Quantifying neural network uncertainty under volatility clustering

Anonymous authors
Paper under double-blind review

Abstract

Time-series with complex structures pose a unique challenge to uncertainty quantification methods. Time-varying variance, such as volatility clustering as seen in financial time-series, can lead to large mismatch between predicted uncertainty and forecast error. In this work, we propose a novel framework to deal with uncertainty quantification under the presence of volatility clustering, building and extending the recent methodological advances in uncertainty quantification for non-time-series data. To illustrate the performance of our proposed approach, we apply it to two types of datasets: a collection of non-time-series data to show the general applicability of our framework and its ability to quantify the uncertainty better than the state-of-the art methods; and to two sets of financial time-series exhibiting volatility clustering: cryptocurrencies and U.S. equities.

1 Introduction

Neural networks for regression problems are typically trained using mean squared error (MSE) as the loss function and provide point estimates for the mean of a distribution conditional on the input (see Goodfellow et al., 2016). However, in certain applications, both the conditional mean and the uncertainty of the conditional mean are equally important. Example applications include safety critical applications such as self-driving cars (Bojarski et al., 2016; Amini et al., 2020) and in applications that involve trade-off of risk and reward, such as portfolio selection. To motivate the discussion, consider the following simple thought experiment. Suppose an investor has a model that can perfectly forecast next day's asset returns and that the investor's goal is to maximize terminal wealth. Then, on each day, the most rational decision would be to place all of the investor's wealth into the asset with the highest expected return on the next day. Next, suppose that the investor's model is a noisy estimator of future asset returns. Then, the investor may choose to diversify across multiple assets. This has led to the development of various models for optimal bet allocation that depend on some measures of risk or probability of outcome, such as Kelly criterion (Kelly, 1956; Byrnes & Barnett, 2018) and Bayesian-based portfolio optimization (Black & Litterman, 1991). Forecast uncertainty of expected return is an important input into the portfolio construction process.

In this work, we are concerned with neural network uncertainty quantification for complex time-series structures. In particular, we focus on time-series that exhibit *time-varying variance*, such as time-series of asset returns. In other words, asset returns exhibit irregular bursts of high volatility that cluster in time (termed *volatility clustering*; Cont, 2001). We argue that this feature is an alternative form of *out-of-distribution* observation in time-series applications and is a relatively less studied area of the uncertainty quantification literature. In this work, we propose a framework for uncertainty quantification under volatility clustering. To illustrate our contributions, we apply our proposed method to cryptocurrency and U.S. equities time-series forecasting, and highlight the usefulness of our proposed method. Cryptocurrencies are an emerging class of digital assets. They are highly volatile and frequently exhibit price bubbles (Fry & Cheah, 2016; Hafner, 2018; Chen & Hafner, 2019; Núñez et al., 2019; Petukhina et al., 2021), with large volumes of high frequency data (e.g., prices in hourly intervals) freely available from major exchanges. This makes cryptocurrencies an ideal testbed for uncertainty quantification methodologies in financial applications. Given the extreme levels of volatility, we view cryptocurrencies as one of the most challenging datasets for this type of applica-

tion. However, a comparison in U.S. equities is also provided which illustrates performance in conventional financial time-series.

Recently, significant advances in neural network uncertainty quantification have been made (see Gawlikowski et al., 2021 for a recent survey). In particular, using a neural network to generate parameters of a conditional distribution that is assumed to have generated the data (Lakshminarayanan et al., 2017; Amini et al., 2020) offers an attractive trade-off between adequately quantifying uncertainty and avoiding the computational cost of a full Bayesian treatment[1]. These methods offer a familiar procedure to *maximum likelihood estimation*. In Lakshminarayanan et al. (2017) (the *Ensemble* method), the objective is to minimize the negative log-likelihood (NLL) of a Normal distribution, parameterized by μ and σ^2, which are outputted by the neural network. Parameter σ^2 in this likelihood function models data uncertainty (also called statistical or aleatoric uncertainty) but is incapable of modelling the uncertainty in model parameters (also called systemic or epistemic uncertainty; Kendall & Gal, 2017). Thus, Lakshminarayanan et al. (2017) proposed to use an ensemble of networks with random initialization to approximate model uncertainty. To address this, Amini et al. (2020) (the *Evidential* method) proposed to place an evidential prior[2], the Normal-Inverse-Gamma distribution (NIG), on μ, σ^2. In Evidential, μ is assumed to be drawn from a Normal distribution with unknown mean γ and scaled variance $\sigma^2\nu^{-1}$. In this construct, model uncertainty is reflected by the uncertainty in γ, which is assumed to be a fraction of σ^2 and is drawn from an Inverse Gamma distribution. The fraction is controlled by ν, which is learnt from the data and varies according to the strength of the data. The resultant marginal likelihood is a non-standardized Student's t-distribution. This mimics a Bayesian setup and offers an intuitive interpretation of the model mechanics — due to uncertainty in the model parameters, the tails of the marginal likelihood are heavier than the a Normal distribution. This has the effect of regularizing the network and provides an avenue of modelling model uncertainty. Ensemble and Evidential require only minimal modifications to a conventional neural network architecture — requiring only new loss function (NLL of the marginal distribution) and output layer. These works demonstrated state-of-the-art performance in quantifying uncertainty in a range of benchmark datasets.

However, in analyzing the marginal distribution of Evidential, we observe that ν relates ambiguously to the scale parameter of the t-distribution. We argue that this may impede first-order optimization methods, such as stochastic gradient descent (SGD), in traversing the loss surface and thus hamper neural network learning. The output layer produces hyperparameters of the marginal distribution, which are computed as a linear combination of the latent representation outputted by the last hidden layer. We consider this feature to be a weakness of these approaches as the latent representation has to provide a sufficiently rich encoding to linearly derive all hyperparameters of the distribution. Moreover, these advances are based on conventional applications of deep learning and remain largely untested in financial applications.

In this work, we show that the current state-of-the-art in quantifying uncertainty of time-series exhibiting non-stationary variance can be further advanced. We combine and extend Ensemble and Evidential into a framework for quantifying forecast uncertainty of non-stationary time-series (the *Combined* method). We propose a novel parameterization of the problem through variance scaling. We argue that modelling forecast uncertainty through the introduction of a variance scaling factor provides a simpler alternative to the NIG prior in Evidential. Variance scaling is achieved through placing a Gamma prior solely on variance of the Normal distribution, rather than both the mean and variance in Evidential. This results in an unambiguous relationship between the scaling factor and the scale parameter of the resultant marginal t-distribution. We show through experimental results that our proposed method provides competitive, if not superior, forecast uncertainty quantification performance to the state-of-the-art methods. We propose a novel architecture to model parameters of the prior distribution, where parameters of the prior distribution are modelled using disjoint subnetworks. We recognize the significant potential of model averaging in improving forecast accuracy in time-series forecasting. Whilst Lakshminarayanan et al. (2017) did not use ensembling for the purpose of improving forecast accuracy. We propose to use model averaging with an evidential prior to simultaneously improve forecast accuracy and uncertainty quantification provided by our methodology. We argue that this architecture will improve uncertainty quantification performance where uncertainty has

[1]Such as the Bayesian neural network (Neal, 1996).

[2]In contrast to conventional priors in Bayesian inference where the modeller has to specify the parameters of the prior distribution, the evidential prior (e.g., NIG in Evidential) learns these hyperparameters from the data.

complex relationships with the input. We use the information of squared returns to inform the neural network of time-varying variance and show that this framework vastly improves uncertainty quantification of financial time-series forecasts. We also show that this framework can benefit non-time-series uncertainty quantification problems and illustrates this on three real world datasets: the University of California Irvine Machine Learning Repository (UCI) benchmark datasets (non-time-series), previously analyzed in Hernández-Lobato & Adams (2015); Gal & Ghahramani (2016); Lakshminarayanan et al. (2017); Amini et al. (2020), and two financial time-series datasets (cryptocurrency and U.S. equities).

2 Preliminaries

2.1 Problem setup

Consider an investor making iterative forecasts of asset returns. At every period $t \in \{1, \ldots, T\}$, an investor observes price history up to t and uses the preceding $\{K \in \mathbb{Z} \,|0 < K < t\}$ period returns to forecast one-step ahead returns. We define an asset's return at time t as the log difference in price $r_t = \log p_t - \log p_{t-1}$ and, consistent with empirical findings in finance literature (Pesaran & Timmermann, 1995; Cont, 2001), we assume that the data generation process (DGP) is time-varying:

$$r_t \sim \mathrm{N}(\mu_t, \sigma_t^2). \tag{1}$$

Let $\boldsymbol{\zeta}_t = (\mu_t, \sigma_t^2)$, $\boldsymbol{x}_{t-1} = \{r_{t-K}, r_{t-K+1}, \ldots, r_{t-1}\}$ be a K-length input sequence[3] using returns up to $t-1$ and $y_{t-1} = r_t$ be forward one period return. The training dataset is comprised of $\mathcal{D}_t = \{(\boldsymbol{x}_{\mathtt{t}-1}, y_{\mathtt{t}-1}) | \, \mathtt{t} \in \mathbb{N} : \mathtt{t} \leq t\}$ input-output pairs[4] and our goal is to forecast y_t (which corresponds to r_{t+1}). At each t, the investor's goal is to solve the optimization problem[5],

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}^*} - \sum_{\mathtt{t}=K}^{t-1} \log \mathrm{p}(y_{\mathtt{t}} | F(\boldsymbol{x}_{\mathtt{t}}; \boldsymbol{\theta}^*)), \tag{2}$$

where $F(\boldsymbol{x}; \boldsymbol{\theta})$ is a neural network with input $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$, $\boldsymbol{\theta} = \bigcup_{\ell=1}^{L} \{\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)}\}$ is the set of network weights and biases and, in this context, $\mathrm{p}(y|F(\boldsymbol{x}; \boldsymbol{\theta}))$ is the likelihood of observing y based on the outputs of neural network $F(\cdot; \cdot)$ and the assumed marginal distribution. In other words, the investor is concerned with recovering the parameters $\hat{\boldsymbol{\zeta}}_t = (\hat{\mu}_t, \hat{\sigma}_t^2) = F(\boldsymbol{x}_t; \boldsymbol{\theta}_t)$ that are most likely to have generated the observed data. In this setup, $\hat{\sigma}_t^2$ can be interpreted as an estimate of data uncertainty and is an estimate of the contemporaneous variance of the DGP at time t. Note that this is different from forecasting variance (e.g., forward 30-day volatility), which is the average squared deviation from the mean over a desired horizon.

