University of St.Gallen

Bachelor's Thesis

Submitted in partial fulfillment of the requirements for the degree:

Bachelor of Arts in
Economics

Liquid Time-Constant Neural Networks: A Continuous-Time Approach to Volatility Forecasting

Abstract

Addressing the limitations of discrete-time models in handling irregular financial data, this thesis explores Liquid Time-Constant Networks (LTCs) – biologically-inspired neural networks operating in continuous time via ordinary differential equations – for daily volatility forecasting. This work evaluates LTCs against the RealGARCH model using a kernel-estimated realized volatility measure derived from high-frequency IBM stock data (2000-2024). The two models are evaluated on their one-step-ahead forecasts throughout the COVID-19 crisis and the following years until 2024. LTCs, especially when incorporating logarithmically-transformed time intervals between observations, consistently outperform RealGARCH, achieving exceptionally high statistical significance outside the crisis period. Trained in only a few minutes on a single CPU core, the LTCs demonstrated exceptional computational and data efficiency while hinting at a non-linear relationship between market closure duration and volatility dynamics.

Keywords: Volatility Forecasting, Liquid Time-Constant Networks, RealGARCH, High-Frequency Data, Continuous Time

Author:
Luca Boschung
22-607-972
Zwinglistrasse 15, 9000 St. Gallen
luca.boschung@student.unisg.ch

Supervisor:
Prof. PhD. Lyudmila Grigoryeva
Faculty of Mathematics and Statistics
University of St. Gallen

19th May 2025

Contents

List of Abbreviations

ARCH Autoregressive Conditional Heteroskedasticity

ARMA Autoregressive Moving Average

BPV Bipower Variation

BPTT Backpropagation Through Time

CAM Content Addressable Memory

CfC Closed-form Continuous-depth Network

CT-GRU Continuous-Time Gated Recurrent Unit

CTRNN Continuous-Time Recurrent Neural Network

DCM Dynamic Causal Model

GARCH Generalized Autoregressive Conditional Heteroskedasticity

GRU Gated Recurrent Unit

HAR Heterogeneous Autoregressive

IV Integrated Variance

IVV Integrated Volatility of Volatility

LSTM Long Short-Term Memory

LTC Liquid Time-Constant Network

LTH Lottery Ticket Hypothesis

NCP Neural Circuit Policies

NN Neural Network

MAD Mean Absolute Deviation

MSE Mean Squared Error

MLP Multilayer Perceptron

ODE Ordinary Differential Equation

QLIKE Quasi-Likelihood

RealGARCH Realized GARCH

RV Realized Variance

RNN Recurrent Neural Network

TAQ Trade and Quote

TSRV Two Scale Realized Variance

TSRSV Two Scale Realized Spot Variance

UHFD Ultra High Frequency Data

1 Introduction

Following the seminal papers by Mandelbrot (1963) and Fama (1965) there has been an ever-growing interest in the mathematical study of speculative returns. While returns appear approximately serially uncorrelated and hence extraordinarily difficult to predict consistently, it is well accepted that the volatility of returns is highly predictable. This has wide-ranging implications for the pricing of derivatives, risk management practices, and portfolio optimization more generally. Hence, volatility stands as a cornerstone concept in financial markets and risk management. However, despite the near universal need to quantify this characteristic of investment opportunities, no single superior approach has emerged among the many proposed.

One major challenge is that volatility is not directly observable from price data, and must hence be estimated. Because squared daily returns are an extraordinarily noisy estimate (Andersen & Bollerslev, 1998), there have been increasing efforts to use daily intraday returns computed from high frequency data to obtain a more informed estimate. Despite this practice being pioneered thirty years ago by Foster and Nelson (1994), this is still an active field of research. While offering richer information about intraday price movements, high-frequency data introduces additional complexities in the form of intraday seasonality, market microstructure noise, discontinuities, and great irregularity in observation timing.

Traditional approaches to volatility forecasting have predominantly relied on the Autoregressive Conditional Heteroskedasticity (ARCH) (Engle, 1982), Generalized Autoregressive Conditional Heteroskedasticity (GARCH) family (Bollerslev, 1986), and the Heterogeneous Autoregressive (HAR) family of models (Corsi, 2009) and their numerous sophisticated extensions. These models capture key stylized facts of volatility, such as clustering and mean reversion, but typically assume regular time intervals between observations. Market closures, holidays, and varying trading hours create temporal discontinuities that these standard models are not designed to handle. When faced with irregular data, practitioners either resort to imputation techniques or typically ignore the temporal irregularities, potentially introducing bias into their forecasts.

Liquid Time-Constant Networks (LTCs), introduced by Hasani (2020), offer a promising alternative framework. Unlike traditional Recurrent Neural Networks (RNNs) that process data in discrete steps, LTCs model continuous-time dynamics through ordinary differential equations. This property makes them inherently suitable for irregular time series, as they can naturally account for varying time intervals between observations by adjusting the integration time of their internal ODE solver. Furthermore, their sparse, biologically-inspired architecture offers computational efficiency while maintaining expressive power. Neural Networks more generally are an attractive tool for the prediction of volatility, as their flexibility in principle enables them to learn much more subtle relationships from the data than one could specify with a parametric model.

To evaluate the capabilities of LTCs, they are compared to the Realized GARCH (RealGARCH) model, a variation of the classical GARCH specification that incor-

porates a Realized Variance (RV) measure in its formulation (Hansen, Huang, & Shek, 2012). For this purpose a realized measure is constructed using Ultra High Frequency Data (UHFD) data of the IBM stock from 2000 to 2024 inclusive, using a cutting edge estimation technique by Figueroa-López and Wu (2024). The models are evaluated on their one-step-ahead forecast ability in the period from 2020 to 2024, which crucially contains the COVID-19 crisis.

This thesis hence addresses three primary research questions: (1) Can LTCs outperform established econometric models in one-step ahead volatility forecasting? (2) Does explicit incorporation of temporal information improve forecast accuracy in LTCs? (3) How do these models perform across different market regimes, specifically during crisis versus non-crisis periods?

The findings show that properly configured LTCs can significantly outperform the RealGARCH benchmark, particularly during periods of normal market activity. Furthermore, the incorporation of temporal information through a logarithmic transformation of time intervals between observations, but not a linear transformation, enhances predictive accuracy, suggesting a non-linear relationship between market closure duration and subsequent volatility dynamics.

The remainder of this thesis is organized as follows. Section 2 reviews the relevant literature on volatility modeling, the estimation of realized volatility, and the challenges of evaluating competing volatility models. Section 3 embeds the innovations of LTCs in the context of continuous-time Neural Networks (NNs) and illustrates their biological inspiration and practical advantages. Section 4 describes the data used and the process of implementing the realized volatility estimate. Section 5 presents the methodology of the empirical analysis. Section 6 presents the results, while section 7 discusses them. Finally, section 8 concludes and suggests directions for future research.

2 Volatility Model

At its core, volatility attempts to quantify the intuitive notion of unpredictability of prices as signified by the magnitude and frequency of their changes over time. The more erratic the behavior of a price series, the higher its volatility. This measurement proves essential for applications ranging from derivatives pricing to portfolio optimization and risk assessment. Generally speaking, an investor might be considering an investment with a continuously compounded return r_t given the information set $\mathcal{F}_s$ available at time $s < t$ and hence be interested in $\text{var}(r_t \mid \mathcal{F}_s)$. The return r_t represents the continuously compounded return over the interval $[s, t]$ and is defined as $Y_t - Y_s$, where Y_t is log price at time t. The information set $\mathcal{F}_s$ consists of the natural filtration generated by the price process up to time s, representing all market information available to investors at that time. However, the information set $\mathcal{F}_s$ can be prohibitively large, especially over long timespans, and the conditional distribution of returns is often unknown (Bauwens, Hafner, & Laurent, 2012; Andersen, Bollerslev, Christoffersen, & Diebold, 2006). These difficulties have led to the development of various models and estimators, each

attempting to approximate this unobservable volatility.

To capture the dynamic and erratic nature of price movements, financial economists often model volatility using stochastic processes. The common approach is to work in the probability space $\left(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P}\right)$ where $\{\mathcal{F}_t\}_{t\geq 0}$ is the natural filtration generated by the observed price process, augmented to satisfy the usual conditions of completeness and right-continuity. A common assumption is that log prices Y_t are efficient, with continuously compounded returns r_t exhibiting a finite and smoothly evolving mean. Under this framework, log prices are typically described by an Itô process

$$dY_t = \mu_t dt + \sigma_t dW_t \tag{2.1}$$

where μ_t is a predictable drift, W_t is a Brownian motion capturing random shocks, and σ_t is the volatility process. This specification ensures that the log price process is a semi-martingale, consistent with the no-arbitrage principles of financial theory, while allowing for risk premia and time-varying expected returns in the physical measure. The notional volatility $\tilde{\sigma}_t^2$ in period $t-s$ is then

$$\tilde{\sigma}_t^2 = \int_{t-s}^{t} \sigma_u^2 \, du \tag{2.2}$$

and is an approximately unbiased estimator of $\text{var}(r_t \mid \mathcal{F}_s)$ (Bauwens et al., 2012), meaning

$$\text{var}(r_t \mid \mathcal{F}_s) \simeq E[\tilde{\sigma}_t^2 \mid \mathcal{F}_s] \tag{2.3}$$

The notional volatility is also commonly referred to as Integrated Variance (IV), hence $\tilde{\sigma}_t^2 = IV(t)$. However, just like the "true" volatility it is not directly observable and must hence be estimated. A consistent estimator of notional volatility is realized variance. Suppose we observe log prices Y_t at discrete time points over the interval $[0, T]$, where observations are taken at regular intervals of size Δ. Specifically, we observe prices at times $t_i = i\Delta$ for $i = 0, 1, 2, ..., n$, where $n\Delta = T$. Then the RV is defined as

$$RV(T, \Delta) = \sum_{i=1}^{n} (Y_{t_i} - Y_{t_{i-1}})^2 = \sum_{i=1}^{n} r_{t_i}^2 \tag{2.4}$$

where $r_{t_i} = Y_{t_i} - Y_{t_{i-1}}$ represents the continuously compounded returns over intervals of length Δ. The idea behind this stems from the fact that quadratic variation of the price process Y, defined as

$$[Y, Y]_T = \lim_{\Delta \to 0} \sum_{i=1}^{n} (Y_{t_i} - Y_{t_{i-1}})^2 \tag{2.5}$$

is equivalent to the IV given the model in Equation 2.1 (Bauwens et al., 2012), suggesting that at least in principle with a high enough sampling frequency, summing squared returns gives a good estimate of IV. Due to microstructure noise and discontinuities in the price process, this is unfortunately not possible, and more

sophisticated estimation techniques have to be used, which are discussed in subsection 2.2. When one speaks of the properties of volatility, one refers to observations and inferences made with such an estimate or with the output of a parametric model such as those described below.

The most well known and well studied properties of volatility are its long-range dependence, the leverage effect, fat tails, and the influence of price jumps (for a detailed overview, see Masset, 2011). Long-range dependence refers to the persistent autocorrelation in volatility that decays very slowly over time. This leads to what is called volatility clustering, a term that describes the observation that periods of high volatility tend to be followed by high volatility periods, and similarly for low volatility periods, creating distinguishable regimes of market turbulence and calm. The leverage effect describes the negative correlation between asset returns and volatility, where price declines tend to be associated with volatility increases. Fat tails refer to the non-normal distribution of returns, where extreme events occur more frequently than would be expected under a normal distribution, resulting in a higher probability of large price movements. Finally, abrupt prices jumps have a strong positive effect on volatility, and are typically associated with a number of aftershocks. More recently, following the formulation of the heterogeneous market hypothesis (Müller et al., 1993) studies of volatility at different granularities showed that coarsely measured volatility is predictive of more finely measured volatility at a multitude of lags (both positive and negative), but not the other way around (Müller et al., 1997; Lynch & Zumbach, 2003). Moreover, there is evidence of additional non-linear relationships in volatility data, such as structural breaks and regime switches (Corsi, Audrino, & Renó, 2012).

While non-parametric methods, such as realized variance, provide consistent estimates of historical volatility, they are inherently limited in their ability to predict future volatility. This is because non-parametric approaches rely solely on past price data without imposing any structure on how volatility evolves over time. As a result, they cannot capture the persistence and clustering of volatility that are often observed in financial markets.

The systematic modeling of time-varying volatility began with parametric models. Building on the pioneering work of Engle (1982) on the ARCH model, Bollerslev (1986) introduced the GARCH model. The power of the ARCH type models is their ability to model heteroscedasticity without the need of an exogenous variable, which greatly simplifies the estimation. The major innovation of GARCH models is that they assume a specific functional form for the evolution of volatility, allowing them to capture key features such as volatility clustering and mean reversion. Importantly, ARCH type models were developed before the widespread availability of high-frequency data, and hence born out of the need to model volatility with lower-frequency (e.g., daily) returns.

