-1} - \bar{\mathbf{y}}_t\right]'$ is the covariance between the states and the measurement equation. Then, by replacing $\boldsymbol{\mu}_t$ and $\mathrm{P}_{S,t}$ as initial values, we can compute new sigma points and obtain the likelihood at $t + 1$.

As the last step, we estimate the parameters by maximizing the sum of llk_t, expressed as

$$\hat{\Theta} : \max \sum_{t=0}^{N} llk_t(\Theta) \tag{A7}$$

In the practical implementation, we follow Kelly, Manzo, and Palhares (2020) and employ a parallel unscented Kalman filter. The goal is further accelerating the calibration. For each time t, we run isolated unscented Kalman filters that function as follows: given the initial states, we obtain the pricing errors, update the states, and recompute the pricing errors. We repeat this process about five times and then select the log-likelihood that corresponds to the minimum sum of squared errors. Usually, a few iterations are enough to observe a significant (exponential) reduction in the pricing errors.

DISCLAIMER, FUNDING, AND ACKNOWLEDGMENTS

The views expressed are those of the individual authors and do not necessarily reflect official positions of Kepos Capital LP (Kepos). This article is not an offer to sell, or a solicitation of an offer to buy, any investment product or services offered by Kepos. Kepos does not opine on the accuracy or completeness of the information contained herein, and any information provided by third parties is not independently verified by Kepos. All errors are our own.

The authors wish to thank a number of colleagues for helpful comments: Mark Carhart, Giorgio De Santis, Mark Huang, Arlen Khodadadi, Mike Lock, and participants at the Kepos Capital seminar and the QuantInsti seminar.

REFERENCES

Altman, E. I., B. Brady, A. Resti, and A. Sironi. 2005. "The Link between Default and Recovery Rates: Theory, Empirical Evidence, and Implications." *The Journal of Business* 78 (6): 2203–2228.

Black, F., and J. C. Cox. 1976. "Valuing Corporate Securities: Some Effects of Bond Indenture Provisions." *The Journal of Finance* 31 (2): 351–367.

Black, F., and M. Scholes. 1973. "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy* 81 (3): 637–654.

Brigo, D., A. Pallavicini, and R. Torresetti. 2010. *Credit Models and the Crisis: A Journey into CDOs, Copulas, Correlations and Dynamic Models*. John Wiley & Sons.

Carr, P., and D. Madan. 1999. "Option Valuation Using the Fast Fourier Transform." *Journal of Computational Finance* 2 (4): 61–73.

Carr, P., and L. Wu. 2007. "Theory and Evidence on the Dynamic Interactions between Sovereign Credit Default Swaps and Currency Options." *Journal of Banking & Finance* 31 (8): 2383–2403.

Cybenko, G. 1989. "Approximation by Superpositions of a Sigmoidal Function." *Mathematics of Control, Signals and Systems* 2 (4): 303–314.

Du, D., R. Elkamhi, and J. Ericsson. 2019. "Time-Varying Asset Volatility and the Credit Spread Puzzle." *The Journal of Finance* 74 (4): 1841–1885.

Duan, J. C. 1999. "Conditionally Fat-Tailed Distributions and the Volatility Smile in Options." Rotman School of Management, University of Toronto, Working Paper.

Duffie, D. 1999. "Credit Swap Valuation." *Financial Analysts Journal* 55 (1): 73–87.

Duffie, D., and R. Kan. 1996. "A Yield-Factor Model of Interest Rates." *Mathematical Finance* 6 (4): 379–406.

Feldhütter, P., and S. M. Schaefer. 2018. "The Myth of the Credit Spread Puzzle." *The Review of Financial Studies* 31 (8): 2897–2942.

Feng, G., S. Giglio, and D. Xiu. 2020. "Taming the Factor Zoo: A Test of New Factors." *The Journal of Finance* 75 (3): 1327–1370.

Feng, G., N. Polson, and J. Xu. 2019. "Deep Learning in Characteristics-Sorted Factor Models." SSRN 3243683.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press.