There are two parts to this problem. The first part concerns advancing methods of uncertainty quantification across general applications. In Section 4.1, we show that our proposed approach can still benefit non-time-series problems in spite of it being designed to deal with a series of data points indexed in time order and exhibiting volatility clustering. The second part concerns uncertainty quantification specifically for time-series that exhibit volatility clustering, and is detailed in Section 3.2.

2.2 Related work

As discussed in Section 1, forecast uncertainty has an important role in many applications. Such quantity is easy to obtain for a statistical model such as linear regression. However, *classical* neural networks for regression problems are typically trained using MSE and provide point estimates for the mean prediction conditional on the input. As a modeller (in our case, an investor), one is concerned with *predictive uncertainty* (Gawlikowski et al., 2021). This is the total uncertainty around the point estimate. Predictive uncertainty can

[3]For illustrative purposes, we have stated that the sequence only contains returns r_t. However, as discussed in Section 3.2, we also include squared returns r_t^2 as part of the input sequence.

[4]Note that at each portfolio selection period t, the training set can at most contain data up to $t-1$ as we have not yet observed r_{t+1}.

[5]For clarity, the case of a single asset is shown. At each t, there are N assets and the dataset is typically in a $t \times N$ layout. It is easy to see the generalization of Equation 2 over N assets, where the average loss is calculated over $(t-K-1) \times N$ instances.

be decomposed into (Hüllermeier & Waegeman, 2019; Gawlikowski et al., 2021): data uncertainty (relating to uncertainty in the data such as input noise), and model uncertainty (relating to uncertainty on model parameters).

A Bayesian neural network (BNN) is a full probabilistic interpretation of neural network, by placing priors on network weights and inducing a distribution over a parametric set of functions (MacKay, 1992; Neal, 1995; Gal, 2016). Modern BNNs can be trained using Markov Chain Monte Carlo (MCMC, e.g., the Metropolis-Hastings algorithm; Hastings, 1970) and *Variational Inference* techniques (Jospin et al., 2022). Jospin et al. (2022) notes four advantages of using BNNs over classical neural networks, with two being relevant to uncertainty quantification. First, Bayesian methods provide a natural approach to uncertainty quantification and are better *calibrated* than classical neural networks (Mitros & Namee, 2019; Kristiadi et al., 2020; Ovadia et al., 2019; Jospin et al., 2022). Second, BNN allows distinguishing between model uncertainty and data uncertainty. However, despite their advantages, MCMC-based methods are computationally expensive (Quiroz et al., 2019). Thus, limiting the applicability of BNNs.

Recent advances have focused on predicting the conditional distribution that is most likely to have generated the data and thus bridging the gap between BNNs and classical neural networks. In the Ensemble method, the data is assumed to be drawn from a Normal distribution with parameters μ and σ^2 (Lakshminarayanan et al., 2017). The neural network is modified to output both μ and σ^2. In this setup, σ^2 models data uncertainty and is incapable of quantifying model uncertainty. Lakshminarayanan et al. (2017) addressed this by using an ensemble of neural networks with randomly initialized weights. Each network settles in a different local minima and produces different μ and σ^2 for the same input. The variance of μ across the ensemble thus provides an estimate of model uncertainty. Addressing this shortcoming, Amini et al. (2020) (the Evidential method) proposed to place an evidential prior, the NIG distribution, on the model parameters μ, σ^2 of the Normal data distribution:

$$\begin{aligned} \text{Data} : y &\sim \mathrm{N}(\mu, \sigma^2) \\ \text{NIG prior} : \mu &\sim \mathrm{N}(\gamma, \sigma^2\nu^{-1}), \quad \sigma^2 \sim \mathrm{InvGam}(\alpha, \beta), \end{aligned} \tag{3}$$

where μ is assumed to be drawn from a Normal prior distribution with unknown mean γ and scaled variance $\sigma^2\nu^{-1}$, ν is a scaling factor for σ^2, InvGam (or IG) is the Inverse-Gamma distribution, and shape $\alpha > 1$ and scale $\beta > 0$ parameterize the Inverse-Gamma distribution[6]. We require $\alpha > 1$ to ensure the mean of the marginal distribution is finite.

In this construct, model uncertainty is reflected by the uncertainty in μ, which is assumed be a fraction of σ^2 and is itself assumed to be drawn from an Inverse-Gamma distribution. This fraction is controlled by ν, which is learnt from the data and, in an abstract sense, varies according to the amount of information in the data. Parameter ν is interpreted as the number of virtual observations for the mean parameter μ. In other words, ν virtual instances of μ are assumed to have been observed in determining the prior variance of μ (Jordan, 2009; Amini et al., 2020).

For the NIG prior in 3, the marginal distribution of μ after integrating out σ^2 is a non-standardized Student's t-distribution (Bernardo & Smith, 2000),

$$\begin{aligned} \mathrm{p}(\mu|\gamma, \nu, \alpha, \beta) &= \int_{\sigma^2=0}^{\infty} \mathrm{p}_{\mathrm{N}}(\mu|\gamma, \sigma^2\nu^{-1})\mathrm{p}_{\mathrm{IG}}(\sigma^2|\alpha, \beta)\,\mathrm{d}\sigma^2 \\ &= \mathrm{St}\left(\gamma, \frac{\beta}{\nu\alpha}, 2\alpha\right), \end{aligned} \tag{4}$$

using the fact that $\sigma^2 \sim \mathrm{InvGam}(\alpha, \beta)$ corresponds to $\sigma^{-2} \sim \mathrm{Gam}(\alpha, \beta)$. Hence, assigning a $\mathrm{Gam}(\alpha, \beta)$ prior to precision σ^{-2} in (Equation (3)) gives the Normal-Gamma (NG) prior and is equivalent to assigning $\mathrm{InvGam}(\alpha, \beta)$ to σ^2 which gives the NIG prior. The variance of this t-distribution is $\frac{\beta}{\nu(\alpha-1)}$. Predictions

[6]Time index t has been omitted for brevity and legibility. Note that variables in this section are indexed by time for each asset: $\{y_t, r_t, \mu_t, \sigma_t^2, \gamma_t, \nu_t, \alpha_t, \beta_t\}$.

based on the NIG prior can be computed as (Amini et al., 2020),

$$\begin{aligned} \text{Prediction} : \mathrm{E}[\mu] &= \gamma \\ \text{Data uncertainty} : \mathrm{E}[\sigma^2] &= \tfrac{\beta}{\alpha-1} \\ \text{Model uncertainty} : \mathrm{Var}[\mu] &= \tfrac{\beta}{\nu(\alpha-1)}. \end{aligned} \tag{5}$$

The marginal variance $\mathrm{Var}[\mu]$ refers to the variance of the marginal t-distribution in (Equation (4)) for the NIG prior. We note that ν can also be interpreted as a factor that attributes uncertainty between data uncertainty ($\frac{\beta}{\alpha-1}$) and model uncertainty ($\frac{\beta}{\nu(\alpha-1)}$). If $\nu = 1$, then total uncertainty is evenly split between model and data.

Whilst not the focus of Amini et al. (2020), we note that model uncertainty can be further decomposed into uncertainties attributable to parameters μ and σ^2,

$$\begin{aligned} \text{Model } \mu \text{ uncertainty} : \mathrm{Var}[\mu|\sigma^2] &\approx \tfrac{\beta}{\nu\alpha} \\ \text{Model } \sigma^2 \text{ uncertainty} : \mathrm{Var}[\mu] - \mathrm{Var}[\mu|\sigma^2] &\approx \tfrac{\beta}{\nu\alpha(\alpha-1)}, \end{aligned} \tag{6}$$

where the difference between the marginal and conditional variances of μ gives the uncertainty of σ^2. Moreover, since $\mu|\sigma^2$ is normally distributed with $\mathrm{Var}[\mu|\sigma^2] = \sigma^2/\nu$ (from Equation (3)), and $\mathrm{E}[\sigma^{-2}] = \mathrm{E}[\frac{1}{\sigma^2}] \approx \frac{1}{\mathrm{E}[\sigma^2]} = \alpha/\beta$ (from the Gamma distribution of σ^{-2}). This leads to $\mathrm{Var}[\mu|\sigma^2] \approx \frac{\beta}{\nu\alpha}$. We note that this ability to finely attribute model uncertainty to μ and σ^2 is a strength of the Evidential method.

In this construct, the marginal distribution of y after integrating out μ and σ^2 is a non-standardized Student's t-distribution (Amini et al., 2020),

$$\begin{aligned} \mathrm{p}(y|\gamma,\nu,\alpha,\beta) &= \int_{\sigma^2=0}^{\infty}\int_{\mu=-\infty}^{\infty} \mathrm{p_N}(y|\mu,\sigma^2)\mathrm{p_{NIG}}(\mu,\sigma^2|\gamma,\nu,\alpha,\beta)\,\mathrm{d}\mu\,\mathrm{d}\sigma^2 \\ &= \mathrm{St}\left(y;\gamma,\frac{\beta(1+\nu)}{\nu\alpha},2\alpha\right). \end{aligned} \tag{7}$$

Variance of this t-distribution is $\frac{\beta(1+\nu)}{\nu(\alpha-1)}$, which corresponds to the sum of model and data uncertainty,

$$\mathrm{Var}[y] = \frac{\beta}{\alpha-1} + \frac{\beta}{\nu(\alpha-1)} = \frac{\beta(1+\nu)}{\nu(\alpha-1)}. \tag{8}$$

The corresponding NLL of Equation (7) is (Amini et al., 2020),

$$\begin{aligned} \mathcal{L}_{\mathrm{NIG}}(y|\boldsymbol{\zeta}) = {} & \tfrac{1}{2}\log\left[\tfrac{\pi}{\nu}\right] - \alpha\log\left[2\beta(1+\nu)\right] \\ & + (\alpha+\tfrac{1}{2})\log\left[(y-\gamma)^2\nu + 2\beta(1+\nu)\right] + \log\left[\frac{\Gamma(\alpha)}{\Gamma(\alpha+\frac{1}{2})}\right], \end{aligned} \tag{9}$$

where $\boldsymbol{\zeta} = \{\gamma,\nu,\alpha,\beta\}$ are parameters of the posterior distribution. This mimics a Bayesian setup, granting classical neural networks the ability to estimate both model and data uncertainty, and offers an intuitive interpretation of the model mechanics — due to uncertainty in the model parameters, the tails of the marginal likelihood are heavier than a Normal distribution. This has the effect of regularizing the network and provides an avenue of estimating model uncertainty. As the distribution of asset returns has heavy tails (Cont, 2001), we argue that the marginal t-distribution also provides a better fit of the data. The implementation is remarkably simple — Equation (9) replaces MSE as the loss function (for a regression problem) and the final layer of the network is replaced with a layer that simultaneously outputs four parameters of the marginal distribution.