The GARCH framework models conditional variance σ_t^2 as a function of past squared returns and past conditional variances, providing a parsimonious yet flexible

way to describe the time-varying nature of volatility

$$\begin{aligned} r_t &= \mu_t + \epsilon_t, \\ \epsilon_t &= \sigma_t z_t \\ \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \end{aligned} \tag{2.6}$$

where $z_t \sim$ i.i.d$(0, 1)$. The conditional mean μ_t of returns is typically also modeled as an Autoregressive Moving Average (ARMA) process (Box, Jenkins, Reinsel, & Ljung, 2015). The inclusion of past volatility estimates allows it to better capture the long memory of volatility. GARCH models achieve remarkable performance with a low number of parameters, making them both parsimonious and flexible. The most commonly used specification is the GARCH(1,1) model, which includes one lag for past squared residuals (ARCH term) and one lag for past conditional variances (GARCH term). Even by today's standards it is a very capable model.

The GARCH framework has inspired numerous extensions, such as Exponential GARCH (EGARCH), Threshold GARCH (TGARCH), and Integrated GARCH (IGARCH), which allow for asymmetric effects, leverage effects, and other stylized facts of financial data. The model chosen here is RealGARCH, which models returns and realized volatility simultaneously in order to predict volatility (Hansen et al., 2012). The core idea behind it is that the realized measure of volatility gives a much better signal of volatility than a single return, which allows it to adapt more quickly to volatility changes. The model is as follows

$$\begin{aligned} r_t &= \sigma_t z_t \\ \sigma_t^2 &= \omega + \beta \sigma_{t-1}^2 + \gamma x_{t-1} \\ x_t &= \xi + \varphi \sigma_t^2 + \tau\left(z_t\right) + u_t \end{aligned} \tag{2.7}$$

it resembles the classical GARCH formulation, but it has the RV measure x_t and the third equation is hence called the measurement equation, with $u_t \sim i.i.d.(0, \varsigma^2)$. The model has shown superior performance compared to the base GARCH (Hansen et al., 2012).

2.1 Evaluation

The evaluation of volatility models is inherently problematic because of the latent nature of volatility. While there are ways to test the performance of a volatility model based on point forecasts, many evaluations are not interested in the predicted volatility value itself, but in a related economic or statistical quantity. These include Value-at-Risk evaluation, interval forecasts, probability forecasts, market timing tests, and density forecast evaluation (for an overview see, e.g., Andersen, Bollerslev, Christoffersen, & Diebold, 2005). The idea behind the point estimate evaluation is of course not to get an exact measure of the ability of the model to forecast the latent volatility, but to obtain a correct ranking of alternative models. Say that σ_t^2 represents the latent volatility and $\hat{\sigma}_t^2$ for $t = 1, ..., n$ an estimate (proxy) for

it, such as RV. Let $p_{i,t}$ and $p_{j,t}$ represent volatility forecasts (predictions) from two competing models i and j respectively for $t = 1, ..., n$. To evaluate these forecasts, we use a loss function $L(x, y)$ that measures the discrepancy between the ground truth x and its forecast y. The point estimate evaluation is useful to judge the relative quality of the models if $E[L(\hat{\sigma}_t^2, p_{i,t})] < E[L(\hat{\sigma}_t^2, p_{j,t})]$ implies that $E[L(\sigma_t^2, p_{i,t})] < E[L(\sigma_t^2, p_{j,t})]$ (Hansen & Lunde, 2006). Whether this is true or not depends on the loss function and the volatility proxy. Hansen and Lunde (2006) identify the Mean Squared Error (MSE), defined as $L(\hat{\sigma}_t^2, p_t) = E(\hat{\sigma}_t^2 - p_t)^2$, as a robust loss function, meaning if the proxy is conditionally unbiased, it preserves the ranking of volatility models. They also show that when using a very noisy proxy, such as squared returns, and non-robust loss function one can obtain an incorrect model ranking. They advocate instead for the use of realized volatility measures as a proxy, because of how less noisy and hence efficient they are. Patton (2011) additionally identifies the Quasi-Likelihood (QLIKE), defined as $L(\hat{\sigma}_t^2, p_t) = \log p_t + \hat{\sigma}_t^2 / p_t$, as another robust loss function, that is also less sensitive to outliers in the predictions p_t than MSE. These results are important because it had been a common issue in literature to obtain inconsistent ranking of forecasts depending on the error metric (e.g., Akgiray, 1989; Tse, 1991; Kuen & Hoong, 1992; Lamoureux & Lastrapes, 1993; Hamilton & Susmel, 1994; Bollerslev & Ghysels, 1996; Hansen & Lunde, 2005) as pointed out already by Brailsford and Faff (1996). Moreover, the initially common evaluation against squared returns had generated the false impression that volatility models had little to no predictive power (Andersen & Bollerslev, 1998).

2.2 Estimation of Integrated Variance

In their paper addressing the unwarranted dismissal of volatility models, Andersen and Bollerslev (1998) demonstrate that the apparent low predictive power of these models when evaluated against squared daily returns is entirely expected. This occurs because squared daily returns are an extremely noisy proxy for integrated variance, and Andersen and Bollerslev (1998) even suggest that the measurement error outweighs the variance of volatility by over an order of magnitude. They show that these models are actually very capable of predicting realized variance. While theory would suggest that the higher the sampling rate of returns for the construction of RV the better an estimate of IV it will be, in practice this is not the case. High frequency returns (of say a few minutes or less) are said to be contaminated by microstructure noise. That is, these returns present statistical properties that violate models like Equation 2.1, such as negative first-order autocorrelation, unit roots, and leptokurtosis, which obscure the distinction between price movements from market frictions (Goodhart & Figliuoli, 1991; Zhou, 1996). This noise is typically modeled as

$$Y_t = X_t + \epsilon_t \tag{2.8}$$

where Y_t is the observed price, X_t is the efficient price, and $\epsilon \in \{\epsilon_t\}$ is the noise process. This process is assumed to be i.i.d with zero mean and independent of the process X in the simplest models, but often it is allowed to depend on X in

some limited way (Bauwens et al., 2012). The primary causes of these frictions are widely known to be bid-ask spreads, where transaction prices bounce between bid and ask quotes; discrete price grids imposed by minimum tick sizes; delayed price adjustments to information; strategic order splitting by large traders; and order book imbalances. However, the distinction between price movements and microstructure noise becomes easier at lower frequencies, which lead to the scientifically paradoxical practice of seeking a more accurate measure with less data, typically with 5-minute returns (Andersen & Bollerslev, 1998; Andersen, Bollerslev, Diebold, & Labys, 1999; Andersen, Bollerslev, Diebold, & Ebens, 2001).

Beyond microstructure noise, another critical feature of high frequency financial time series that makes the model of Equation 2.1 a bad approximation, are jumps. These are sudden large price movements that cannot be captured by continuous diffusion processes, even when allowing for conditional heteroskedasticity (Jorion, 1988; Bates, 1996). To accommodate jumps, efficient prices are modeled as a continuous-time jump diffusion process with the following differential equation

$$dX_t = \mu_t dt + \sigma_t dW_t + d\kappa_t \tag{2.9}$$

where $\kappa_t = j_t \cdot \mathbb{1}(q_t = 1)$ is the jump process, with q_t being a counting process with (possibly) time varying intensity λ_t, meaning $P(dq_t = 1) = \lambda_t dt$, such that κ_t is always zero, except when a jump occurs, which then has size j_t (Andersen et al., 2006; Andersen, Bollerslev, & Diebold, 2007). These jumps pose a difficulty in the estimation of IV because the quadratic variation of the price process is now

$$[X, X]_T = \int_0^T \sigma_s^2 \, ds + \sum_{0<s\leq T} \kappa_s^2 \tag{2.10}$$

implying that the realized variance (see Equation 2.4) includes also the jumps in its IV estimation, and is hence not consistent anymore (e.g., Barndorff-Nielsen & Shephard, 2004). Various non-parametric approaches have been proposed to isolate the volatility component of quadratic variation.

One of them is as the Bipower Variation (BPV) (Barndorff-Nielsen & Shephard, 2004; Kabanov et al., 2006) which led to significant out of sample volatility forecasting improvements in a variety of contexts, highlighting the importance of excluding jumps from volatility estimates (Andersen et al., 2007). BPV is defined as

$$BPV = \frac{\pi}{2} \sum_{i=2}^{n} |\Delta_{i-1}^n Y||\Delta_i^n Y| \tag{2.11}$$

where $\Delta_i^n Y = Y_{t_i} - Y_{t_{i-1}}$ are returns of log prices Y_{t_i} observed at discrete time points $t_i = i\Delta$ over the interval $[0, T]$, with observations taken at regular intervals of size $\Delta = T/n$.

Another widely used one is the Two Scale Realized Variance (TSRV) estimator developed by L. Zhang, Mykland, and Aït-Sahalia (2005) by revisiting the idea of sub-sampling prices for the calculation of returns. Consider the grid that contains all the observation points

$$\mathcal{G} = \{t_0, \ldots, t_n\} \tag{2.12}$$

is partitioned into K nonoverlapping grids, $\mathcal{G}^{(k)} \subseteq \mathcal{G}$, $k = 1, \ldots, K$, meaning

$$\mathcal{G} = \bigcup_{k=1}^{K} \mathcal{G}^{(k)}, \quad \text{where } \mathcal{G}^{(k)} \cap \mathcal{G}^{(l)} = \emptyset \text{ when } k \neq l. \tag{2.13}$$

If $t_i \in \mathcal{G}^{(k)}$ then the previous element in $\mathcal{G}^{(k)}$ is denoted by $t_{i,-}$ and the successive element as $t_{i,+}$. That means that $t_{i,-} < t_{i-1}$ and $t_{1,+} > t_{i+1}$. Let $n_k = |\mathcal{G}^{(k)}|$ denote the number of returns that can be calculated with the observations made on the time points in $\mathcal{G}^{(k)}$, which corresponds to the number of time points in $\mathcal{G}^{(k)}$ minus one. This implies that $|\mathcal{G}| = n$. One can then define

$$\overline{n} = \frac{1}{K} \sum_{k=1}^{K} n_k. \tag{2.14}$$

The realized variance on such a subgrid $\mathcal{G}^{(k)}$ is

$$RV_T^{(k)} = \sum_{t_i, t_{i,-} \in \mathcal{G}^{(k)}} (Y_{t_i} - Y_{t_{i,-}})^2. \tag{2.15}$$

L. Zhang et al. (2005) show that the estimator

$$RV_T^{(avg)} = \frac{1}{K} \sum_{k=1}^{K} RV_T^{(k)} \tag{2.16}$$

provides a better estimation of IV compared to simply squaring the log returns of sparsely sampled prices, even if the sampling rate is optimal. However, even $RV_T^{(avg)}$ is biased. They use the fact that the variance of the noise $\gamma = E(\epsilon^2)$ can be consistently approximated by

$$\hat{\gamma} = \frac{1}{2n} RV_T \tag{2.17}$$

where RV_T is simply the realized volatility calculated with all price observations. Hence, their final estimator is

$$TSRV_T = RV_T^{(avg)} - \frac{\overline{n}}{n} RV_T \tag{2.18}$$

which takes its name from the fact that it uses realized volatility over two different time scales for its estimation, but crucially also is capable of providing an approximately unbiased estimate using complete tick-by-tick data. Zu and Boswijk (2014) construct a spot variance estimator based on TSRV which they call Two Scale Realized Spot Variance (TSRSV) estimator. They use the fact that the IV over a period $[t, t-h]$ is

$$\tilde{\sigma}^2_{[t,t-h]} = \int_{t-h}^{t} \sigma_s^2 \, ds = [X, X]_t - [X, X]_{[t-h]} \tag{2.19}$$

and hence the spot volatility is

$$\sigma_t^2 = \frac{d[X, X]_t}{dt} \tag{2.20}$$

and approximate it as

$$TSRSV_t = \frac{\text{TSRV}_{t+b/2} - \text{TSRV}_{t-b/2}}{b} \tag{2.21}$$

where b is the bandwidth of their estimator, which they determine based on the charateristics of the data.

An entirely different family of estimators is what is called pre-averaging estimators. The fundamental idea is to mitigate microstructure noise by creating "pre-averaged returns." These are formed by taking weighted averages of observed returns $\Delta_n^i Y$ over a local window of size k_n, using a suitable weighting function g (Hautsch & Podolskij, 2013). The estimator is primarily based on the sum of squares of these pre-averaged returns. However, this introduces a bias due to the remaining noise variance. To correct this, a term proportional to the standard realized variance of the observed returns is subtracted (Jacod, Li, Mykland, Podolskij, & Vetter, 2009; Hautsch & Podolskij, 2013). Both the main term and the bias correction are appropriately scaled using factors derived from k_n, Δ_n, and the function g. The window size k_n is chosen to be of order $\Delta_n^{-1/2}$ to optimally balance the trade-off between the variance from the efficient price process and the bias from the noise (Jacod, Podolskij, & Vetter, 2010), allowing the estimator to achieve the optimal $n^{-1/4}$ convergence rate (Podolskij & Vetter, 2009).