Gu, S., B. T. Kelly, and D. Xiu. 2019. "Autoencoder Asset Pricing Models." *Journal of Econometrics* 222 (1): 429–450.

——. 2020. "Empirical Asset Pricing via Machine Learning." *The Review of Financial Studies* 33 (5): 2223–2273.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Heston, S. L. 1993. "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options." *The Review of Financial Studies* 6 (2): 327–343.

Horvath, B., A. Muguruza, and M. Tomas. 2019. "Deep Learning Volatility." SSRN 3322085.

Huang, J.-Z., Z. Shi, and H. Zhou. 2020. "Specification Analysis of Structural Credit Risk Models." *The Review of Finance* 24 (1): 45–98.

Hutchinson, J. M., A. W. Lo, and T. Poggio. 1994. "A Nonparametric Approach to Pricing and Hedging Derivative Securities via Learning Networks." *The Journal of Finance* 49 (3): 851–889.

Jarrow, R. A., and S. M. Turnbull. 1995. "Pricing Derivatives on Financial Securities Subject to Credit Risk." *The Journal of Finance* 50 (1): 53–85.

Kelly, B. T., G. Manzo, and D. Palhares. 2020. "Credit-implied volatility." Accessed September 7, 2021 from https://www.aqr.com/Insights/Research/Working-Paper/Credit-Implied-Volatility.

Kingma, D. P., and J. Ba. 2014. "Adam: A Method for Stochastic Optimization." *arXiv* preprint. arXiv1412.6980.

Kou, S. G. 2002. "A Jump-Diffusion Model for Option Pricing." *Management Science* 48 (8): 1086–1101.

Leland, H. E., and K. B. Toft. 1996. "Optimal Capital Structure, Endogenous Bankruptcy, and the Term Structure of Credit Spreads." *The Journal of Finance* 51 (3): 987–1019.

Liu, S., A. Borovykh, L. A. Grzelak, and C. W. Oosterlee. 2019. "A Neural Network-Based Framework for Financial Model Calibration." *Journal of Mathematics in Industry* 9 (1): 9.

Liu, S., C. W. Oosterlee, and S. M. Bohte. 2019. "Pricing Options and Computing Implied Volatilities Using Neural Networks." *Risks* 7 (1): 16.

Longstaff, F. A., J. Pan, L. H. Pedersen, and K. J. Singleton. 2011. "How Sovereign Is Sovereign Credit Risk?" *The American Economic Review* 3: 75–103.

Manzo, G. 2013. "Political Uncertainty, Credit Risk Premium and Default Risk." PhD dissertation, August 31. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.439.5419&rep=rep1&type=pdf.

Manzo, G., and A. Picca. 2020. "The Impact of Sovereign Shocks." *Management Science* 66 (7): 3113–3132.

Manzo, G., and P. Veronesi. 2016. "Sovereign Credit Risk." *Handbook of Fixed-Income Securities*: 561–586.

Merton, R. C. 1974. "On the Pricing of Corporate Debt: The Risk Structure of Interest Rates." *The Journal of Finance* 29 (2): 449–470.

——. 1976. "Option Pricing when Underlying Stock Returns Are Discontinuous." *Journal of Financial Economics* 3: 125–144.

Pan, J., and K. J. Singleton. 2008. "Default and Recovery Implicit in the Term Structure of Sovereign CDS Spreads." *The Journal of Finance* 63 (5): 2345–2384.

Rosenblatt, F. 1958. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review* 65 (6): 386.

Ruf, J., and W. Wang. 2019. "Neural Networks for Option Pricing and Hedging: A Literature Review." SSRN 3486363.

Wan, E. A., and R. Van Der Merwe. 2000. "The Unscented Kalman Filter for Nonlinear Estimation." In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium,* Cat. No. 00EX373: 153–158.

Yan, S. 2011. "Jump Risk, Stock Returns, and Slope of Implied Volatility Smile." *Journal of Financial Economics* 99 (1): 216–233.