When analysing Equation (7), we note that ν appears in both the numerator and denominator of the scale parameter of the t-distribution in Equation (7) in the form of $1 + \frac{1}{\nu}$. We argue that this functional form increases the difficulty of a neural network to find the optimal solution as increasing "evidence" (ν) may not monotonically lead to a decrease in scale of the t-distribution. Motivated by this observation, we propose a simpler formulation which we detail in Section 3.1.

3 Uncertainty quantification under volatility clustering

3.1 Modelling forecast uncertainty using a scale mixture distribution

Our formulation of the problem is motivated by two observations. First, as discussed in Section 2.2, we argue that the functional form of ν in the scale of the marginal t-distribution could contribute to the difficulty in minimizing the t-distributed NLL by a first-order optimiser, such as SGD. Second, the NIG prior provides the ability to perform granular attribution of uncertainty to various parts of the model (e.g., Equation (5) and (6)), which comes at the cost of model complexity. We sought to propose a simpler formulation of the problem than Evidential while offering the ability to quantify predictive uncertainty, which is the type of forecast uncertainty that we are most concerned about in our motivating application.

To address these issues, we propose to simplify the model by formulating the problem as a scale mixture distribution[7] (SMD; Andrews & Mallows, 1974),

$$y \sim \mathrm{N}(\gamma, \sigma^2\nu^{-1}), \quad \nu \sim \mathrm{Gam}(\alpha, \beta), \tag{10}$$

where $\nu > 0$ is the scaling factor, Gam is the Gamma distribution, and $\alpha > 1$ and $\beta > 0$ are the shape and scale parameters of the Gamma distribution, respectively. Our proposed formulation effectively omits the prior on μ and places a prior on ν, the scaling factor of σ^2. We argue that uncertainty of variance can be modelled through either σ^2 or ν. In here, y is assumed to be drawn from $\mathrm{N}(\gamma, \sigma^2\nu^{-1})$, with mean γ and unknown variance $\sigma^2\nu^{-1}$ where ν is a latent variable that introduces uncertainty into the variance of the assumed Normal distribution of y. Relative to Equation (7), σ^2 replaces ν in the parameter set when taking the SMD approach as σ^2 has a richer interpretation — it directly indicates the scale of the conditional data distribution. Note that Equation (10) is equivalent to $\sigma^{-2} \sim \mathrm{Gam}(\alpha, \beta)$ as ν and σ^{-2} are indistinguishable in $\sigma^2\nu^{-1}$ but it is distinct from the NG prior as there is no Normal prior on μ in Equation (10).

The marginal distribution of a Normal distribution with unknown variance (Equation (10)) is a non-standardized t-distribution (derivation is provided in Appendix A),

$$\begin{aligned} \mathrm{p}(y|\gamma, \sigma^2, \alpha, \beta) &= \int_{\nu=0}^{\infty} \mathrm{p}_{\mathrm{N}}(y|\gamma, \sigma^2\nu^{-1})\mathrm{p}_{\mathrm{G}}(\nu|\alpha, \beta)\,\mathrm{d}\nu \\ &= \mathrm{St}\left(y; \gamma, \frac{\sigma^2\beta}{\alpha}, 2\alpha\right). \end{aligned} \tag{11}$$

Analogous to Equation (7), the shape parameter of this marginal Student's t-distribution is 2α. Equation (11) is similar to Equation (4) with y replacing μ, and it can be interpreted as the Normal distribution being "stretched out" into a heavier tailed distribution due to the uncertainty in its variance. Jointly, $\boldsymbol{\zeta} = (\gamma, \sigma^2, \alpha, \beta)$ are hyperparameters of the SMD distribution and are outputs of the neural network. This has the effect of regularizing the mean estimate (γ) and, similar to NIG, provides the ability to handle heavy tails of the distribution that characterize asset returns. We make two remarks. Firstly, there are essentially three free parameters in Equation (11) as $\sigma^2\beta$ together should be treated as one. We provide a more detailed discussion on this point in Section 3.2. Secondly, SMD encapsulates several well-known distributions as special cases. According to Andrews & Mallows (1974) and Choy & Chan (2008), in our case of $\alpha \neq \beta$, Equation (10) gives the *Pearson Type VII* distribution which can be re-expressed as Student's t-distribution in Equation (11). If $\alpha = \beta$, then Equation (10) is a Student's t-distribution with 2α degrees of freedom, and is Cauchy if $\alpha = \beta = 1$.

The corresponding NLL of Equation (11) (derivation is provided in Appendix A) is,

$$\mathcal{L}_{\mathrm{SMD}}(y|\boldsymbol{\zeta}) = \log\left[\frac{\Gamma(\alpha)}{\Gamma(\alpha + \frac{1}{2})}\right] + \tfrac{1}{2}\log[2\pi\sigma^2\beta] + (\alpha + \tfrac{1}{2})\log\left[\frac{(y-\gamma)^2}{2\sigma^2\beta} + 1\right]. \tag{12}$$

Then, $\mathcal{L}_{\mathrm{SMD}}$ is used in place of the marginal likelihood function in Equation (2), in which the neural network learns to output parameters in $\boldsymbol{\zeta}$.

[7] Time index t has been omitted for brevity and legibility. Note that variables in this section are indexed by time for each asset: $\{y_t, \gamma_t, \sigma_t^2, \nu_t, \alpha_t, \beta_t\}$. We use the same notations in Equation (10) as Equation (3) where the symbols have the same meaning to improve comparability.

In Equation (10), conditional on the scaling factor ν, the data is normal with variance given by the scale of the marginal t-distribution ($\frac{\sigma^2\beta}{\alpha}$). This variance gives the uncertainty of the data. Since the predictive uncertainty given by the variance of the marginal t-distribution contains both model and data uncertainties, the difference between predictive and data uncertainties gives the model uncertainty. This is illustrated in Equation (13) below:

$$\begin{aligned}
\text{Prediction} : \mathrm{E}[y] &= \gamma \\
\text{Data uncertainty} : \mathrm{E}[\tfrac{\sigma^2}{\nu}] &\approx \tfrac{\sigma^2\beta}{\alpha} \\
\text{Predictive uncertainty} : \mathrm{Var}[y] &= \tfrac{\sigma^2\beta}{\alpha} \cdot \tfrac{2\alpha}{2\alpha-2} = \tfrac{\sigma^2\beta}{\alpha-1} \\
\text{Model uncertainty} : \mathrm{Var}[y] - \mathrm{E}[\tfrac{\sigma^2}{\nu}] &= \tfrac{\sigma^2\beta}{\alpha-1} - \tfrac{\sigma^2\beta}{\alpha} = \tfrac{\sigma^2\beta}{\alpha(\alpha-1)}.
\end{aligned} \tag{13}$$

We argue that SMD (prior on variance through the scaling factor) offers an attractive trade-off between model complexity and granularity, occupying the middle ground between Ensemble (no prior) and Evidential (prior on both mean and variance). Recall that the result in Equation (11) can be interpreted as a Normal distribution being stretched out into a heavier tailed t-distribution when variance is unknown. Kurtosis of the t-distribution is controlled by the shape parameter (2α). In analysing Equation (11) and (13), we argue that α is analogous to "virtual observations" (ν) in NIG. Model uncertainty $\frac{\sigma^2\beta}{\alpha(\alpha-1)}$ is smaller than data uncertainty $\frac{\sigma^2\beta}{\alpha}$ by a factor of $\frac{1}{\alpha-1}$, when $\alpha > 2$. Thus, as α increases, both model uncertainty and scale of the marginal t-distribution monotonically decrease. Importantly, model uncertainty also drops relative to data uncertainty, as the t-distribution converges to the Normal distribution on increasing α. This stands in contrast to the model with NIG prior (Equation (7)), where increasing evidence ν does not monotonically lead to a decrease in scale of the t-distribution. We argue that this eases the challenge faced by the neural network in finding the optimal solution.

3.2 Architecture of the neural network

For the main application of this work, uncertainty quantification of financial time-series forecasts, we propose a novel architecture for the modelling of distribution parameters, as illustrated in Figure 1. To predict $\hat{y}_t$, time-series inputs of both *returns* $(r_{t-K+1}, \ldots, r_t)$ and *log-transformed squared returns* $(\log[r^2_{t-K+1}], \ldots, \log[r^2_t])$ are fed into one or more long short-term memory (LSTM) layers (Hochreiter & Schmidhuber, 1997). We log-transform squared returns to reduce skewness. The LSTM layers convert each time-series into a latent representation. The latent representation is then fed into four subnetworks, where each subnetwork is comprised of one or more fully connected layers and applies non-linear transformations on the latent representation. This allows the network to model complex relationships between the parameters in $\boldsymbol{\zeta}$ and the sequence. Return forecast $\hat{y}_t$ is given by γ, while $\{\sigma^2, \alpha, \beta\}$ are used to compute predictive uncertainty $\mathrm{Var}[\hat{y}]$. Note that in this setup, β is a redundant parameter as it exists as a product together with σ^2 in both the marginal NLL (Equation (12)) and in all three uncertainty measures (Equation (13)). Parameters σ^2 and β indicate scales of the Normal and Gamma distributions, respectively. Together, they contribute to the scale of the marginal t-distribution. Thus, one potential simplification of the network is to combine σ^2 and β into a single $\sigma^2\beta$ parameter and have the four subnetworks reduced to three. In other words, the network architecture illustrated in Figure 1 can be modified to output three parameters: $\boldsymbol{\zeta} = (\gamma, \sigma^2\beta, \alpha)$. We have kept β to be comparable to Evidential but provide empirical results in Appendix B using the UCI dataset (to be introduced in Section 4.1) to show that the two networks are indeed equivalent. In the following, we explore the proposed design of the architecture in detail.