An alternative and widely adopted non-parametric approach for estimating spot volatility, leverages kernel smoothing techniques. The core idea is to compute a weighted average of local squared returns (or related quantities) around the time point t, where the weights are determined by a kernel function $K_b(\Delta t)$ with bandwidth parameter b, such that $K_b(\Delta t) = K(\Delta t/b)/b$. A common form of the kernel spot volatility estimator is

$$\hat{\sigma}_{t_i}^2 = \sum_{j=1}^{n} K_b(t_j - t_i)(\Delta_j^n Y)^2 \tag{2.22}$$

where the weight of the kernel depend on the distance between the observation time t_{j-1} and the target time t_i, scaled by the bandwidth b. The bandwidth plays a crucial role, controlling the trade-off between bias and variance. That's because it controls the locality of the averaging by determining how quickly observations loose importance as a function of their distance to t_i. Early kernel estimators for spot volatility were introduced by Foster and Nelson (1994). Significant advances came from Fan and Wang (2008) and Kristensen (2010). Partially addressing the challenge of jumps, work by Yu, Fang, Li, Zhang, and Zhao (2014) and Mancini, Mattiussi, and Renò (2015) extended these approaches, though all with suboptimal convergence rates. Optimal convergence rate was achieved by the estimator by Jacod and Protter (2011).

The kernel estimator chosen in this work is the one by Figueroa-López and Wu (2024), who use a two-sided kernel and achieve optimal convergence rate, while allowing for jumps and leverage effects. One of their core ideas is to combine pre-averaging and kernel smoothing, an innovative approach that has almost not been explored at all in the literature. Their spot volatility estimate is of the form

$$\hat{\sigma}^2_{t_i} = \sum_{j=1}^{n} K_b(t_j - t_i) IV^{pre-av} \tag{2.23}$$

where IV^{pre-av} is the pre-averaging spot volatility estimator by Jacod et al. (2009). They show theoretically and through Monte Carlo simulations the superior performance of exponential kernels, and compare their estimator to the TSRSV estimator by Zu and Boswijk (2014) (Equation 2.21) and find that it has significantly higher accuracy, especially when noise levels are high. The work by Figueroa-López and Wu (2024) stands out by its comprehensiveness and the generality of its results, making it an obvious choice for the estimation of integrated variance. The implementation of their estimator is illustrated in subsection 4.1.

2.3 Conceptual Tension in Volatility Modeling

In order to proceed with this thesis, it is necessary to address what appears to be a conceptual tension within volatility theory. The notion of volatility seems to conflate an ontological with an epistemological idea. On one hand it is treated as an epistemological concept since it is fundamentally defined as the conditional expected second moment of returns $\text{var}(r_t|\mathcal{F}_s), s < t$ conditional on a theoretical information set $\mathcal{F}_s$. This information set constitutes the natural filtration generated by the price process up to time s, representing all market information available to investors at that time. That is, volatility is thought of as some sort of (ideal) prediction, meaning its real value depends on our state of knowledge (even if idealized). On the other hand it is treated as an ontological concept because it is treated as a quantity that can be measured and estimated without any regard to any such information set, real or imaginary. Realized variance as a "measurement of realizations of the latent volatility process without the need to rely on an explicit model" (Andersen et al., 2006, p. 830) is deemed good enough that "for practical purposes we can treat the daily volatility as *observed*" (Andersen et al., 1999, p. 2, italics in the original), meaning it "provides the natural benchmark for forecast evaluation purposes" (Andersen et al., 2006, p. 830). And even if one cannot exploit all observations one has at one's disposal to due microstructure noise, it is accepted that "as the sampling frequency of the returns approaches infinity, they are also, in theory, free from measurement error" (Andersen et al., 2001, p. 45).

Measuring or estimating the volatility of a period t, say a day, only with data from t, assumes that volatility is in fact a real property of the process that generates the price series. If volatility is instead assumed to be a property of our state of knowledge, then such a measurement is inappropriate and futile, because such a conditional quantity is indeed latent, in that the data from t alone is uninformative

about how much of it we could have known beforehand and hence predicted. The problem is that, naturally, no quantity can contemporarily depend on our knowledge and be a real property of the markets.

To be clear, this analysis does not aim to determine which conceptualization of volatility—epistemological or ontological—is "correct". Rather, it highlights that the field simultaneously employs both characterizations without directly addressing their incompatibility. Hence, it shouldn't be viewed as a critique of either characterization, but of the simultaneous usage of both.

While this conceptual tension in how volatility is treated remains unresolved, and its origins in the theoretical development of the field lie beyond the scope of this analysis, the empirical success of volatility forecasting is widely recognized. The practical utility of these predictions for risk management, option pricing, and portfolio allocation has been extensively documented. Therefore, this thesis will proceed with the empirical analysis of volatility forecasting, acknowledging but setting aside these deeper theoretical questions in favor of pragmatic application. However, it is reasonable to expect that these underlying theoretical contradictions will, or already do, limit or hinder the performance and applicability of these models. Therefore, it is important to acknowledge that the pragmatic application of these models does not necessarily resolve the underlying theoretical questions.

3 Liquid Time-Constant Neural Networks

3.1 Background

Traditional RNNs and their variants like Long Short-Term Memory (LSTM) networks (Hochreiter, 1997; Graves, Mohamed, & Hinton, 2013) or Gated Recurrent Units (GRUs) (Cho et al., 2014; Cho, 2014) process temporal data using discrete time steps and fixed memory mechanisms. The discretization of events is problematic because it introduces noise by destroying some valuable information in the exact time interval between two events, and because it reduces the precision of the prediction of the timing of future events. Consequently, traditional RNNs are incapable of handling missing values, the imputation of which can further hinder prediction as the missingness and its timing can be informative in a number of contexts (Lipton, Kale, & Wetzel, 2016; Che, Purushotham, Cho, Sontag, & Liu, 2018). One proposed solution was to mark the timings of events and feed those as an additional feature to the RNN (Du et al., 2016; Choi, Bahadori, Schuetz, Stewart, & Sun, 2016).

A more specialized and sophisticated solution are NNs whose state transitions are defined by continuous-time dynamics (Rubanova, Chen, & Duvenaud, 2019) through the use of neural Ordinary Differential Equations (ODEs) (Chen, Rubanova, Bettencourt, & Duvenaud, 2018). The idea behind this family of models is to add more layers and take smaller steps in the transformation of the hidden

state $h(T)$, which in the limit results in a continuous dynamic described by an ODE

$$\frac{dh(t)}{dt} = f(h(t), t, \theta) \tag{3.1}$$

where θ are the parameters of the neural network. The hidden state $h(T)$ can therefore be evaluated at any point in time T as the solution to the initial value problem of this ODE. This approach boasts several benefits. It naturally handles data that arrives at irregular intervals, it can be trained with constant memory cost as a function of depth, and the ability to tune its accuracy through settings of the ODE solver at inference (Chen et al., 2018).

The training of Neural ODEs differs from traditional networks because instead of backpropagating through discrete layers, one must compute gradients through the continuous trajectory of the ODE solver itself, which is accomplished using, for example, the adjoint sensitivity method (Pontryagin, 2018). This method solves a backward ODE to compute gradients with respect to both the network parameters and the dynamics of the hidden state, avoiding the need to store intermediate computations of the forward pass (Chen et al., 2018).

The perhaps most obvious application of this insight are RNNs in which the state transformation of the individual cells is modeled with an ODE, which have been coined ODE-RNNs (Rubanova et al., 2019). A second family of networks born from the work of Chen et al. (2018) are generative latent-variable time series models. This kind of model is not autoregressive and hence stops being a RNN and instead are a kind of variational autoencoder. They represent a time series with a latent trajectory, where a sequence of latent states $z_0, z_1, ..., z_N$ corresponds to the time series $x_1, x_2, ..., x_N$. The initial local state z_0 (whose distribution is determined by the encoder) together with the global set of dynamics determine the trajectory. To obtain a prediction at any point in time, the latent space is decoded:

$$\begin{gathered} z_0 \sim q(z_0|(x_1, x_2, \ldots, x_N), \theta_e) \\ z_1, z_2, \ldots, z_N = \text{ODESolve}(z_0, f, \theta_f, (t_0, t_1, \ldots, t_N)) \\ \text{each} \quad x_i \sim p(x|z_i, \theta_d) \end{gathered} \tag{3.2}$$

where $q(\cdot|\cdot, \theta_e)$ is the encoder with parameters θ_e, f is the neural ODE function with parameters θ_f that defines the dynamics, and $p(\cdot|\cdot, \theta_d)$ is the decoder with parameters θ_d. The function ODESolve integrates the dynamics function f from initial state z_0 over the time points $(t_0, t_1, \ldots, t_N)$.

While the decoder is a feed-forward neural network, the encoder is either a standard RNN (Chen et al., 2018) or an ODE-RNN (Rubanova et al., 2019) which makes for a more efficient parametrization and makes the model purely continuous. All these components are then trained end-to-end. These models function similarly to the sequence-to-sequence approach introduced by Sutskever (2014) in that they first encode a variable-length input sequence into a fixed-dimensional embedding, and then decode this embedding to generate a variable-length output sequence during inference.

An older approach is that of Continuous-Time Recurrent Neural Networks (CTRNNs). These models, despite their name, are not (necessarily) recurrent in

design. Rather, they are a collection of neurons with continuous inputs and outputs that are connected arbitrarily (i.e., typically not fully connected). They were originally invented by Hopfield (1984), specifically to mimic the design of biological neurons and to give the NN some of their properties, such Content Addressable Memory (CAM) (associative memory). The internal state of those neurons is modeled by a differential equation of the form

$$\frac{d\mathbf{h}(t)}{dt} = -\frac{\mathbf{h}(t)}{\tau} + f(\mathbf{h}(t), \mathbf{I}(t), t, \theta) \tag{3.3}$$

where $\mathbf{h}(t)$ is the vector of hidden states across n neurons, $\mathbf{I}(t)$ is the external input vector, τ is a time constant controlling the decay rate of the states, and θ is the vector of parameters for the nonlinear function f which typically includes connection weights between neurons.

In the context of CAM a stable state can be thought of as the memory most closely associated with the input. Hence, the stability of the system is of fundamental concern, and much effort has flown into the study of the stability of CTRNNs (for an overview see H. Zhang, Wang, & Liu, 2014). In fact, this formulation is more stable than that of Equation 3.2 thanks to the term with the time constant τ. CTRNNs are also very powerful as they have been shown to be universal approximators, meaning, they can approximate any input-output to an arbitrary level of precision (Funahashi & Nakamura, 1993).

3.2 The innovations of Liquid Time-Constant Networks

LTCs build upon CTRNNs by modifying the equation that characterizes the evolution of the neurons. Hasani, Lechner, Amini, Rus, and Grosu (2021) propose that the hidden state flow follow the equation

$$\begin{aligned} \frac{d\mathbf{h}(t)}{dt} &= -\frac{\mathbf{h}(t)}{\tau} + f(\mathbf{h}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{h}(t)) \\ &= -\left[\frac{1}{\tau} + f(\mathbf{h}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{h}(t) + f(\mathbf{h}(t), \mathbf{I}(t), t, \theta)A \end{aligned} \tag{3.4}$$

where A is an additional parameter. The most notable feature of this kind of system is that the effective time constant is not simply τ, but is also determined by the neural network f since

$$\tau_{sys} = \frac{\tau}{1 + \tau f(\mathbf{h}(t), \mathbf{I}(t), t, \theta)} \tag{3.5}$$

which allows the neurons to adapt their time constant to the inputs, allowing them to capture idiosyncratic non-linear dynamics very succinctly and causally. This way of increasing the amount of calculation that each neuron is capable of is what gives Liquid Time-Constant Neural Networks their name (Hasani et al., 2021).

The design of LTC neurons mimics that of biological neural circuits, where nonlinear synaptic propagation introduces an input-dependent, variable time constant

for each neuron (Hasani, 2020). It is inspired by the neurons of small animals such as *Ascaris* (Davis & Stretton, 1989), *Leech* (Lockery & Sejnowski, 1992), and *C. Elegans* (Wicks, Roehrig, & Rankin, 1996) which don't present the all or nothing action potentials so commonly observed in the animal kingdom, and instead present graded active responses (the neurons are non-spiking). The nervous system of *C. Elegans* in particular has been extensively studied and modeled with ODEs (Gleeson, Lung, Grosu, Hasani, & Larson, 2018; Szigeti et al., 2014). The potential $h_i(t)$ of such a neuron i can be modeled as a linear ODE of the form (Koch & Segev, 1998):

$$\frac{dh_i}{dt} = \frac{-g_{l_i} \cdot (h_i(t) - h_{Rest_i})}{C_{m_i}} + \frac{\sum_{j=1}^{n} I_{ij}}{C_{m_i}}. \tag{3.6}$$

In this equation, h_{Rest_i} is the resting potential of the neuron, meaning, the potential the neuron returns to over time when not receiving inputs or firing (it is idle). g_{l_i} is the leakage conductance of the membrane of the neuron, representing the charge that leaks through the membrane by default, meaning, the transfer of charge through the channels in the membrane that are consistently active regardless of neural state. I_{ij} is the represents the external current (input) from neuron j to neuron i. C_{m_i} is the capacitance of the membrane which measures the difference in charge that traverses the membrane for a given electric potential. It represents the membrane's ability to store charge, and hence in the equation determines how much the potential changes as a reaction to the charge leaking from the membrane and to the inputs. From this formulation it is easy to see that $\tau_i = C_{m_i}/g_{l_i}$ represents the intrinsic time constant of the neuron, which affects how quickly it responds to inputs and returns to its resting state in the absence of synaptic effects. The synaptic inputs I_{ij} can be modeled as a sigmoid non-linearity with parameters (γ_{ij}, μ_{ij}) (Wicks et al., 1996; Koch & Segev, 1998):

$$I_{ij} = \frac{w_{ij}(E_{ij} - h_i(t))}{1 + e^{-\gamma_{ij}(h_j(t) + \mu_{ij})}} \tag{3.7}$$

where w_{ij} is the weight of the connection from neuron j to neuron i, and E_{ij} is the reversal potential of that synaptic connection. Its value determines the sign of the input, and hence if the synapse from j has an excitatory or inhibitory effect on the target neuron i. If we now denote the sigmoid non-linearity as $\sigma_i(h_j(t)) = 1/(1 + e^{-\gamma_{ij}(h_j(t)+\mu_{ij})})$, we can write the state dynamics of neuron i as it receives one synapse from neuron j as

$$\frac{dh_i}{dt} = \frac{-g_{l_i}(h_i(t) - h_{Rest_i})}{C_{m_i}} + \frac{w_{ij}\sigma_i(h_j(t))(E_ij - h_i(t))}{C_{m_i}} \tag{3.8}$$

which corresponds to the formulation of LTCs (Hasani, 2020). Once again, we can see the effective time constant of the system by rewriting this expression as is done in Equation 3.4

$$\frac{dh_i}{dt} = -\left(\frac{1}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(h_j(t))\right) h_i + \left(\frac{h_{Rest_i}}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(h_j(t))E_{ij}\right) \tag{3.9}$$

The interface between neurons is controlled by sparsity masks, which are binary matrices that define the connectivity structure of the network. These parameters

don't get updated during training, as they depend only on the wiring of the NN (see subsection 3.3).

Additionally, to properly interface external data with the neuronal dynamics, the model employs affine transformations at both the input and output stages. At the input stage, an affine transformation converts the raw input datapoint x_i into appropriate currents that can be processed by the neuronal dynamics:

$$I_i = w_{input} \cdot x_i + b_{input} \tag{3.10}$$

where w_{input} and b_{input} are learnable parameters that scale and shift the input data to match the operating range of the neuron. Similarly, at the output stage, another affine transformation converts the neural states into the desired output format y_i

$$y_i = w_{output} \cdot h_i + b_{output} \tag{3.11}$$

These input and output transformations serve multiple purposes: they allow the model to adapt to different scales and ranges of data, they provide additional learnable parameters that increase the model's flexibility, and they create a clear interface between the biologically-inspired internal dynamics and the external data representations.

Vorbach, Hasani, Amini, Lechner, and Rus (2021) show that differential equations can form causal structures and in particular that LTCs, as opposed to neural ODEs, are Dynamic Causal Models (DCMs) (Friston, Harrison, & Penny, 2003). DCMs are a class of models used to infer the causal architecture of dynamical systems, particularly in scenarios where the interactions between system components are governed by differential equations. Unlike traditional statistical models, DCMs are designed to capture both internal and external interventions, allowing them to model how changes in one part of the system influence other parts over time (Friston et al., 2003; Penny, Ghahramani, & Friston, 2005). This makes DCMs particularly powerful for tasks that require understanding and predicting the effects of interventions, such as in neuroscience, where they are used to model brain connectivity and the effects of external stimuli on neural activity (Breakspear, 2017; Penny et al., 2005).

Unlike standard Neural ODEs, which lack the ability to account for external interventions, LTCs are able to approximate the behavior of DCMs by incorporating input-dependent time constants. This allows LTCs to adapt their dynamics based on external inputs, effectively modeling both internal state transitions and external interventions. As a result, LTCs can learn causal mappings directly from high-dimensional sensory data (Vorbach et al., 2021).

The LTC architecture presents some important additional of benefits and improvements over neural ODEs and CTRNNs (Hasani et al., 2021; Hasani, 2020). First, they don't present numerical instability even with unbounded inputs, and they are universal approximators. LTCs exhibit greater expressivity than CTRNNs and neural ODEs. This is demonstrated by longer and more complex activation trajectories in latent space, allowing LTCs to model intricate time-series patterns more effectively. While neural ODEs often rely on the adjoint sensitivity method, which

can introduce numerical errors, LTCs use Backpropagation Through Time (BPTT) to achieve higher accuracy in gradient computation at the cost of increased memory usage. This avoids the numerical errors that arise when performing reverse-mode automatic differentiation with the adjoint sensitivity method (Gholami, Keutzer, & Biros, 2019) as is done with neural ODEs (Chen et al., 2018). Additionally, because ODEs governing LTCs realize a system of stiff equations, Hasani et al. (2021) developed a novel specialized ODE solver that combines the stability of implicit methods with the efficiency of explicit ones, addressing the issues that traditional solvers such as the explicit Euler method or Runge-Kutta (Runge, 1895) have with stiff equations (Press, 2007). As a result, the update over time span Δ of the state of the neuron i is computed as

$$h_i(t+\Delta) := \frac{h_i(t)C_{m_i}/\Delta + g_{l_i}h_{Rest_i} + \sum_{j=1}^{n} I_{ij}w_{ij}\sigma_i(h_j(t))E_{ij}}{C_{m_i}/\Delta + g_{l_i} + \sum_{j=1}^{n} I_{ij}w_{ij}\sigma_i(h_j(t))} \tag{3.12}$$

In order to increase the precision of this update, it is split it into k unfolding steps, each one updating the state of the neuron over timespan Δ/k, meaning each update computes $h_i(t+\Delta/k)$. Note that this approach naturally accommodates non-integer time intervals (Δ), allowing the model to process inputs that arrive at arbitrary, non-uniform times - a key property for modeling continuous-time systems.

LTCs outperform CTRNNs, neural ODEs, LSTMs and other RNN architectures on a variety of time series tasks, and excel in particular in modeling irregularly sampled data (Hasani et al., 2021; Hasani, 2020). During this evaluation Hasani (2020) noticed that LTCs have an increased ability compared to CTRNNs and LSTMs to generalize beyond their training domains. LTCs demonstrated a structured and symmetric extrapolation behavior, closely aligning with what a human observer might intuitively predict. This suggests that they better learned the underlying dynamics of the system, promising better sample efficiency and forecasts of previously unseen events, an ability that is particularly attractive for financial applications. However, this improved extrapolative power comes at the cost of longer training times (Hasani, 2020).

Hasani et al. (2022) directly address this limitation by introducing an approximate closed-form solution for LTC networks. By deriving an explicit mathematical expression that introduces time dependence for the evolution of neuron states, this approach eliminates the need for computationally intensive numerical solvers traditionally required for integrating the LTC differential equations. This approximation is tight, meaning it's very close to the true solution. As a result, training and inference become significantly faster—achieving speedups of one to five orders of magnitude—while maintaining the expressivity and dynamic adaptability of the original LTC architecture. They call this new type of neural network Closed-form Continuous-depth Network (CfC). Their computational efficiency makes them more scalable and hence better suited for large datasets and complex tasks. They also excel at dealing with irregularly spaced or sparse time series data, such as financial data, and have shown strong performance in tasks like autonomous driving and time-series prediction. However, they tend to suffer from a vanishing gradient issue when the sequence length is larger than a couple of hundred steps, which can be

mitigated by combining them with mixed memory architectures (Lechner & Hasani, 2020).

3.3 Neural Circuit Policies

The work discussed so far has demonstrated how the design of Liquid Time-Constant neurons is biologically inspired, mimicking the graded, non-spiking responses observed in the nervous system of simple organisms such as *C. Elegans* (Hasani, 2020; Hasani et al., 2021). Building on this foundation, Lechner et al. (2020) extend this biological inspiration to the wiring of the neural network itself, using LTC or CfC neurons to create sparse and efficient architectures.

Recently there has been ever-increasing efforts into researching sparse neural networks (for an overview see Liu & Wang, 2023), inspired especially by the proposal of the Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2018). Pruning, the practice of eliminating redundant or less significant weights or neurons from a fully connected neural network, had been around since the 1980s, and had been shown to increase generalization, sampling efficiency, and to lower inference time and costs (LeCun, Denker, & Solla, 1989; Mozer & Smolensky, 1989; Janowsky, 1989). This technique is only applicable to already trained neural networks, meaning that while inference costs can be reduced significantly, there is no improvement in training performance. The core insight that led to the formulation of the LTH was that randomly initialized fully connected networks contain sparse subnetworks (lottery tickets) that could be trained from the ground up to match or even exceed the performance of their dense counterparts. Frankle and Carbin (2018) additionally proposed an algorithm for finding these sparse high-performing subnetworks.

The brain of *C. Elegans*, composed of just 302 neurons, has a sparsity of 91%, yet gives the nematode the ability to perform complex behaviors with great degree of control (Ardiel & Rankin, 2010; Kato et al., 2015; Stephens, Johnson-Kerner, Bialek, & Ryu, 2008; Gray, Hill, & Bargmann, 2005). This led Hasani, Lechner, Amini, Rus, and Grosu (2020) to study the properties of one of its neural circuits, the Tap-Withdrawal neural circuit (Chalfie et al., 1985), both in terms of its theoretical and empirical merit. They concluded that this circuit represents a "natural lottery winner network"—a sparse, high-performing subnetwork selected by evolution for its exceptional functionality. The wiring constraints of this circuit—such as layered neuron organization, selective sparsity, and specific patterns of recurrent and feedforward connections—were found to maximize information flow through the network, enabling efficient signal propagation and superior control performance compared to arbitrary sparse architectures. Confirming prior experiments with similar wiring (Lechner, Hasani, Zimmer, Henzinger, & Grosu, 2019) LTCs wired according to the wiring constraints of the Tap-Withdrawal circuit, when trained from the ground up outperformed randomly wired LTCs, in addition to Multilayer Perceptrons (MLPs) and LSTMs on reinforcement learning tasks (Hasani et al., 2020).

With these insights, Lechner et al. (2020) introduce Neural Circuit Policies

(NCP), which take the biologically inspired design principles of the Tap-Withdrawal circuit and apply them to real-world autonomous control tasks. NCPs leverage the same sparse, layered architecture and LTC or CfC neurons to create networks that are not only efficient and high-performing but also more easily interpretable. A key contribution of their work is the demonstration that these networks can govern complex, real-time control systems—such as autonomous driving—while maintaining transparency in their decision-making processes. This interpretability is achieved through the structured wiring of the network, where the flow of information can be traced from sensory inputs to motor outputs, allowing for fine-grained auditing of the system's behavior.

Besides autonomous driving NCPs have demonstrated great performance in time series tasks (Hasani et al., 2022) and flight navigation (Vorbach et al., 2021). When evaluating NCPs on control and flight navigation tasks, Vorbach et al. (2021) found that while traditional deep learning architectures could solve the task in offline settings, they often failed when deployed in real-world scenarios, especially under changing or perturbed conditions. In contrast, NCPs demonstrated robust performance, maintaining stability and accuracy even when faced with environmental perturbations such as occlusions, dynamic obstacles, or varying weather conditions. This robustness stems from their ability to model causal relationships and adapt to external interventions.

The biologically inspired design principles of NCPs highlight the potential for leveraging insights from natural systems to create more efficient, interpretable, and robust artificial intelligence systems. This approach could pave the way for future innovations in neural network design.

4 Data

In this paper, a volatility measure is derived from the NYSE's Trade and Quote (TAQ) data. This dataset provides tick by tick records of trades and quotes from 1993 to the present, and is considered a gold standard in the industry for financial analysis with UHFD. However, only data from the year 2000 and onward is used, because the data from previous years showed some noticeably different volatility dynamics. These differences might be explained by the widespread shift to electronic trading systems, which greatly increased the liquidity in equity markets, and perhaps a more generally improved risk management among market participants following the dot-com crash. Additionally, for the sake of simplicity, only data from a single arbitrarily chosen stock (IBM) is used in the analysis. The general procedure for cleaning the data proposed by Brownlees and Gallo (2006) is followed. In particular, erroneous or delayed trades are eliminated, and when there are multiple trades for the same tick (second), only the median price is kept. Trades outside of trading hours (9:30 am - 4:00 pm) are also discarded, except for an extra 5-minute window after 4:00 pm, since the closing hours are not strict and in fact vary unpredictably between days. The exact unambiguous closing price reflecting market consensus is difficult to pinpoint due to this irregularity (especially because

TAQ data contains trades from multiple exchanges). However, for practical purposes, the last price before 4:05 pm is typically a good enough value (Brownlees & Gallo, 2006), and is also the value chosen in this work for the calculation of daily (close to close) returns.