Lakshminarayanan et al. (2017) and Amini et al. (2020) introduced the `Gaussian` and `NormalInverseGamma` layers as the final layer of a neural network. These final layers output parameters of the posterior distribution. Let $\boldsymbol{a} \in \mathbb{R}^{H^{(I)}}$ be the input vector of the final layer with $H^{(I)}$ dimensions and $H^{(O)}$ be the dimension of the output layer. In the case of the `NormalInverseGamma` layer, $H^{(O)} = 4$. The `NormalInverseGamma` layer outputs,

$$\begin{aligned}
\boldsymbol{\zeta} = \mathrm{O}(\boldsymbol{a}; \boldsymbol{\theta}) &= \boldsymbol{a}^{\mathsf{T}} \cdot \boldsymbol{W}^{(O)} + \boldsymbol{b}^{(O)} \\
\gamma = \zeta_1, \quad \nu = \zeta_2, \quad \alpha = \zeta_3, \quad \beta &= \zeta_4,
\end{aligned} \tag{14}$$

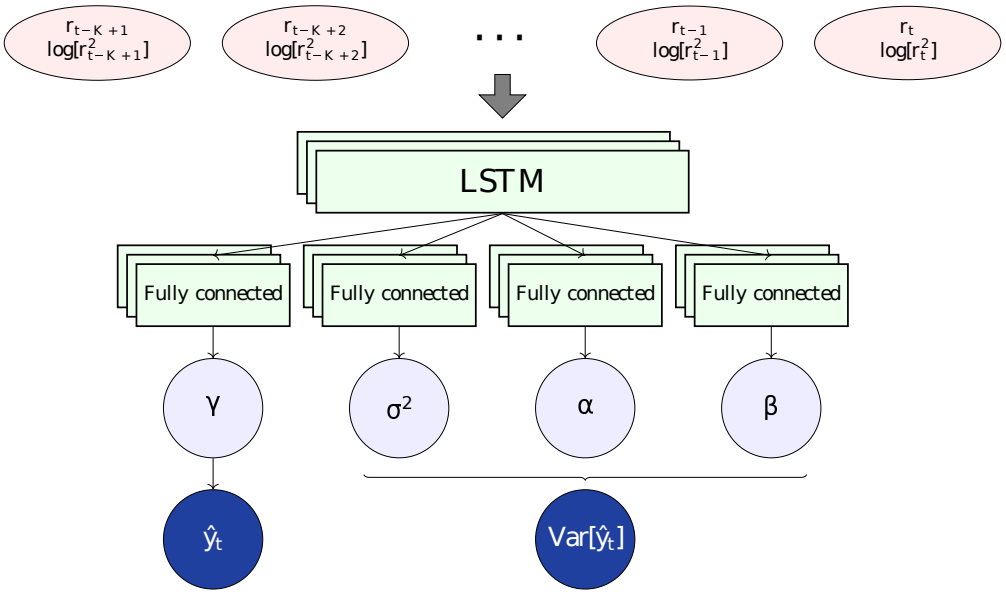


Figure 1: Input sequence (shaded in red) is passed into one or more LSTM layers. Output from the LSTM layers is then fed into four subnetworks of one or more fully connected layers with ReLU activation. The final layer of each subnetwork is a fully-connected layer with linear activation. Softplus is applied to σ^2, α and β to ensure positivity. The four outputs of the neural network are then used to compute $\hat{y}$ and $\text{Var}[\hat{y}]$ (shaded in blue).

where O denotes the `NormalInverseGamma` output layer, $\{\zeta_{1,\ldots,4}\}$ are 1^{st}, ..., 4^{th} elements of vector $\boldsymbol{\zeta}$, $\boldsymbol{W}^{(O)} \in \mathbb{R}^{H^{(I)} \times H^{(O)}}$ and $\boldsymbol{b}^{(O)} \in \mathbb{R}^{H^{(O)}}$ are weights and bias of the output layer, respectively. Each dimension of $\boldsymbol{\zeta}$ corresponds to each of γ, ν, α and β.

Outputs of the `NormalInverseGamma` layer are linear transformations of a common input $\boldsymbol{a}$ (Equation (14)). We argue that this construct is too restrictive for complex applications, such as in quantifying uncertainty of financial time-series forecasts, as detailed in Section 4.2. We propose to model each of the four parameters of SMD with its own subnetwork of one or more fully connected layers. This allows for a more expressive modelling of $\boldsymbol{\zeta}$, where each parameter may have complex, non-linear relationships with the input.

Additionally, we enforce constraints on $\sigma^2 > 0$, $\alpha > 1$ and $\beta > 0$ by applying softplus transformation with a constant term, $z' = \log(1 + \exp(z)) + c$, where $z \in \{\sigma^2, \alpha, \beta\}$ and c is the minimum value of the respective parameters. The transformed values constitute the final output of the network: $\boldsymbol{\zeta}' = \{\gamma, (\sigma^2)', \alpha', \beta'\}$. In Section 4.2, we show that this modification vastly improves quantification of forecast uncertainty of financial time-series. For other network architectures, we argue that the same approach can be applied. In the case of a feedforward network, we recommend having at least one common hidden layer that reduces the input to a single latent representation. The latent representation is then passed to individual stacks of hidden layer(s) for specialization. We argue that the common hidden layer allows information sharing across the four parameters, while having no common hidden layer (i.e., if the input is fed into the four disjoint stacks of hidden layers directly) will prevent sharing of information across the stacks.

Machine learning models are typically trained using pooled dataset of historical observations. As such, they learn the average uncertainty within the historical data. However, as noted in Section 1, asset returns exhibit time-varying volatility clustering patterns. Thus, we expect forecast uncertainty to be correlated with time-varying variance of the DGP. In other words, forecast uncertainty is high when σ_t^2 of the DGP is high and the model is "surprised" by the volatility. As we will show in Section 4.2, this feature of asset returns complicates uncertainty quantification of time-series forecasts. To inform the neural network of the prevailing volatility environment, we propose to include the log of squared returns $\{\log(r_{t-K+1}^2), \ldots, \log(r_t^2)\}$ as part of the input matrix. This follows from the use of squared returns in volatility forecasting literature (Brownlees et al.,

2011). We argue that this allows the neural network to infer the prevailing volatility environment. However, we note that there are two differences between uncertainty quantification and volatility forecasting. First, we are concerned with predictive uncertainty (the sum of data and model uncertainties), whereas volatility forecasting is concerned with data uncertainty only. Second, the most common definition of volatility in finance literature is realized volatility (e.g., sample variance of returns over the past 30 days; Ge et al., 2022), which is the mean squared deviation from the sample mean and is not linked to accuracy of the forecast. By contrast, the better the model is at predicting y_t, the lower the predictive uncertainty. In the extreme case where the model can perfectly predict future returns, predictive uncertainty is nil but returns will still exhibit volatility around its sample mean.

Model averaging, as a special case of ensembling, is a well studied statistical method for improving predictive power of estimators (Breiman, 1996; Goodfellow et al., 2016), and has previously been shown to improve accuracy of financial time-series forecasting (Wong et al., 2021) and sequential predictions (Raftery et al., 2010). As accuracy of both return forecast accuracy and predictive uncertainty are important in our motivating application, we propose to incorporate model averaging to simultaneously improve return forecasts and predictive uncertainties. For an ensemble of M models, we compute the ensemble forecast $\tilde{y}$ and predictive variance $\text{Var}[\tilde{y}]$ as,

$$\tilde{y} = \frac{1}{M}\sum_{i=1}^{M}\hat{y}_i, \quad \text{Var}[\tilde{y}] = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i^2 + \text{Var}[\hat{y}_i]) - \tilde{y}^2, \tag{15}$$

where $\hat{y}_i$ and $\text{Var}[\hat{y}_i]$ are mean and predictive variance of model i, respectively. In Section 4, we show that model averaging resulted in the highest predictive performance while maintaining uncertainty quantification performance.

Popular tools for modelling time-varying volatility are Autoregressive Conditional Heteroskedasticity (ARCH; Engle, 1982) and Generalized ARCH (GARCH; Bollerslev, 1986) models. GARCH, when applied to stock returns, assumes the same DGP as Equation (1). Time-varying variance σ_t^2 is modelled using an ARMA model (Box et al., 1994). μ_t can assume a fixed value (e.g., sample mean) or modelled using time-series models such as ARMA (leading to the ARMA-GARCH formulation). In our proposed framework, squared returns are provided as inputs to LSTM in similar spirit to the autoregressive terms of squared returns in GARCH. However, our proposed framework also has few differences to ARMA-GARCH. A neural network offers greater flexibility in modelling and can automatically discover interaction effects of first- and second-moment of returns. For example, higher volatility is negatively correlated with future asset returns (known as the *leverage effect*; Cont, 2001). By contrast, modelling of interaction effects in additive models (such as GARCH) requires explicit specification by the user. LSTM can also be interpreted as having dynamic autoregressive orders (as opposed to fixed orders in GARCH). The input and forget gates of LSTM allow the network to control the extent of long-memory depending on features of the time-series. Lastly, multi-step ahead forecasting is an iterative process for ARMA-GARCH and forecast errors may compound. LSTM is able to predict multi-step ahead directly. In Section 4.2, we apply our framework to forecast forward 1-month U.S. stock returns using daily returns. Nonetheless, we do not directly compare against ARMA-GARCH models for two reasons. First, in this work, we are mainly focused on advancing uncertainty quantification methodologies for neural networks. We argue that several of our advances can be beneficial to both time-series and non-time-series datasets (as demonstrated in Section 4.1). Second, we lean on the plethora of literature in comparing LSTM to ARMA-variants (e.g., Siami-Namini et al., 2018) and ARCH-variants (e.g., Liu, 2019).

For ease of comparison, we outline the differences of our method to Ensemble (Lakshminarayanan et al., 2017) and Evidential (Amini et al., 2020) in Table 1.

4 Experiments

SMD parameterization, distribution parameter modelling and ensemble predictions can be applied to general applications of prediction uncertainty quantification, while modelling volatility clustering relate specifically to time-series predictions with complex structures. In this section, we detail the experiment results in three real world datasets to illustrate the benefits of our improvements.