In order to eliminate outliers the suggestion of Verousis and Ap Gwilym (2010) to use the Mean Absolute Deviation (MAD) is followed. These outliers can be due to human or technological errors, and don't reflect the actual trading activity. The difficulty in identifying them lies in the fact that their causes are near impossible to track down, and hence when eliminating outliers one risks to "overscrub" the dataset and eliminate legitimate data points (Falkenberry, 2002). The advantage of the MAD is that it is more robust to outliers than a mean and standard deviation calculation, and it is more efficient with non-normal data (Verousis & Ap Gwilym, 2010), such as UHFD, which is long tailed (Barnett, Lewis, et al., 1994; Ap Gwilym & Sutcliffe, 2001). It is computed as follows

$$\mathrm{MAD_d} = \mathrm{median}\{|p_{d,t} - \mu_d|\} \tag{4.1}$$

where $p_{d,t}$ is the price at tick t (a second in this case) of period d and μ_d is the median price of the period d. In order to determine an outlier the *Hampel X84* technique (Hellerstein, 2008) is used which classifies a data point as an outlier if

$$|p_{d,t} - \mu_d| > 1.4826 \cdot k \cdot \mathrm{MAD_d} \tag{4.2}$$

where 1.4826 is the value of the *consistency coefficient* which converts the MAD to what's called the normalized MAD allowing direct comparison with standard deviation for normally distributed data (Hubert, 2004). This means that if one were to choose a value of $k = 1$ one would classify all points that are more than one standard deviation away from the median as outliers under assumptions of normality.

While often a day is chosen as the period d for which to compute the MAD, in this work the period d is a sliding 6-minute window, which moves by 2 minutes at each step. A point is classified as an outlier if it is flagged in at least 2 windows. A rather conservative value of $k = 4$ is chosen for the computation of the normalized MAD, which resulted in about 0.3-0.8% of the non-faulty ticks as being classified as outliers (depending on the year), consistent with more detailed and involved analyses found in the literature (for an overview see Verousis & Ap Gwilym, 2010). The advantage of this sliding window approach is that it enables anomaly detection within a more localized context. If d corresponds to the entire day, when the stock has a significant upward movement, very low outliers are not found by the normalized MAD because, despite being obviously outliers upon visual inspection, they are still close enough to the daily median to not be detected, and vice versa for rapidly falling stocks. The fact that the window slides by a third of its width, means that all prices except for those in the first and last four minutes of the trading day will be examined three times. This combined with the majority voting provides the classification algorithm with both detailed insight into the local price context and sufficient patience to avoid misclassifying genuine large price movements.

Finally, the first and last five minutes of trading time are removed, due to their unusual trading activity (Gerety & Mulherin, 1992), and because of the lower accuracy of the outlier detection algorithm in those times. This means that the data to compute the realized volatility is that from 9:35 am to 15:55 pm.

4.1 Kernel-Based Realized Volatility Estimation

To obtain a state of the art realized volatility measure, the spot volatility estimator by Figueroa-López and Wu (2024) was implemented from scratch in Python, as the authors provided no reference code. Refer to subsection 2.2 for an overview of IV estimators and the reasoning for choosing this one. This implementation was optimized through extensive vectorization and enhanced numerical stability measures. Specifically, the `Decimal` library was employed for high-precision arithmetic, while vectorized operations utilized the `SciPy` library with 64 bit floating point values. These numerical precision considerations were essential due to the miniscule values involved in the calculations, and their use significantly influenced the outcomes of the calculations.

As is standard, Figueroa-López and Wu (2024) model the contamination of prices by microstructure noise in the following way

$$Y_{t_i} = X_{t_i} + \epsilon_{t_i} \tag{4.3}$$

where Y_{t_i} is the observed price, X_{t_i} is the efficient price as described by Equation 2.9, $\epsilon \in \{\epsilon_t\}$ is the noise process, and $t_i := i\Delta_n, 0 \leq i \leq n$, with $\Delta_n := T/n$ with a fixed time horizon $T \in (0, \infty)$. In their model, the noise is allowed to depend on X, but conditional on the entire process X, $\{\epsilon_t\}$ is a family of independent centered random variables. For the implementation T is taken to be a day, meaning $T := 1$. To construct a pre-avering estimator, one needs to first compute the pre-averaging window as

$$k_n = \left\lfloor \frac{1}{\theta\sqrt{\Delta_n}} + \frac{1}{2} \right\rfloor \tag{4.4}$$

where θ is a model parameter, and choose a weight function g on $[0, 1]$, which needs to be continuous, piecewise C^1 with a piecewise Lipschitz derivative g' such that $g(0) = g(1) = 0$ and $\int_0^1 g(s)^2\, ds = 1$. The function chosen by Figueroa-López and Wu (2024) and hence in this implementation is

$$g(x) = 2x \wedge (1 - x) = \min(2x, 1 - x). \tag{4.5}$$

With these one can find the sequences

$$\begin{aligned} \bar{Y}_i^n &= \sum_{j=1}^{k_n - 1} g\left(\frac{j}{k_n}\right) \Delta_{i+j-1}^n Y = -\sum_{j=1}^{k_n} \left(g\left(\frac{j}{k_n}\right) - g\left(\frac{j-1}{k_n}\right) \right) Y_{i+j-2}^n, \\ \widehat{Y}_i^n &= \sum_{j=1}^{k_n} \left(g\left(\frac{j}{k_n}\right) - g\left(\frac{j-1}{k_n}\right) \right)^2 \left(\Delta_{i+j-1}^n Y\right)^2. \end{aligned} \tag{4.6}$$

where $\bar{Y}_i^n$ is the weighted average of the high frequency log returns $\Delta_{i+j-1}Y, j = 1, \ldots, k_n - 1$, and $\widehat{Y}_i^n$ is a debiasing term. Note that these sequences are computed in a merely forward-looking manner. Now, in order to compute the pre-averaging estimators, one needs to compute an estimator weight

$$\phi_{k_n}(g) = \sum_{i=1}^{k_n} g\left(\frac{i}{k_n}\right)^2 \tag{4.7}$$

and a truncation level v_n for the jump detection, such that when $\bar{Y}_i^n > v_n$ one can induce that a jump is occurring, and hence this return should not be considered for the estimation of the spot volatility. This truncation level should satisfy

$$v_n = \alpha \left(k_n \Delta_n\right)^{\varpi} \tag{4.8}$$

with $\alpha > 0, \varpi \in (0, 0.5)$. Commonly such a level is chosen as $v_n = c\Delta_n^{\Gamma}$, where c depends on an estimate of the volatility level. In their simulation study Figueroa-López and Wu (2024) chose $v_n = 1.8\sqrt{BPV\phi_{k_n}(g)}(\Delta_n)^{0.47}$, where BPV is the Bi-Power Variation (Equation 2.11) and calculated on sparsely sampled data (5-min frequencey). However, in the present implementation this approach performed poorly, flagging an unrealistic amount of returns as jumps. Hence the approach of Jacod and Todorov (2014) was used with

$$v_n = \alpha\sqrt{BPV}(\Delta_n)^{\Gamma} \tag{4.9}$$

with $\alpha = 4$ and $\Gamma = 0.49$, and the BPV is calculated with all high frequency returns. With this approach between 0% and 2.5% of returns were flagged as jumps, depending on the day.

Figueroa-López and Wu (2024) define three pre-averagint estimators of the spot variance c_τ at $\tau \in (0, T)$: one nontruncated, and two truncated ones. The non-truncated one doesn't exclude jumps from the volatility estimation and so takes the form

$$\hat{c}\left(k_n, b_n, \tau\right) = \frac{1}{\phi_{k_n}(g)} \sum_{j=1}^{n-k_n+1} K_{b_n}\left(t_{j-1} - \tau\right)\left(\left(\bar{Y}_j^n\right)^2 - \frac{1}{2}\widehat{Y}_j^n\right) \tag{4.10}$$

while the truncated ones are

$$\begin{aligned}
\hat{c}\left(k_n, b_n, v_n, \tau\right)_1 &= \frac{1}{\phi_{k_n}(g)} \sum_{j=1}^{n-k_n+1} K_{b_n}\left(t_{j-1} - \tau\right)\left(\left(\bar{Y}_j^n\right)^2 \mathbb{1}_{\{|\bar{Y}_j^n| \le v_n\}} - \frac{1}{2}\widehat{Y}_j^n\right), \\
\hat{c}\left(k_n, b_n, v_n, \tau\right)_2 &= \frac{1}{\phi_{k_n}(g)} \sum_{j=1}^{n-k_n+1} K_{b_n}\left(t_{j-1} - \tau\right)\left(\left(\bar{Y}_j^n\right)^2 - \frac{1}{2}\widehat{Y}_j^n\right) \mathbb{1}_{\{|\bar{Y}_j^n| \le v_n\}}.
\end{aligned} \tag{4.11}$$

where $K_{b_n}(x) := K(x/b_n)/b_n$ is the kernel function with bandwidth b_n. The bandwidth takes the form $b_n = \beta\Delta_n^{1/4}$, with an initial value of $\beta = 4$, as per the recommendation of Figueroa-López and Wu (2024) based on their extensive simulations. To find the optimal kernel function $K(x)$ they generalize the results of Figueroa-López and Li (2020) which showed that using the exponential kernel $K(x) = \frac{1}{2}e^{-|x|}$

minimizes the asymptotic variance of the spot volatility estimator in absence of microstructure noise. Figueroa-López and Li (2020) had demonstrated that this kernel outperforms both uniform (e.g., Jacod & Protter, 2011) and Epanechnikov (Epanechnikov, 1969) kernels, reducing the integrated estimation error variance by approximately 14% and 6% respectively, while also providing computational advantages by reducing the time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ when estimating volatility across multiple time points.

Because of the symmetrical nature of the exponential kernel, one faces the issue that at the edges the spot volatility estimation is heavily biased downwards, up to half the true value (Müller, 1991). To alleviate these edge effects, the suggestion by Kristensen (2010) is followed by Figueroa-López and Wu (2024) and hence in this implementation, to substitute $K_{b_n}(t_{i-1} - \tau)$ with

$$K_{b_n}^{adj}(t_{i-1} - \tau) = \frac{K_{b_n}(t_{i-1} - \tau)}{\Delta_n \sum_{j=1}^{n-k_n+1} K_{b_n}(t_{j-1} - \tau)} \tag{4.12}$$

and since $T := 1$ this is the equivalent of normalizing the value of kernel by the average of all kernel values used for the given spot volatility estimate $\hat{c}_\tau$.

Based on simulations with different jump levels, Figueroa-López and Wu (2024) conclude that the second truncated estimator is better than the first truncated one at eliminating jumps, and that it is particularly superior when the jumps tend to be large. For this reason, while all three estimators are implemented in this work, the estimations given by $\hat{c}\,(k_n, b_n, v_n, \tau)_2$ are chosen.

To decrease the reliance on the initial guesses of the parameters of this estimation method, Figueroa-López and Wu (2024) provide optimization methods for the bandwidth and θ. They mention that also the truncation level is a candidate for optimization, but leave it for future research to study it. The idea of the optimizations is that the initial estimations of the spot volatilities can be used to improve the parameters of the model, until a convergence to a stable estimate is achieved.

They find that the bandwidth that minimizes the integrated asymptotic variance of the estimator is given by

$$b_n^\star = \Delta_n^{1/4} \sqrt{\frac{4\Theta(\theta; g) \int K^2(u)du}{\int_0^T \tilde{\sigma}_t^2 dt \int L^2(v)dv}} \tag{4.13}$$

where $\int_0^T \tilde{\sigma}_t^2 \, dt$ is the Integrated Volatility of Volatility (IVV), $L(t) = \int_t^\infty K(u)du\mathbb{1}_{\{t>0\}} - \int_{-\infty}^t K(u)du\mathbb{1}_{\{t\leq 0\}}$, and

$$\Theta(\theta; g) := \frac{\Phi_{22}}{\theta} \int_0^T c_t^2 dt + 2\Phi_{12}\theta \int_0^T \gamma_t c_t dt + \Phi_{11}\theta^3 \int_0^T \gamma_t^2 dt \tag{4.14}$$

where γ_t is the variance of the noise, and $\Phi_{11}, \Phi_{12}, \Phi_{22}$ are three constants that depend on $g(x)$. See Appendix A for their derivation as Figueroa-López and Wu (2024) don't provide them. However, the optimization of the bandwidth can be slightly simplified, since when the kernel is the exponential kernel $K(x) = \frac{1}{2}e^{-|x|}$,

$\int K^2(u)du = \int L^2(v)dv$, see Appendix B for proof. To obtain the values of γ_t the simplyfing assumption is made that the variance of the noise is constant during the period T and hence $\gamma_t \equiv \gamma$ is estimated as

$$\hat{\gamma} = \frac{1}{2n}\sum_{i=1}^{n}(Y_i^n - Y_{i-1}^n)^2 \tag{4.15}$$

following results by L. Zhang et al. (2005) (Equation 2.17). Since the IVV is just the quadratic variation of the squared volatility process

$$IVV_T = [\sigma^2, \sigma^2]_T = \lim_{|t_i - t_{i-1}|\to 0}\sum_{i=1}^{n}(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)^2 \tag{4.16}$$

it is estimated with the realized variance of sparsely sampled (approx 5 min) spot variance estimates

$$\widehat{IVV}_T = \sum_{i=1}^{[n/p]}\left(\hat{c}_{t_{(i)p},0} - \hat{c}_{t_{(i-1)p},0}\right)^2 \tag{4.17}$$

with a value of $p = 300$ chosen in this implementation.