Table 1: A comparison of Combined to Deep Ensemble and Deep Evidential regressions. *Output layer* refers to the structure of output layer(s) of the network that outputs the parameters of the likelihood function.

Method	**Ensemble**	**Evidential**	**Combined**
Prior	None	NIG	Gamma
Ensemble	Yes	No	Yes
Likelihood	Gaussian	Student's t	Student's t
Output layer	Single layer μ, σ^2	Single layer $\gamma, \nu, \alpha, \beta$	Multi-layer $\gamma, \sigma^2, \alpha, \beta$

4.1 Benchmarking on UCI dataset

We first compare our method to Ensembles and Evidential on the UCI collection used in Hernández-Lobato & Adams (2015); Gal & Ghahramani (2016); Lakshminarayanan et al. (2017); Amini et al. (2020). The collection consists of 9 real world regression problems, each with 10–20 features and hundreds to tens of thousands of observations. We follow Lakshminarayanan et al. (2017) and Amini et al. (2020) in evaluating our method on root mean squared error (RMSE, which assesses forecast accuracy) and NLL (which assesses overall distributional fit), and compare against Ensemble and Evidential. While we do not explicitly compare inference speed, as our Combined method also uses ensembling, inference speed is expected to be comparable to Ensemble while being slower than Evidential. We use the source code provided by Amini et al. (2020), with the default topology of a single hidden layer with 50 units for both Ensemble and Evidential[8]. For Combined, as individual modelling of distribution parameters (Section 3.2) requires a network with two or more hidden layers, we have used a single hidden layer with 24 units, followed by 4 separate stacks of a single hidden layer with 6 units each. Thus, the total number of non-linear units is 48 (compared to 50 for Ensemble and Evidential). Note that even though the total number of units are similar across the three models, learning capacity may differ due to different topologies.

Table 2 records experiment results on the UCI dataset. On RMSE, we find that both Ensemble and Combined have performed well, having the best RMSE in four datasets each. In two of the sets (*Kin8nm* and *Naval*), all three methods produced highly accurate results that are not separable to two decimal points. Turning to NLL, we observe a trend towards Combined having lower NLL than the other two methods for four sets, followed by Ensemble with three sets. Comparing Combined to Evidential, we find that Combined generally has lower RMSE (7 of 9 sets) and NLL (6 of 9 sets). Although our method is designed for uncertainty quantification of complex time-series and all 9 datasets are pooled (non-time-series) datasets, we still observe some improvements in both RMSE and NLL.

Next, we present further ablation studies on the UCI dataset. Table 3 records results of *Alternative*, which utilizes ensembling and SMD parameterization but not separate modelling of hyperparameters. Alternative has the same network topology as Ensemble and Evidential (a single hidden layer with 50 units), as opposed to Combined which has two hidden layers with a total of 48 units. We observe that Ensemble has the lowest RMSE in 5 (of 9) datasets, followed by Alternative (3 of 9), while Alternative has the best NLL in 6 (of 9) datasets and Ensemble has 3 (of 9). On both metrics, Evidential has the least favourable performance. Comparing Combined in Table 2 and Alternative in Table 3, Combined has lower RMSE and NLL in 5 of 9 datasets. Thus, we conclude that separate modelling of hyperparameters provided an incremental benefit on the UCI datasets.

In Table 4, we further remove model averaging. The network used is identical to Evidential but trained using the SMD parameterization (i.e., we simply change the loss function in Evidential to Equation (12)). We observe that the network trained using the SMD parameterization has lower RMSE in 6 of 9 and lower NLL in 8 out 9 datasets. We argue that the improved performance of the SMD parameterization is due to its simplicity.

[8]Source code for Amini et al. (2020) is available on Github: `https://github.com/aamini/evidential-deep-learning`

Table 2: Comparing Ensemble (Lakshminarayanan et al., 2017), Evidential (Amini et al., 2020) and Combined (this work) on RMSE and NLL using the UCI benchmark datasets. Average result and standard deviation over 5 trials for each method. The best method for each dataset and metric are highlighted in **bold**.

	RMSE			NLL		
Dataset	**Ensemble**	**Evidential**	**Combined**	**Ensemble**	**Evidential**	**Combined**
Boston	$\mathbf{2.66 \pm 0.20}$	2.95 ± 0.29	2.89 ± 0.31	2.28 ± 0.05	2.30 ± 0.05	$\mathbf{2.23 \pm 0.05}$
Concrete	5.79 ± 0.16	5.98 ± 0.23	$\mathbf{5.40 \pm 0.18}$	3.07 ± 0.02	3.11 ± 0.04	$\mathbf{2.98 \pm 0.03}$
Energy	1.86 ± 0.04	1.84 ± 0.06	$\mathbf{1.71 \pm 0.20}$	1.36 ± 0.02	1.41 ± 0.04	$\mathbf{1.35 \pm 0.05}$
Kin8nm	$\mathbf{0.06 \pm 0.00}$	0.06 ± 0.00	0.06 ± 0.00	$\mathbf{-1.39 \pm 0.02}$	-1.28 ± 0.03	-1.35 ± 0.02
Naval	$\mathbf{0.00} \pm 0.00$	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{-6.10 \pm 0.05}$	-5.99 ± 0.09	-5.89 ± 0.35
Power	3.02 ± 0.09	3.02 ± 0.08	$\mathbf{2.95 \pm 0.08}$	2.57 ± 0.01	2.56 ± 0.03	$\mathbf{2.53 \pm 0.02}$
Protein	3.71 ± 0.10	4.28 ± 0.23	$\mathbf{3.67 \pm 0.13}$	$\mathbf{2.61 \pm 0.03}$	2.73 ± 0.08	2.70 ± 0.05
Wine	0.60 ± 0.03	$\mathbf{0.56 \pm 0.02}$	0.59 ± 0.03	0.94 ± 0.04	$\mathbf{0.92 \pm 0.04}$	1.00 ± 0.03
Yacht	$\mathbf{1.22 \pm 0.22}$	1.48 ± 0.47	3.97 ± 1.06	1.06 ± 0.08	$\mathbf{0.96 \pm 0.19}$	1.17 ± 0.11

Table 3: Comparing Ensemble, Evidential and Alternative (without separate modelling of the four parameters of SMD) on RMSE and NLL using the UCI benchmark datasets. Average result and standard deviation over 5 trials for each method. The best method for each dataset and metric is highlighted in **bold**.

	RMSE			NLL		
Dataset	**Ensemble**	**Evidential**	**Alternative**	**Ensemble**	**Evidential**	**Alternative**
Boston	$\mathbf{2.66 \pm 0.20}$	2.95 ± 0.29	2.87 ± 0.18	$\mathbf{2.28 \pm 0.05}$	2.30 ± 0.05	2.29 ± 0.04
Concrete	5.79 ± 0.16	5.98 ± 0.23	$\mathbf{5.72 \pm 0.15}$	3.07 ± 0.02	3.11 ± 0.04	$\mathbf{3.03 \pm 0.02}$
Energy	1.86 ± 0.04	$\mathbf{1.84 \pm 0.06}$	1.88 ± 0.04	1.36 ± 0.02	1.41 ± 0.04	$\mathbf{1.35 \pm 0.03}$
Kin8nm	$\mathbf{0.06 \pm 0.00}$	0.06 ± 0.00	0.06 ± 0.00	$\mathbf{-1.39 \pm 0.02}$	-1.28 ± 0.03	-1.38 ± 0.02
Naval	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	0.00 ± 0.00	-6.10 ± 0.05	-5.99 ± 0.09	$\mathbf{-6.12 \pm 0.06}$
Power	3.02 ± 0.09	3.02 ± 0.08	$\mathbf{2.97 \pm 0.10}$	2.57 ± 0.01	2.56 ± 0.03	$\mathbf{2.54 \pm 0.02}$
Protein	$\mathbf{3.71 \pm 0.10}$	4.28 ± 0.23	3.75 ± 0.11	$\mathbf{2.61 \pm 0.03}$	2.73 ± 0.08	2.72 ± 0.02
Wine	0.60 ± 0.03	0.56 ± 0.02	$\mathbf{0.55 \pm 0.02}$	0.94 ± 0.04	0.92 ± 0.04	$\mathbf{0.92 \pm 0.02}$
Yacht	$\mathbf{1.22 \pm 0.22}$	1.48 ± 0.47	1.45 ± 0.33	1.06 ± 0.08	0.96 ± 0.19	$\mathbf{0.93 \pm 0.09}$

4.2 Uncertainty quantification in financial time-series forecasting

Next, we demonstrate the benefits of our method in the main applications of this work, uncertainty quantification in financial time-series forecasting. In this section, we will first describe the cryptocurrency dataset, then the U.S. equities dataset. The same neural network architectures are used in the two datasets, with hyperparameter tuned independently.

Our cryptocurrency dataset consists of hourly returns downloaded from Binance over July 2018 to December 2021, for 10 of the most liquid, non-*stablecoin*[9] cryptocurrencies. Tickers for these cryptocurrencies are BTC, ETH, BNB, NEO, LTC, ADA, XRP, EOS, TRX and ETC, denominated in USDT[10]. Following (Ambachtsheer, 1974; Grinold & Kahn, 1999; Wong et al., 2021), we use mean cross-sectional correlation ($\frac{1}{T}\sum_{t=1}^{T}\rho(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$) as a measure of predictive accuracy, in addition to RMSE and NLL. Data from July 2018 to June 2019 are used for hyperparameter tuning, chronologically split into 70 % training and 30 % validation. Data from July 2019 to December 2021 are used for out-of-sample testing. Networks are trained every 30 days using an expanding window of data from July 2018. Each input sequence consists of 10 days of hourly returns r and squared returns $\log(r^2)$ (i.e., the input is a matrix with dimensions 240×2), and are

[9]Stablecoins are cryptocurrencies that are pegged to real world assets (e.g., U.S. Dollar). As such, they exhibit lower volatility than other non-pegged cryptocurrencies.

[10] *Tether* (USDT) is a stablecoin that is pegged to USD. It has the highest market capitalization amongst the USD-linked stablecoins Lipton (2021).

Table 4: Comparing Normal-Inverse-Gamma and Normal-Gamma on RMSE and NLL using the UCI benchmark datasets. Average result and standard deviation over 5 trials for each method. The best method for each dataset and loss function is highlighted in **bold**.

	RMSE		NLL	
Dataset	**NIG**	**SMD**	**NIG**	**SMD**
Boston	$\mathbf{2.95 \pm 0.29}$	2.97 ± 0.20	$\mathbf{2.30 \pm 0.05}$	2.31 ± 0.05
Concrete	5.98 ± 0.23	$\mathbf{5.78 \pm 0.23}$	3.11 ± 0.04	$\mathbf{3.05 \pm 0.04}$
Energy	$\mathbf{1.84 \pm 0.06}$	1.87 ± 0.16	1.41 ± 0.04	$\mathbf{1.33 \pm 0.05}$
Kin8nm	0.06 ± 0.00	$\mathbf{0.06 \pm 0.00}$	-1.28 ± 0.03	$\mathbf{-1.37 \pm 0.01}$
Naval	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	-5.99 ± 0.09	$\mathbf{-6.27 \pm 0.09}$
Power	3.02 ± 0.08	$\mathbf{2.98 \pm 0.12}$	2.56 ± 0.03	$\mathbf{2.53 \pm 0.02}$
Protein	4.28 ± 0.23	$\mathbf{3.72 \pm 0.16}$	2.73 ± 0.08	$\mathbf{2.39 \pm 0.05}$
Wine	$\mathbf{0.56 \pm 0.02}$	0.56 ± 0.03	0.92 ± 0.04	$\mathbf{0.87 \pm 0.04}$
Yacht	1.48 ± 0.47	$\mathbf{1.44 \pm 0.49}$	0.96 ± 0.19	$\mathbf{0.91 \pm 0.18}$

used to predict forward one hour return. This training scheme is shared across all three models. Network topology consists of LSTM layers, followed by fully connected layers with $ReLU$ activation and the corresponding output layers of Ensemble and Evidential. For Combined, we use four separate subnetworks as illustrated in Figure 1. As discussed in Section 1, we consider uncertainty quantification in cryptocurrencies to be especially challenging due to their high volatility. Note that in this section, "forecast uncertainty" and "uncertainty forecast" refer to predictive uncertainty (i.e., sum of model and data uncertainties).

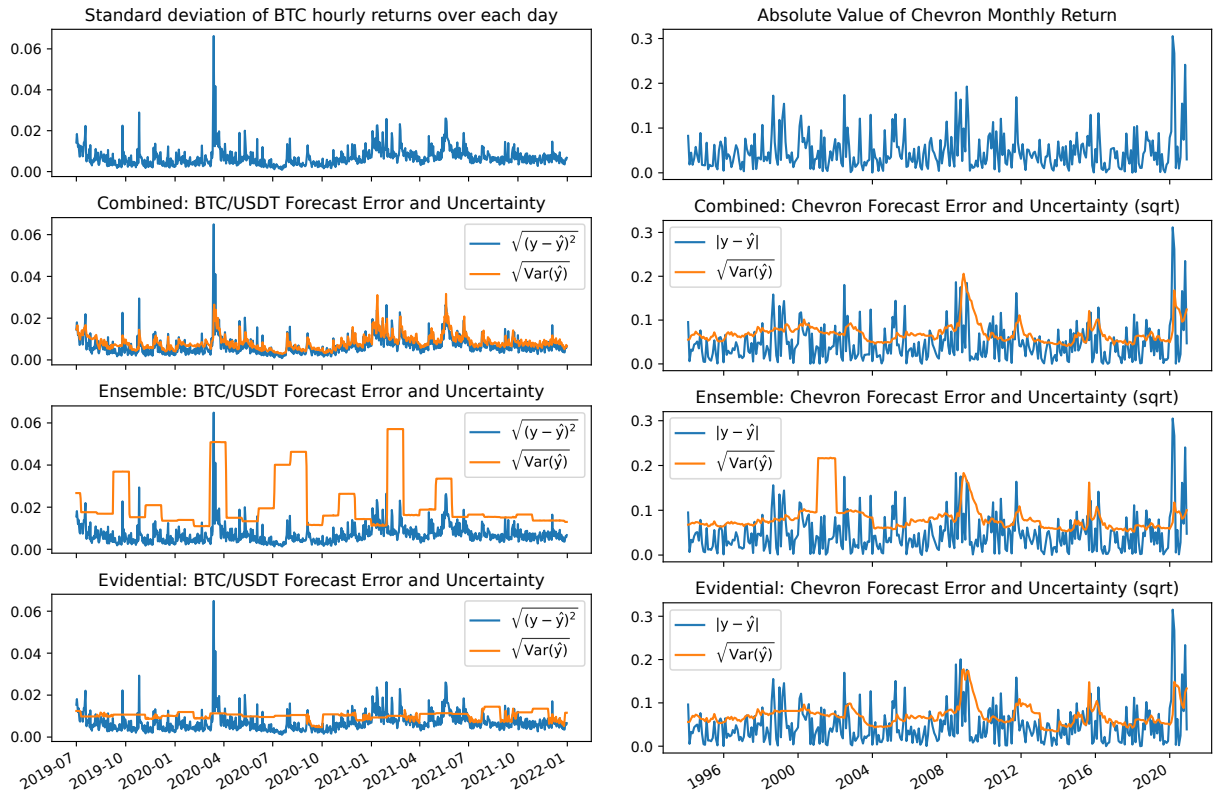
Our U.S. equities experiment follows the same setup as cryptocurrencies. Mimicking the S&P 500 index universe, the dataset consists of daily returns downloaded from the Wharton Research Data Service over 1984 to 2020, for the 500 largest stocks[11] listed on NASDAQ, NYSE and NYSE American. Data from 1984 to 1993 are used for hyperparameter tuning, while 1994 to 2020 are used for out-of-sample testing. The network is refitted every January using a rolling 10-year window. Each input sequence consists of 240 trading days of daily returns r and squared returns $\log(r^2)$, forecasting forward 20-day return and its uncertainty. At this point, it is useful to remind readers that prior literature have found both datasets to exhibit time-varying variance (e.g., Cont, 2001; Hafner, 2018), which is also visible in Figure 2.

We start with the main empirical results of this work. Table 5 contains forecast results on both the cryptocurrency dataset (left) and U.S. equities (right). We observe that Combined has the highest average cross-sectional correlation (higher is better), and lowest RMSE and NLL (both lower is better) in both datasets. This indicates that Combined has higher cross-sectional predictive efficacy (as measured by correlation) and is able to better forecast uncertainty of the time-series prediction. Evidential has higher (better) correlation and lower RMSE (better) than Ensemble but higher (worse) NLL in cryptocurrency. In U.S. equities, Evidential has worse correlation, RMSE and NLL than Ensemble. Correlations in U.S. equities are materially lower for all three methods compared to the cryptocurrency dataset. We hypothesize that this is due to both the difference in forecast horizon and maturity of the U.S. stock market.

Table 5: **Main results**: Comparing Ensemble, Evidential and Combined on average cross-sectional correlation, RMSE and NLL for cryptocurrencies (left) and U.S. equities (right), respectively. Average result and standard deviation over 10 trials for each method. Best method for each dataset is highlighted in **bold**.

	Cryptocurrency			U.S. equities		
Metric	**Ensemble**	**Evidential**	**Combined**	**Ensemble**	**Evidential**	**Combined**
Correlation (×100)	2.78 ± 1.09	3.94 ± 1.84	$\mathbf{9.87 \pm 3.17}$	0.40 ± 0.66	0.09 ± 0.93	$\mathbf{1.22 \pm 0.65}$
RMSE (×100)	0.874 ± 0.022	0.874 ± 0.003	$\mathbf{0.867 \pm 0.001}$	9.426 ± 0.044	9.433 ± 0.033	$\mathbf{9.379 \pm 0.020}$
NLL	-3.74 ± 0.10	-3.24 ± 0.02	$\mathbf{-4.14 \pm 0.01}$	-1.65 ± 0.17	-0.82 ± 0.03	$\mathbf{-1.71 \pm 0.01}$

[11] The list of stocks is refreshed every June, keeping the same stocks until the next rebalance.



(a) Uncertainty of Bitcoin

(b) Uncertainty of Chevron

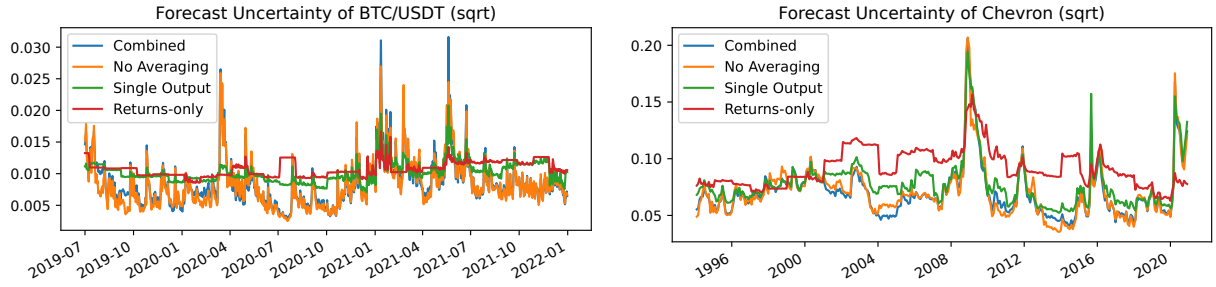
Figure 2: Forecast uncertainties of Combined, Ensemble and Evidential applied to Bitcoin (left) and Chevron Corp (right), respectively. For Bitcoin, square root of the average over each day shown.