Further, Figueroa-López and Wu (2024) find that the optimal value $\theta^\star$ of θ, which minimizes the integrated asymptotic variance of the truncated pre-averaing kernel estimators is such that

$$(\theta^\star)^2 = \frac{\sqrt{\Phi_{12}^2\left(\int_0^T c_t\gamma_t dt\right)^2 + 3\Phi_{11}\Phi_{22}\int_0^T \gamma_t^2 dt \int_0^T c_t^2 dt} - \Phi_{12}\int_0^T c_t\gamma_t dt}{3\Phi_{11}\int_0^T \gamma_t^2 dt}. \tag{4.18}$$

While the optimization of the bandwidth worked as expected, with stability in the estimate of the daily integrated variance being achieved typically after just a single round of optimization, consistent with the simulations of Figueroa-López and Wu (2024), the optimization of θ was problematic, as it systematically produced values of θ that were very big, leading to a very small pre-averaging window and hence implausibly low volatility values. Interestingly, Figueroa-López and Wu (2024) only use the bandwidth optimization in their simulations, while showing that the spot volatility estimates depend more on the bandwidth than on the value of θ, implying that they manually tuned the latter's values. Perhaps, this choice was not coincidental, as it was certainly not driven by computational limitations. In this implementation, a fixed value of $\theta = 5$ is chosen, as per the recommendation of Figueroa-López and Wu (2024) when dealing with prices observed each second.

To get a superficial qualitative understanding of the estimates obtained with this kernel, its values are plotted against the estimates of a naive RV estimates obtained by summing square log returns of sparsely sampled prices (5 minute frequency). In Figure 1 we can see how the two estimates compare in 2006, which when judged by the average closing price of the VIX (Whaley, 2009) was a very calm year (*VIX Volatility Index - Historical Chart*, 2025). We can see that the kernel estimate and the naive estimate tend to agree on the volatility level, and cross each other without an obvious pattern. Over the entire year, the kernel estimate was below the naive

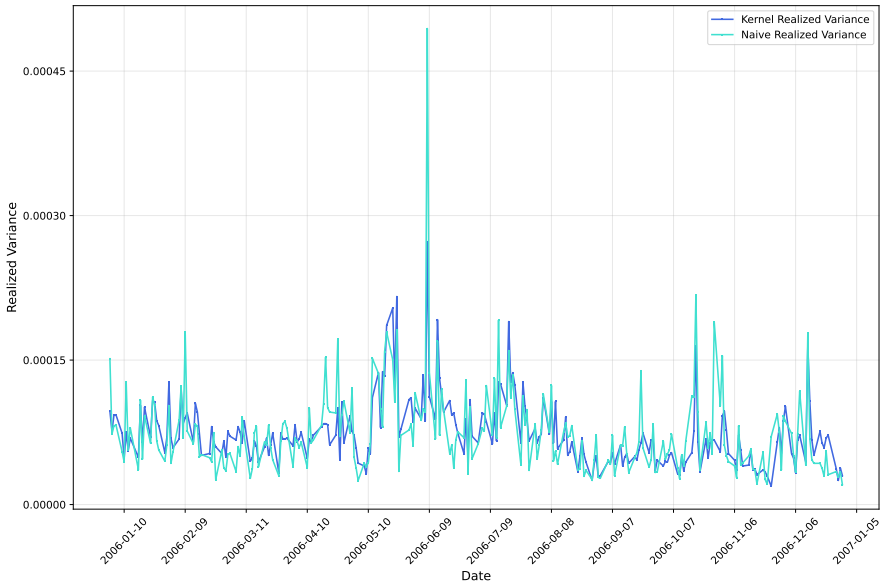

Figure 1: Comparison of kernel integrated variance and naive daily integrated variance for IBM in 2006.

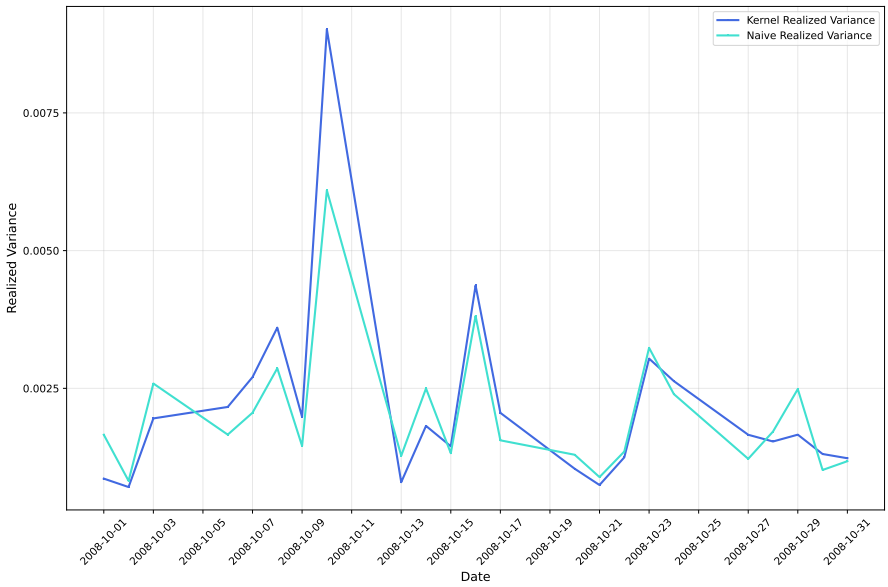

Figure 2: Comparison of kernel integrated variance and naive daily integrated variance for IBM in October 2008.

Mean	Standard deviation	Skew	Kurtosis
1.479×10^{-4}	2.659×10^{-4}	10.93	244.8

Table 1: First four moments of the kernel realized variance.

estimate by on average 2.438×10^{-7}. This is to be expected, since it better handles noise and jumps.

On the other hand we can see in Figure 2 that in periods known for their frequent and big price swings, such as the days of the stock crash in October 2008, the kernel estimate much better captures the extraordinary volatility. Over the entire month, the estimate of the kernel estimate was above the naive estimate by on average 1.357×10^{-4}. We can see that for the entire week from the 6th to the 10th the kernel systematically estimates higher integrated variance, and in particular on 10th of October. That week was called "the worst [trading] week in at least 75 years", and Friday was characterized by particularly rapid and violent swings of stock prices in both directions, sometimes within the span of just minutes (Bajaj, 2008).

Table 1 presents the first four moments of the realized variance data. Interesting is how asymmetric the distribution is given the high skew, meaning most values are concentrated below the mean, while there is a very long right tail representing the extraordinary volatility of extreme events. It's also noteworthy that the standard deviation is larger than the mean, despite RV values being (by definition) always strictly positive. The kurtosis also warrants some attention as it is about 80 times higher than that of a normal distribution.

5 Methodology

To rigorously evaluate the one-step ahead volatility forecasting performance of LTC networks relative to the RealGARCH benchmark, a specific experimental protocol was established. The analysis utilized UHFD data of the IBM stock, excluding the period prior to 2000 due to the distinctly different volatility dynamics observed then, as detailed in section 4. Models were trained with data spanning January 2000 to December 2019 inclusive. Within this training data, the final 10%, corresponding approximately to the period from July 2017 to December 2019, was designated as a validation set for hyperparameter tuning. The models' out-of-sample forecasting capabilities were subsequently assessed on the test period from January 2020 to December 2024 inclusive. This test period was deliberately chosen as it encompasses both the extreme market volatility associated with the COVID-19 crisis in 2020 and the subsequent, more typical yet still active, market conditions from 2021 to 2024.

Regarding model configuration, the benchmark RealGARCH model underwent a careful hyperparameter tuning process. Various specifications of the RealGARCH model, implemented using the `rugarch` package (Galanos, 2024) in R, were trained on the initial segment of the training data (2000 to mid-2017) and evaluated based

on their MSE loss on the validation set (mid-2017 to 2019). Key hyperparameters considered were the ARCH (p) and GARCH (q) orders, alongside the assumed distribution for the innovations. A GARCH(1,1) structure proved adequate, as higher-order models did not demonstrate improvements on the validation set. However, the choice of innovation distribution was found to be critical, with the Skewed Student's t-distribution yielding the superior validation performance. Following the identification of these optimal hyperparameters, the final RealGARCH model was re-trained using the entire 2000-2019 dataset, incorporating the validation data into the final training stage. This decision was based on the finding that models trained solely on subsets of the training data optimized for validation loss exhibited substantially weaker generalization to the test set, thus not representing the strongest possible benchmark.

Similarly, the LTC networks required configuration and training. This was done in Python thanks to the `ncps` package available following the corresponding paper on Neural Circuit Policies (see subsection 3.3) (Lechner et al., 2020). The identical validation set (mid-2017 to 2019) was employed to evaluate network performance during training and to ultimately select the optimal hyperparameters and training strategy. An extensive hyperparameter search was undertaken, covering both architectural aspects – such as the number of neurons, the window size of past observations used as input, the scale of the time delta between observations, the number of ODE solver steps per inference, input feature selection, and gradient clipping thresholds – and training dynamics, including batch sizes, learning rates, optimizers, and learning rate schedulers. A comprehensive list of the final hyperparameters is detailed in Appendix C.

Several key findings emerged from this tuning process. The Adam optimizer (Kingma & Ba, 2014) consistently outperformed simpler alternatives like Stochastic Gradient Descent, aligning with practices in the original LTC research (see references within section 3). Among the tested learning rate schedulers (including ReduceLROnPlateau, Cosine Annealing variants, and OneCycle), Cosine Annealing without warm restarts generally yielded the lowest validation losses. To account for the influence of random weight initialization, promising hyperparameter configurations were trained multiple times, confirming the consistency of observed performance differences. Furthermore, each training run incorporated an early stopping mechanism: training was halted if the validation loss did not improve for 25 consecutive epochs, and the model weights corresponding to the epoch with the minimum validation loss achieved during that run were retained.

A specific investigation focused on the model's ability to leverage the irregular timing of financial observations. Two variants were compared: one provided with the actual time elapsed (in days) since the previous observation, and another treating observations as equidistant in time (time-unaware). Interestingly, providing the raw time deltas resulted in poorer validation performance compared to the time-unaware baseline. However, when these time deltas underwent a power transformation (logarithm, square root), the resulting time-aware models consistently achieved lower validation losses than their time-unaware counterparts.

The configuration achieving the best validation performance, and thus selected

for final evaluation, employs a sparse LTC architecture with 15 neurons. It uses the 10 preceding RV observations along with the logarithmically scaled time deltas (or always the same time-delta value in case of the time-unaware model) between these observations as input to predict the subsequent RV value. To distinguish them, LTC^K is the name given to the model which is time-uninformed (constant time deltas), and LTC^V is then the model which is informed on the variable time deltas. Input RV values were scaled to the [0, 1] range using min-max normalization to facilitate network learning. Notably, experiments indicated that providing daily returns as an additional input feature, which is essential for the RealGARCH model, did not improve the predictive performance of the LTC network. The final model architecture utilizes 12 ODE solver steps for each state update. This configuration results in a model with 1249 total parameters, of which 1009 are trainable. As the low number of parameters hints at, the training time for these models are very low. Also, due to the nature of the ode-solver, they must be deployed on the CPU. In this case they were trained on a single core of a 2021 M1 Pro processor. The time-aware model trained for 250 epochs, which took about 12.5 minutes, while the time-unaware model trained for 145 epochs, which took only 7 minutes. Training and validation losses can be seen in Figure 3. The exceptional computational efficiency of the LTCs is further underscored by the fact that these rapid training times were achieved using a moderately sized training dataset of only 4402 observations. This suggests the model architecture is not only computationally lean but also data-efficient, capable of extracting relevant dynamics without requiring the vast datasets often associated with deep learning approaches.

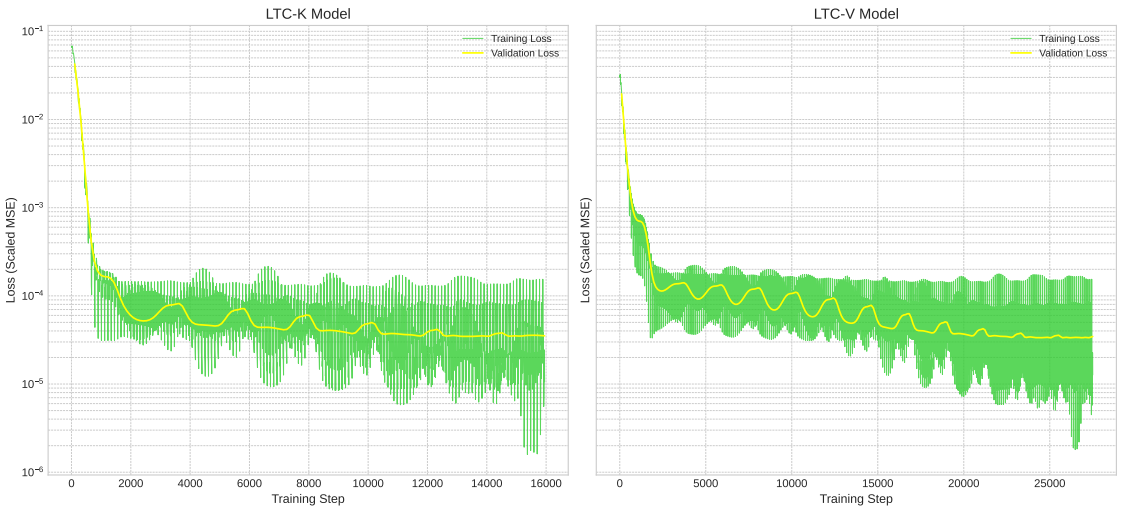

Figure 3: Training and validation losses of the LTC^K model (left) and the LTC^V model (right) as a function of the training step number.