Next, we compare predicted uncertainty and actual prediction error of the three methods to actual volatility of Bitcoin (BTC/USDT), the cryptocurrency with the highest market capitalization, and Chevron Corp., a major U.S. oil producer which have endured multiple market shocks, as illustrated in Figure 2. The true volatility of an asset (σ^2 in Equation (1)) is unobservable (Ge et al., 2022). Thus, in the top row of Figure 2, we use the standard deviation of hourly returns computed over each day for Bitcoin and absolute value of monthly returns for Chevron (as forecast horizon is monthly), as proxies for σ^2. In rows 2–4, for Bitcoin, we convert hourly forecasts to daily data points by first, computing the average squared forecast error (denoted $\sqrt{(y-\hat{y})^2}$), and the average predictive uncertainty (denoted $\sqrt{\mathrm{Var}(\hat{y})}$), over each day. We then plot the square-root of both quantities in the left column of rows 2–4 for the three methods in Figure 2. For Chevron, we plot the absolute error between actual monthly returns and predicted returns (denoted $|y-\hat{y}|$), and square-root of forecast uncertainty (denoted $\sqrt{\mathrm{Var}(\hat{y})}$) on the right column of Figure 2. Comparing the bottom three rows of Figure 2, which correspond to Combined, Ensemble and Evidential, respectively. We observe that Combined's predicted uncertainty of $\hat{\mu}$ tracks actual forecast error closely. This appears to be especially true during periods of elevated volatility, which are important to investors. However, during periods of low volatility, Combined appears to slightly overestimate expected uncertainty. This is also true for Ensemble in Bitcoin, where predicted uncertainty tends to be significantly higher than actual forecast error. For Chevron, we observe multiple spikes of volatility, corresponding to the U.S. recession over 2008-09, the oil shock in 2015 and the 2020 pandemic. Notably, we observe that Combined tracks the spike in forecast error better than Ensemble and Evidential. Note that the "block-like" appearances of the uncertainty forecasts are due to periodic training (monthly for cryptocurrencies and yearly for U.S. equities) and the failure to generalize the prevailing volatility environment. During training, the optimizer can update network weights $\boldsymbol{W}$ or bias $\boldsymbol{b}$ (which is analogous to the intercept in linear models). When the network fails to generalize, it

minimizes the loss function by updating the bias rather than the weights. Thus, outputting the same constant that do not vary with the input, until the network is re-trained in the following month. This produces the block-like appearances of Ensemble and Evidential, and is indicative of the network setup (e.g., no separate modelling of hyperparameters) being unsuitable to this class of problems. Lastly, Evidential underestimates forecast error during heightened volatility (e.g., March 2020 in Figure 2) and overestimates forecast error under periods of low volatility (e.g., July 2020 in Figure 2). We observe similar visual characteristics in the predicted uncertainty of other cryptocurrencies and stocks.

Table 6: **Ablation studies of Combined**: In each column, we remove model averaging (*No Averaging*), separate modelling of distribution parameters (*Single Output*) and using return time-series only (*Returns-only*) for cryptocurrencies (left) and U.S. equities (right), respectively. Average result and standard deviation over 10 trials for each method. Note that cryptocurrency returns are hourly and U.S. stock returns are monthly.

	Cryptocurrency			U.S. equities		
Metric	**No Averaging**	**Single Output**	**Returns-only**	**No Averaging**	**Single Output**	**Returns-only**
Correlation (×100)	4.48 ± 2.80	8.23 ± 2.91	10.46 ± 2.04	0.92 ± 0.65	1.87 ± 1.06	1.21 ± 0.73
RMSE (×100)	0.868 ± 0.001	0.872 ± 0.002	0.866 ± 0.002	9.392 ± 0.020	9.398 ± 0.029	9.384 ± 0.046
NLL	-3.35 ± 0.01	-4.04 ± 0.02	-3.95 ± 0.02	-0.88 ± 0.01	-1.63 ± 0.04	-1.34 ± 0.04



(a) Uncertainty of Bitcoin

(b) Uncertainty of Chevron

Figure 3: Predicted uncertainty $\text{Var}(\hat{y})$ of Combined, without model averaging (*Averaging*), single output layer (*Single Output*) and using returns only (*Returns-only*) for BTC/USDT and Chevron.

Next, we test the effects of removing each of the following for Combined: 1) model averaging; 2) single output layer (similar to Evidential); 3) using return time-series only (i.e., no squared returns). The results are recorded in Table 6 and in Figure 3. We observe that model averaging has a large impact on cross-sectional correlation and NLL. Cross-sectional correlation is 55 % and 25 % lower for cryptocurrencies and U.S. equities, respectively. While NLL is higher by 0.8 in both cases (lower is better), indicating a worse overall fit. However, it does not appear to impede the network's ability to model time-series forecast uncertainty. Using a single output layer leads to marginally worse NLL. Correlation is lower in cryptocurrencies but marginally higher in U.S. equities. While using returns only leads to marginally higher correlation but marginally worse on NLL in cryptocurrency, and lower correlation and worse NLL in U.S. equities. From Figure 3, the block-like appearances indicate that both using single output layer and using returns only result in the network failing to closely track time-varying variance of the DGP. This suggests that both squared returns and separate modelling of distribution parameters are required to model time-varying forecast uncertainty.

5 Conclusions

In this work, we present a method for the modelling of forecast uncertainty in presence of complex time-series structures, such as volatility clustering. Our proposed method extends the work of Lakshminarayanan et al. (2017) and Amini et al. (2020). We propose to use a SMD (which uses a Gamma prior for scale uncertainty ν) as a simpler alternative to a NIG prior (which places a Normal prior on μ and Inverse-Gamma prior on

σ^2). We show clearly in the UCI benchmark dataset (Table 4) that simply swapping the NIG prior with the SMD resulted in improved forecast uncertainty quantification performance. Parameters of SMD are modelled using separate subnetworks. Together with ensembling and the use of second order of returns as inputs, we show that our proposed method can successfully model time-varying variance of the DGP. This is illustrated through the successful quantification of forecast uncertainty of two financial time-series datasets: cryptocurrency and U.S. equities.

We observe that our method tends to marginally overestimate uncertainty during periods of low volatility. This opens an avenue for future research directions. We show that our proposed method can also be used to improve uncertainty quantification in non-time-series regression problems on UCI datasets. From a finance application perspective, forecast uncertainty can be used to size bets, or as advanced warning to protect the portfolio from downside risk. For example, if forecast uncertainty reaches a certain threshold, an investor could purchase portfolio insurance or liquidate positions to reduce risk. Lastly, uncertainty quantification in time-series applications is a relatively under-explored area of literature. We believe this work can lead to further advancements of uncertainty quantification in complex time-series.

References

Keith Ambachtsheer. Profit potential in an almost efficient market. *Journal of Portfolio Management*, 1(1): 84, FALL 1974.

Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In *Advances in Neural Information Processing Systems 33*, NIPS'20, pp. 14927–14937, Vancouver, BC, Canada, 2020. Curran Associates, Inc.

David F. Andrews and Colin L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(1):99–102, 1974.

José M. Bernardo and Adrian F. M. Smith. *Bayesian theory*. John Wiley & Sons Ltd., 2000.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Fischer Black and Robert B Litterman. Asset allocation: Combining investor views with market equilibrium. *Journal of Fixed Income*, 1(2):7–18, 1991.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. In *arXiv*, 2016. URL `https://arxiv.org/abs/1604.07316`.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3): 307–327, 1986. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(86)90063-1.

George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, N.J., USA, 3 edition, 1994.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 1573-0565.

Christian Brownlees, Robert Engle, and Bryan Kelly. A practical guide to volatility forecasting through calm and storm. *Journal of Risk*, 14(2):3–22, 2011.

Tim Byrnes and Tristan Barnett. Generalized framework for applying the kelly criterion to stock markets. *International Journal of Theoretical and Applied Finance*, 21(05):1–13, 2018. doi: 10.1142/S0219024918500334.

Cathy Yi-Hsuan Chen and Christian M. Hafner. Sentiment-induced bubbles in the cryptocurrency market. *Journal of Risk and Financial Management*, 12(2):1–12, 2019. ISSN 1911-8074.

S.T. Boris Choy and Jennifer S.K. Chan. Scale mixtures distributions in statistical modelling. *Australian & New Zealand Journal of Statistics*, 50(2):135–146, 2008. ISSN 1369-1473.

Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1:223–236, 2001.

Robert F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007, 1982.

John Fry and Eng-Tuck Cheah. Negative bubbles and shocks in cryptocurrency markets. *International Review of Financial Analysis*, 47:343–352, 2016. ISSN 1057-5219.

Yarin Gal. *Uncertainty in Deep Learning*. University of Cambridge, 2016. PhD thesis.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *ICML'16*, pp. 1050–1059. JMLR.org, 2016.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseo Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, M. Shahzad, Wen Yang, Richard Bamler, and Xiaoxiang Zhu. A survey of uncertainty in deep neural networks. In *arXiv*, 2021. URL `https://arxiv.org/abs/2107.03342`.

Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen. Neural network–based financial volatility forecasting: A systematic review. *ACM Computing Surveys*, 55(1), jan 2022. ISSN 0360-0300. doi: 10.1145/3483596. URL `https://doi.org/10.1145/3483596`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Richard Grinold and Ronald Kahn. *Active Portfolio Management: A Quantitative Approach for Producing Superior Returns and Controlling Risk*. McGraw-Hill Education, 1999.

Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.). *Advances in Neural Information Processing Systems 30*, NIPS'17, Long Beach, California, USA, 2017. Curran Associates, Inc.

Christian M. Hafner. Testing for Bubbles in Cryptocurrencies with Time-Varying Volatility. *Journal of Financial Econometrics*, 18(2):233–249, 10 2018. ISSN 1479-8409. doi: 10.1093/jjfinec/nby023. URL `https://doi.org/10.1093/jjfinec/nby023`.

Wilfred Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *ICML'15*, pp. 1861–1869. JMLR.org, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. In *arXiv*, 2019. URL `https://arxiv.org/abs/1910.09457`.

Michael Jordan. The exponential family: Conjugate priors, 2009.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks – a tutorial for deep learning users. In *arXiv*, 2022. URL `https://arxiv.org/abs/2007.06823`.

John L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4):917–926, 1956.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In Guyon et al. (2017), pp. 5580–5590. ISBN 9781510860964.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *arXiv*, 2020. URL `https://arxiv.org/abs/2002.10118`.

Balaji Lakshminarayanan, AlexanderPritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon et al. (2017), pp. 6405–6416. ISBN 9781510860964.

Alexander Lipton. Cryptocurrencies change everything. *Quantitative Finance*, 21(8):1257–1262, 2021. doi: 10.1080/14697688.2021.1944490.

Yang Liu. Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. *Expert systems with applications*, 132:99–109, 2019. ISSN 0957-4174.

David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448.

John Mitros and Brian Mac Namee. On the validity of bayesian neural networks for uncertainty estimation. In *arXiv*, 2019. URL `https://arxiv.org/abs/1912.01530`.

Radford M. Neal. *Bayesian learning for neural networks*. University of Toronto, 1995. PhD thesis.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

José Antonio Núñez, Mario I. Contreras-Valdez, and Carlos A. Franco-Ruiz. Statistical analysis of bitcoin during explosive behavior periods. *PLOS ONE*, 14(3):1–22, 03 2019. doi: 10.1371/journal.pone.0213919. URL `https://doi.org/10.1371/journal.pone.0213919`.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32*, NIPS'19, Vancouver, BC, Canada, 2019. Curran Associates, Inc.

M. Hashem Pesaran and Allan Timmermann. Predictability of stock returns: Robustness and economic significance. *Journal of Finance*, 50:1201–1228, 1995.

Alla Petukhina, Simon Trimborn, Wolfgang Karl Härdle, and Hermann Elendner. Investing with cryptocurrencies – evaluating their potential for portfolio allocation strategies. *Quantitative Finance*, 21(11): 1825–1853, 2021.

Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up mcmc by efficient data subsampling. *Journal of the American Statistical Association*, 114(526):831–843, 2019. doi: 10.1080/01621459.2018.1448827.

Adrian E. Raftery, Miroslav Kárný, and Pavel Ettler. Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill. *Technometrics*, 52(1):52–66, 2010. doi: 10.1198/TECH.2009.08104. PMID: 20607102.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In M. Arif Wani (ed.), *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications*, ICMLA, pp. 1394–1401, Orlando, FL, USA, 2018. IEEE.

Steven Y. K. Wong, Jennifer Chan, Lamiae Azizi, and Richard Y. D. Xu. Supervised temporal autoencoder for stock return time-series forecasting. In Wing Kwong Chan, Bill Claycomb, and Hiroki Takakura (eds.), *Proceedings of the IEEE 45th Annual Computer Software and Applications Conference*, COMPSAC 2021, Madrid, Spain, 2021. IEEE.

A Marginal distribution of a Scale Mixture

From Equation (10), we have $\mathrm{N}(y|\gamma, \frac{\sigma^2}{\lambda})\,\mathrm{Gam}(\lambda|\alpha,\beta)$. Marginalizing over λ produces the data likelihood,

$$\begin{aligned}
\mathrm{p}(y|\gamma,\sigma^2,\alpha,\beta) &= \int_\lambda^\infty \mathrm{p_N}(y|\gamma,\sigma^2\lambda^{-1})\mathrm{p_G}(\lambda|\alpha,\beta)\,\mathrm{d}\lambda \\
&= \int_\lambda^\infty \left[\sqrt{\frac{\lambda}{2\pi\sigma^2}}\exp\left\{-\frac{\lambda(y-\gamma)^2}{2\sigma^2}\right\}\right]\left[\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}\exp^{-\beta\lambda}\right]\mathrm{d}\lambda \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}}\int_{\lambda=0}^\infty \lambda^{\alpha-\frac{1}{2}}\exp\left\{-\frac{\lambda(y-\gamma)^2}{2\sigma^2}-\beta\lambda\right\}\mathrm{d}\lambda \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}}\left[\frac{(y-\gamma)^2}{2\sigma^2}+\beta\right]^{-(\alpha+\frac{1}{2})}\int_{\lambda=0}^\infty\left\{\lambda\left[\frac{(y-\gamma)^2}{2\sigma^2}+\beta\right]\right\}^{\alpha-\frac{1}{2}} \\
&\quad \exp\left\{-\lambda\left[\frac{(y-\gamma)^2}{2\sigma^2}+\beta\right]\right\}\mathrm{d}\left\{\lambda\left[\frac{(y-\gamma)^2}{2\sigma^2}+\beta\right]\right\},
\end{aligned}$$

since $\int_0^\infty x^{\alpha-1}\exp(-x)\,\mathrm{d}x = \Gamma(\alpha)$,

$$= \frac{\beta^\alpha}{\sqrt{2\pi\sigma^2}}\frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\left[\frac{(y-\gamma)^2}{2\sigma^2}+\beta\right]^{-(\alpha+\frac{1}{2})}$$

and re-arranging $\beta^\alpha = (\frac{1}{\beta})^{-\alpha} = (\frac{1}{\beta})^{-(\alpha+\frac{1}{2})+\frac{1}{2}}$,

$$\begin{aligned}
&= \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\frac{1}{\sqrt{2\pi\sigma^2\beta}}\left[\frac{(y-\gamma)^2}{2\sigma^2\beta}+1\right]^{-(\alpha+\frac{1}{2})} \\
\mathrm{p}(y|\gamma,\sigma^2,\alpha,\beta) &= \mathrm{St}\left(y;\gamma,\frac{\sigma^2\beta}{\alpha},2\alpha\right). \qquad (16)
\end{aligned}$$

To show that the last step of Equation (16) is true, we start with the probability density function of the t-distribution parameterised in terms of precision $\mathrm{St}(y|\gamma,b^{-1},a)$ (Bishop, 2006),

$$\mathrm{St}(y|\gamma,b^{-1},a) = \frac{\Gamma(\frac{a+1}{2})}{\Gamma(\frac{a}{2})}\left[\frac{b}{\pi a}\right]^{\frac{1}{2}}\left[1+\frac{b(y-\gamma)^2}{a}\right]^{-(\frac{a+1}{2})},$$

where γ is location, b is inverse of scale and a is shape[12]. Substituting in $b^{-1} = \frac{\sigma^2\beta}{\alpha}$ and $a = 2\alpha$,

$$\begin{aligned}
\mathrm{St}\left(y|\gamma,\frac{\sigma^2\beta}{\alpha},2\alpha\right) &= \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\left[\frac{(\frac{\alpha}{\sigma^2\beta})}{2\pi\alpha}\right]^{\frac{1}{2}}\left[1+\frac{\alpha(y-\gamma)^2}{2\sigma^2\alpha\beta}\right]^{-(\alpha+\frac{1}{2})} \\
&= \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\frac{1}{\sqrt{2\pi\sigma^2\beta}}\left[\frac{(y-\gamma)^2}{2\sigma^2\beta}+1\right]^{-(\alpha+\frac{1}{2})}.
\end{aligned}$$

From Equation (16), the NLL of the marginal t-distribution is,

$$\begin{aligned}
\mathrm{p}(y|\gamma,\sigma^2,\alpha,\beta) &= \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha)}\frac{1}{\sqrt{2\pi\sigma^2\beta}}\left[\frac{(y-\gamma)^2}{2\sigma^2\beta}+1\right]^{-(\alpha+\frac{1}{2})} \\
-\log[\mathrm{p}(y|\gamma,\sigma^2,\alpha,\beta)] &= \log\left[\frac{\Gamma(\alpha)}{\Gamma(\alpha+\frac{1}{2})}\right]+\tfrac{1}{2}\log[2\pi\sigma^2\beta]+(\alpha+\tfrac{1}{2})\log\left[\frac{(y-\gamma)^2}{2\sigma^2\beta}+1\right].
\end{aligned}$$

[12]Note that the definition of scale b and shape a is used exclusively in this section. Not to be confused with network bias b and activation vector $\boldsymbol{a}$ used in the rest of this thesis.

B Further analysis of parameters in a Scale Mixture

In the network architecture proposed in Section 3, output of the network is $\boldsymbol{\zeta} = (\gamma, \sigma^2, \alpha, \beta)$, which parameterises the SMD (Equation (10)). However, as noted in Section 3, β exists as a product with σ^2 in both the marginal t-distributed NLL and in the three uncertainty measures (Equation (13)). Thus, an alternative specification of the network is to output $\boldsymbol{\zeta} = (\gamma, \sigma^2\beta, \alpha)$ (i.e., three parameters instead of four and are computed through three subnetworks, instead of four in Figure 1). We label this network *S2B*. In Table 7, we compare Combined (4 parameters) with S2B (3 parameters) using the UCI dataset (as introduced in Section 4.1). We observe that S2B is better than Combined on 4 (of 9) datasets on RMSE, while Combined is better than S2B on 1 (of 9). Four datasets (Boston, Kin8nm, Naval and Power) are inseparable at 2 decimal places on RMSE. On NLL, S2B is better than Combined on 7 (of 9) datasets, while Combined is better than S2B on 2 (of 9). Even though S2B has a higher number of datasets with lower RMSE and NLL, we note that the differences are very small and are within margin of error (due to randomness in neural network training). Thus, we conclude that the two methods provide near identical results but note that S2B is simpler and more interpretable. However, we choose Combined with four subnetworks to conduct our analysis so that parameters can also be compared with those from Evidential.

Table 7: Comparing S2B (3 parameters) to Combined (4 parameters) on RMSE and NLL using the UCI benchmark datasets. Results are averaged over 5 trials and the best method for each dataset and metric are highlighted in **bold**.

	RMSE		**NLL**	
Dataset	**S2B**	**Combined**	**S2B**	**Combined**
Boston	2.89 ± 0.23	2.89 ± 0.31	$\mathbf{2.21 \pm 0.04}$	2.23 ± 0.05
Concrete	5.48 ± 0.20	$\mathbf{5.40 \pm 0.18}$	$\mathbf{2.97 \pm 0.03}$	2.98 ± 0.03
Energy	$\mathbf{1.43 \pm 0.10}$	1.71 ± 0.20	$\mathbf{1.27 \pm 0.04}$	1.35 ± 0.05
Kin8nm	0.06 ± 0.00	0.06 ± 0.00	$\mathbf{-1.36 \pm 0.02}$	-1.35 ± 0.02
Naval	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{-6.00 \pm 0.08}$	-5.89 ± 0.35
Power	2.95 ± 0.09	2.95 ± 0.08	2.53 ± 0.03	$\mathbf{2.53 \pm 0.02}$
Protein	$\mathbf{3.54 \pm 0.13}$	3.67 ± 0.13	2.74 ± 0.04	$\mathbf{2.70 \pm 0.05}$
Wine	$\mathbf{0.58 \pm 0.03}$	0.59 ± 0.03	$\mathbf{0.99 \pm 0.03}$	1.00 ± 0.03
Yacht	$\mathbf{2.38 \pm 0.44}$	3.97 ± 1.06	$\mathbf{1.03 \pm 0.07}$	1.17 ± 0.11