Performance evaluation focused specifically on one-step ahead forecasts, using the Mean Squared Error (MSE) as the primary loss metric, the rationale for which is discussed in subsection 2.1. Model performance was assessed comprehensively across the entire test period (2020-2024), and also separately for the distinct sub-periods of the COVID-19 crisis year (2020) and the post-crisis years (2021-2024).

Time Period	RealGARCH	LTC^K	LTC^V
2020-2024	1.744×10^{-8}	1.619×10^{-8}	$\mathbf{1.348 \times 10^{-8}}$
2020	7.478×10^{-8}	7.176×10^{-8}	$\mathbf{5.839 \times 10^{-8}}$
2021-2024	3.095×10^{-9}	2.317×10^{-9}	$\mathbf{2.264 \times 10^{-9}}$

Table 2: Test MSE of the competing models. Lower is better, best is bold.

To determine the statistical significance of any observed differences in forecast accuracy between the competing models, the Diebold-Mariano (DM) test (Diebold & Mariano, 2002) was employed. Given two competing one-step ahead forecasts $p_{i,t}, p_{j,t}, t = 1, \ldots, T$, we derive two series of errors $e_{i,t}, e_{j,t}$ by computing the predictions' squared difference to the ground truth. The difference in loss at each moment in time is then $d_t = e_{i,t} - e_{j,t}$. The Diebold-Mariano statistic is defined as

$$DM = \frac{\overline{d}}{\sqrt{\hat{V}(\overline{d})}} \tag{5.1}$$

where $\overline{d} = \frac{1}{T}\sum_{t=1}^{T} d_t$ is the sample mean of the loss differential and $\hat{V}(\overline{d})$ is a consistent estimate of the variance of $\overline{d}$. For one-step ahead forecasts, the variance estimator is typically

$$\hat{V}(\overline{d}) = \frac{1}{T}\hat{\gamma}_0 \tag{5.2}$$

where $\hat{\gamma}_0$ is the sample variance of the loss differential. Under the null hypothesis of equal predictive accuracy, the DM statistic follows a standard normal distribution asymptotically.

6 Results

The comparative predictive accuracy of the LTC network and the RealGARCH model on the 2020-2024 test set was assessed using the Mean Squared Error (MSE) loss metric, with a one-sided Diebold-Mariano test (Diebold & Mariano, 2002) used to evaluate the statistical significance of performance differences. The models evaluated are RealGARCH as a benchmark, LTC^K the model which is time-uninformed (constant time deltas), and LTC^V the model which is informed on the variable time deltas. In Table 2 we can see that both models are capable of achieving a lower loss than RealGARCH, and that the time-aware version, LTC^V, outperformed the others both in the crisis year and in the following years. In Table 3 we can see the Diebold-Mariano statistic along with the corresponding p-value for all the model comparisons in all three periods. From that we can infer confidently that the LTC models outperform the RealGARCH model by a significant margin in the post-crisis period, while being only somewhat better during the crisis period, with none of the p-values from the crisis period being significant. This difficulty in the crisis period is also what affects the global performance comparison between the models, as the

Model Comparison	Time Period		
	2020-2024	2020	2021-2024
LTC^K vs RealGARCH			
DM Statistic	-0.274	-0.137	**-4.356**
p-value	0.392	0.445	$\mathbf{6.607 \times 10^{-6}}$
LTC^V vs RealGARCH			
DM Statistic	-1.248	-1.039	**-5.478**
p-value	0.106	0.149	$\mathbf{2.154 \times 10^{-8}}$
LTC^V vs LTC^K			
DM Statistic	-1.606	-1.585	-0.899
p-value	0.054	0.056	0.184

Table 3: One sided Diebold-Mariano test results for model comparisons across different time periods. The test checks whether one model has a significantly lower loss, with null hypothesis being that the forecasting accuracy of the two models is equal. Statistically significant values at a 5% confidence level are bold.

p-values in 2021-2024 are extraordinarily low, presenting very strong evidence of the superior predictive ability of the LTCs.

Figure 4 shows the prediction of the three models compared to the ground truth that is the kernel based IV estimate. The most notable features of the comparison are that all models tend to underestimate the peaks of volatility. RealGARCH shows larger magnitude forecasts during peaks in the crisis period, while the LTC^K model appears to track peaks more closely in the post-crisis years. In general, RealGARCH has a tendency to overestimate the volatility levels, as it seems that it models too slow of a decay after a local peak. The LTCs seem able to follow the decay much more closely, and in a few instances even underestimate the realized variance.

We can also get an idea of the additional performance the LTC architecture gains by having access to the timing at which observations arrive. While none of p-values of the comparison of LTC^V and LTC^K are significant, they are fairly close to 0.05 value, which, while not being clear evidence of better performance, at least suggests that the model is capable of learning some dynamics that adapt to the timing at which observations arrive. This is also supported by the consistently lower validation loss that the LTC^V version of models achieved over the LTC^K.

7 Discussion

What is most interesting about the results is that when the LTC was fed the time information with linear scaling, meaning as a simple multiple of time delta in days, it achieved a consistently *worse* validation loss than the time-unaware model, and it also didn't outperform it on the test data. The superior performance was only

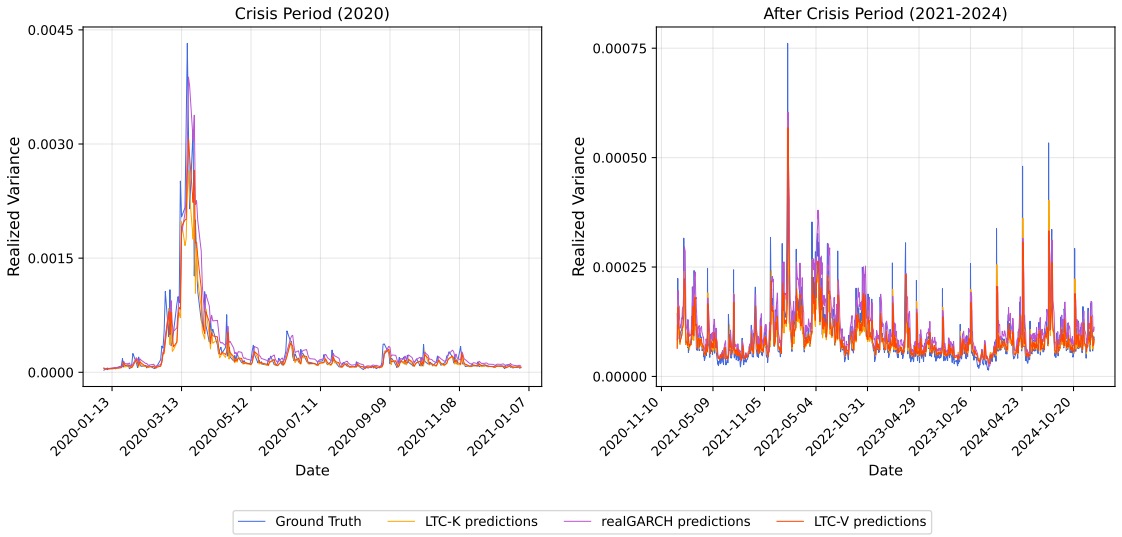

Figure 4: Comparison of models in the crisis and in the after crisis years.

achieved when a power transformation (in particular the logarithm) was applied to those time deltas. The reason as to why this is the case is up for speculation. One might imagine that, while the number of days between observations do in fact influence how past volatility predicts future volatility, there are some diminishing returns to this effect. For example, the market action on a Friday before a regular weekend might be less predictive of the volatility on Monday, than the market action on Thursday was for the volatility on Friday. However, such a Friday volatility might be almost as predictive of the Monday's volatility, as the volatility of a Wednesday before a 4-day-long holiday. The power transformation has the effect of creating diminishing increases in the number of time steps taken by the ode-solver as the number of days or market close increase, aiding the model in not valuing all trading breaks equally. To put it simply, it seems that the market's "memory" of previous volatility decays non-linearly with time, or that new information gets accumulated non-linearly with time, or a combination of the two.

Noteworthy is also the tendency of the RealGARCH model to overestimate volatility, due to underestimating its decay after local peaks. This persistence, while designed to capture volatility clustering, appears to somewhat impede the model's adaptability to rapidly changing market conditions. Conversely, the LTCs showed more responsive behavior, following decay patterns more closely and occasionally even underestimating realized volatility. This suggests that the continuous-time nature of LTC networks enables them to better capture the mean-reverting properties of volatility processes, particularly during periods of decreasing volatility.

The differential performance of models between crisis and post-crisis periods provides valuable insights into their adaptive capabilities. During the highly volatile COVID-19 crisis of 2020, all models struggled to fully capture volatility peaks, with no statistically significant difference between approaches. While the LTC networks achieved an overall lower loss, it seems that RealGARCH better captured the peak volatility of the crisis.

In contrast, the post-crisis period (2021-2024) revealed clear superiority of both LTC variants over the RealGARCH benchmark, with extraordinarily low p-values from the Diebold-Mariano tests. This indicates that LTC networks excel at capturing the nuanced dynamics of volatility during more typical market conditions, where they can leverage their ability to maintain and update internal states in response to evolving market patterns.

8 Conclusion, Scope, and Limitations

This study has evaluated the efficacy of Liquid Time-Constant Neural Networks for one-step ahead daily volatility forecasting, comparing their performance against the established RealGARCH benchmark. The models were evaluated against and trained on a state of the art realized volatility estimate from high frequency data that combines pre-averaging with kernel smoothing.

The results show that LTCs can significantly outperform traditional econometric models in volatility forecasting, particularly during periods of normal market activity. The exceptionally low p-values obtained from Diebold-Mariano tests for the post-crisis period provide strong statistical evidence for this superiority. While these neural networks achieved also a smaller loss during the COVID-19 crisis, the differences were not statistically significant.

This work also highlights the potential value of incorporating temporal information in volatility models. The consistently superior performance of time-aware LTC networks suggests that accounting for the irregular timing of financial observations can enhance predictive accuracy. Crucially, the findings suggest a non-linear relationship between market closure duration and its impact on future volatility. Specifically, the model only outperformed the time-unaware variant under the assumption that, while extended market closures do change the volatility dynamic, this influence follows a concave pattern where each additional day of closure has progressively less marginal impact.

Despite the promising results, several important limitations of this study should be acknowledged. First, the analysis is univariate and focusing solely on the volatility dynamics of a single asset (IBM stock). Second, the evaluation is restricted to one-step-ahead forecasting. While this is a common framework for assessing volatility models, many practical applications—such as risk management and derivatives pricing—require accurate multi-step forecasts. Third, the study period, while including both crisis and post-crisis conditions, represents a specific macroeconomic environment. The generalizability of the findings to other market regimes, such as prolonged bear markets or inflationary environments, remains an open question.

In summary, LTCs represent a promising approach to volatility forecasting that combines the representational power of neural networks with an intrinsic ability to handle irregular time series, offering both superior predictive performance compared to an established econometric model, and higher computational efficiency compared to regular RNNs.

9 Declaration of Authorship

I hereby declare,

- that I have written this thesis independently,
- that I have written the thesis using only the aids specified in the index;
- that all parts of the thesis produced with the help of aids have been declared;
- that I have handled both input and output responsibly when using AI. I confirm that I have therefore only read in public data or data released with consent and that I have checked, declared and comprehensibly referenced all results and/or other forms of AI assistance in the required form and that I am aware that I am responsible if incorrect content, violations of data protection law, copyright law or scientific misconduct (e.g. plagiarism) have also occurred unintentionally;
- that I have mentioned all sources used and cited them correctly according to established academic citation rules;
- that I have acquired all immaterial rights to any materials I may have used, such as images or graphics, or that these materials were created by me;
- that the topic, the thesis or parts of it have not already been the object of any work or examination of another course, unless this has been expressly agreed with the faculty member in advance and is stated as such in the thesis;
- that I am aware of the legal provisions regarding the publication and dissemination of parts or the entire thesis and that I comply with them accordingly;
- that I am aware that my thesis can be electronically checked for plagiarism and for third-party authorship of human or technical origin and that I hereby grant the University of St.Gallen the copyright according to the Examination Regulations as far as it is necessary for the administrative actions;
- that I am aware that the University will prosecute a violation of this Declaration of Authorship and that disciplinary as well as criminal consequences may result, which may lead to expulsion from the University or to the withdrawal of my title.

9.1 Directory of Aids

Spellchecking software was used all throughout the writing process in the form of a language server for latex in the Neovim text editor. Additionally, AI chatbots, in particular Claude (Anthropic) and Gemini (Google) were used extensively for proofreading and rephrasing already written text. This was done to improve the phrasing and style of the content, in order to make it more palatable and elegant.

References

Akgiray, V. (1989). Conditional heteroscedasticity in time series of stock returns: Evidence and forecasts. *Journal of business*, 55–80.

Andersen, T. G., & Bollerslev, T. (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International economic review*, 885–905.

Andersen, T. G., Bollerslev, T., Christoffersen, P., & Diebold, F. X. (2005). *Volatility forecasting.* National Bureau of Economic Research Cambridge, Mass., USA.

Andersen, T. G., Bollerslev, T., Christoffersen, P. F., & Diebold, F. X. (2006). Volatility and correlation forecasting. *Handbook of economic forecasting*, *1*, 777–878.

Andersen, T. G., Bollerslev, T., & Diebold, F. X. (2007). Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility. *The review of economics and statistics*, *89*(4), 701–720.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Ebens, H. (2001). The distribution of realized stock return volatility. *Journal of financial economics*, *61*(1), 43–76.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Labys, P. (1999). *The distribution of exchange rate volatility* (Vol. 6961). National Bureau of Economic Research.

Ap Gwilym, O., & Sutcliffe, C. (2001). Problems encountered when using high frequency financial market data: Suggested solutions. *Journal of Financial Management & Analysis*, *14*(1), 38.

Ardiel, E. L., & Rankin, C. H. (2010). An elegant mind: learning and memory in caenorhabditis elegans. *Learning & memory*, *17*(4), 191–201.

Bajaj, V. (2008, October). Whiplash Ends a Roller Coaster Week. *N.Y. Times*. Retrieved from `https://www.nytimes.com/2008/10/11/business/11markets.html`

Barndorff-Nielsen, O. E., & Shephard, N. (2004). Power and bipower variation with stochastic volatility and jumps. *Journal of financial econometrics*, *2*(1), 1–37.

Barnett, V., Lewis, T., et al. (1994). *Outliers in statistical data* (Vol. 3) (No. 1). Wiley New York.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, *9*(1), 69–107.

Bauwens, L., Hafner, C., & Laurent, S. (2012). *Handbook of Volatility Models and Their Applications*. doi: 10.1002/9781118272039

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, *31*(3), 307–327.

Bollerslev, T., & Ghysels, E. (1996). Periodic autoregressive conditional heteroscedasticity. *Journal of Business & Economic Statistics*, *14*(2), 139–151.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Brailsford, T. J., & Faff, R. W. (1996). An evaluation of volatility forecasting techniques. *Journal of Banking & Finance*, *20*(3), 419–438.

Breakspear, M. (2017). Dynamic models of large-scale brain activity. *Nature neuroscience*, *20*(3), 340–352.

Brownlees, C. T., & Gallo, G. M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational statistics & data analysis*, *51*(4), 2232–2245.

Chalfie, M., Sulston, J. E., White, J. G., Southgate, E., Thomson, J. N., & Brenner, S. (1985). The neural circuit for touch sensitivity in caenorhabditis elegans. *Journal of Neuroscience*, *5*(4), 956–964.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, *8*(1), 6085.

Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, *31*.

Cho, K. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., & Sun, J. (2016). Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference* (pp. 301–318).

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, *7*(2), 174–196.

Corsi, F., Audrino, F., & Renó, R. (2012). Har modeling for realized volatility forecasting.

Davis, R. E., & Stretton, A. (1989). Signaling properties of ascaris motorneurons: graded active responses, graded synaptic transmission, and tonic transmitter release. *Journal of Neuroscience*, *9*(2), 415–425.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), 134–144.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., & Song, L. (2016). Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1555–1564).

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the econometric society*, 987–1007.

Epanechnikov, V. A. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, *14*(1), 153–158.

Falkenberry, T. (2002). *High frequency data filtering* (Technical Report). Tick Data.

Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, *38*(1), 34–105.

Fan, J., & Wang, Y. (2008). Spot volatility estimation for high-frequency data. *Statistics and its Interface*, *1*(2), 279–288.

Figueroa-López, J. E., & Li, C. (2020). Optimal kernel estimation of spot volatility of stochastic differential equations. *Stochastic Processes and their Applications*, *130*(8), 4693–4720.

Figueroa-López, J. E., & Wu, B. (2024). Kernel estimation of spot volatility with microstructure noise using pre-averaging. *Econometric Theory*, *40*(3), 558–607.

Foster, D. P., & Nelson, D. B. (1994). *Continuous record asymptotics for rolling sample variance estimators.* National Bureau of Economic Research Cambridge, Mass., USA.

Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Friston, K. J., Harrison, L., & Penny, W. (2003). Dynamic causal modelling. *Neuroimage*, *19*(4), 1273–1302.

Funahashi, K.-i., & Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, *6*(6), 801–806.

Galanos, A. (2024). rugarch: Univariate garch models. [Computer software manual]. (R package version 1.5-3.)

Gerety, M. S., & Mulherin, J. H. (1992). Trading halts and market activity: An analysis of volume at the open and the close. *The Journal of Finance*, *47*(5), 1765–1784.

Gholami, A., Keutzer, K., & Biros, G. (2019). Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*.

Gleeson, P., Lung, D., Grosu, R., Hasani, R., & Larson, S. D. (2018). c302: a multiscale framework for modelling the nervous system of caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *373*(1758), 20170379.

Goodhart, C. A. E., & Figliuoli, L. (1991). Every minute counts in financial markets. *Journal of International Money and Finance*, *10*(1), 23–52.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 ieee international conference on acoustics, speech and signal processing* (pp. 6645–6649).

Gray, J. M., Hill, J. J., & Bargmann, C. I. (2005). A circuit for navigation in caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, *102*(9), 3184–3191.

Hamilton, J. D., & Susmel, R. (1994). Autoregressive conditional heteroskedasticity and changes in regime. *Journal of econometrics*, *64*(1-2), 307–333.

Hansen, P. R., Huang, Z., & Shek, H. H. (2012). Realized garch: a joint model for returns and realized measures of volatility. *Journal of Applied Econometrics*, *27*(6), 877–906.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: does anything beat a garch (1, 1)? *Journal of applied econometrics*, *20*(7), 873–889.

Hansen, P. R., & Lunde, A. (2006). Consistent ranking of volatility models. *Journal of Econometrics*, *131*(1-2), 97–121.

Hasani, R. (2020). *Interpretable recurrent neural networks in continuous-time control environments* (Unpublished doctoral dissertation). Technische Universität Wien.

Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., … Rus, D. (2022). Closed-form continuous-time neural networks. *Nature Machine Intelligence*, *4*(11), 992–1003.

Hasani, R., Lechner, M., Amini, A., Rus, D., & Grosu, R. (2020). A natural lottery ticket winner: Reinforcement learning with ordinary neural circuits. In *international conference on machine learning* (pp. 4082–4093).

Hasani, R., Lechner, M., Amini, A., Rus, D., & Grosu, R. (2021, May). Liquid time-constant networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(9), 7657-7666. doi: 10.1609/aaai.v35i9.16936

Hautsch, N., & Podolskij, M. (2013). Preaveraging-based estimation of quadratic variation in the presence of noise and jumps: theory, implementation, and empirical evidence. *Journal of Business & Economic Statistics*, *31*(2), 165–183.

Hellerstein, J. M. (2008). *Quantitative data cleaning for large databases* (Report). Berkley, CA: United Nations Economic Commission for Europe.

Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, *81*(10), 3088–3092.

Hubert, M. (2004). *Theory and applications of recent robust methods*. Springer Science & Business Media.

Jacod, J., Li, Y., Mykland, P. A., Podolskij, M., & Vetter, M. (2009). Microstructure noise in the continuous case: the pre-averaging approach. *Stochastic processes and their applications*, *119*(7), 2249–2276.

Jacod, J., Podolskij, M., & Vetter, M. (2010). Limit theorems for moving averages of discretized processes plus noise.

Jacod, J., & Protter, P. (2011). *Discretization of processes* (Vol. 67). Springer Science & Business Media.

Jacod, J., & Todorov, V. (2014). Efficient estimation of integrated volatility in presence of infinite variation jumps.

Janowsky, S. A. (1989). Pruning versus clipping in neural networks. *Physical Review A*, *39*(12), 6600.

Jorion, P. (1988). On jump processes in the foreign exchange and stock markets. *The Review of Financial Studies*, *1*(4), 427–445.

Kabanov, Y., Liptser, R., Stoyanov, J., Barndorff-Nielsen, O. E., Graversen, S. E., Jacod, J., … Shephard, N. (2006). *A central limit theorem for realised power and bipower variations of continuous semimartingales*. Springer.

Kato, S., Kaplan, H. S., Schrödel, T., Skora, S., Lindsay, T. H., Yemini, E., … Zimmer, M. (2015). Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, *163*(3), 656–669.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koch, C., & Segev, I. (1998). *Methods in neuronal modeling: from ions to networks*. MIT press.

Kristensen, D. (2010). Nonparametric filtering of the realized spot volatility: A kernel-based approach. *Econometric Theory*, *26*(1), 60–93.

Kuen, T. Y., & Hoong, T. S. (1992). Forecasting volatility in the singapore stock market. *Asia Pacific Journal of Management*, *9*, 1–13.

Lamoureux, C. G., & Lastrapes, W. D. (1993). Forecasting stock-return variance: Toward an understanding of stochastic implied volatilities. *The Review of Financial Studies*, *6*(2), 293–326.

Lechner, M., Hasani, R., Amini, A., Henzinger, T. A., Rus, D., & Grosu, R. (2020). Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, *2*(10), 642–652.

Lechner, M., Hasani, R., Zimmer, M., Henzinger, T. A., & Grosu, R. (2019). Designing worm-inspired neural networks for interpretable robotic control. In *2019 international conference on robotics and automation (icra)* (pp. 87–94).

Lechner, M., & Hasani, R. M. (2020). Learning long-term dependencies in irregularly-sampled time series. *CoRR*, *abs/2006.04418*. Retrieved from `https://arxiv.org/abs/2006.04418`

LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. *Advances in neural information processing systems*, *2*.

Lipton, Z. C., Kale, D., & Wetzel, R. (2016). Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine learning for healthcare conference* (pp. 253–270).

Liu, S., & Wang, Z. (2023). Ten lessons we have learned in the new" sparseland": A short handbook for sparse neural network researchers. *arXiv preprint arXiv:2302.02596*.

Lockery, S., & Sejnowski, T. (1992). Distributed processing of sensory information in the leech. iii. a dynamical neural network model of the local bending reflex. *Journal of Neuroscience*, *12*(10), 3877–3895.

Lynch, P. E., & Zumbach, G. O. (2003). Market heterogeneities and the causal structure ofvolatility. *Quantitative Finance*, *3*(4), 320.

Mancini, C., Mattiussi, V., & Renò, R. (2015). Spot volatility estimation using delta sequences. *Finance and Stochastics*, *19*, 261–293.

Mandelbrot, B. (1963). New methods in statistical economics. *Journal of political economy*, *71*(5), 421–440.

Masset, P. (2011). Volatility stylized facts. *Available at SSRN 1804070*.

Mozer, M. C., & Smolensky, P. (1989). Using relevance to reduce network size automatically. *Connection Science*, *1*(1), 3–16.

Müller, H.-G. (1991). Smooth optimum kernel estimators near endpoints. *Biometrika*, *78*(3), 521–530.

Müller, U. A., Dacorogna, M. M., Davé, R. D., Olsen, R. B., Pictet, O. V., & Von Weizsäcker, J. E. (1997). Volatilities of different time resolutions—analyzing the dynamics of market components. *Journal of Empirical Finance*, *4*(2-3), 213–239.

Müller, U. A., Dacorogna, M. M., Davé, R. D., Pictet, O. V., Olsen, R. B., & Ward, J. R. (1993, October). Fractals and intrinsic time - a challenge to

econometricians. In *Xxxixth international aea conference on real time econometrics.* Luxembourg. (Also presented at 4th International PASE Workshop, Ascona, Switzerland, November 22-26, 1993. Published in: Lüthje, B. (Ed.), Erfolgreiche Zinsprognose. Verband öffentlicher Banken, Bonn 1994. Technical Report UAM.1993-08-16, Olsen&Associates, Seefeldstrasse 233, 8008 Zürich, Switzerland)

Patton, A. J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, *160*(1), 246–256.

Penny, W., Ghahramani, Z., & Friston, K. (2005). Bilinear dynamical systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *360*(1457), 983–993.

Podolskij, M., & Vetter, M. (2009). Estimation of volatility functionals in the simultaneous presence of microstructure noise and jumps.

Pontryagin, L. S. (2018). *Mathematical theory of optimal processes.* Routledge.

Press, W. H. (2007). *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press.

Rubanova, Y., Chen, R. T., & Duvenaud, D. K. (2019). Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, *32*.

Runge, C. (1895). Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, *46*(2), 167–178.

Stephens, G. J., Johnson-Kerner, B., Bialek, W., & Ryu, W. S. (2008). Dimensionality and dynamics in the behavior of c. elegans. *PLoS computational biology*, *4*(4), e1000028.

Sutskever, I. (2014). Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Szigeti, B., Gleeson, P., Vella, M., Khayrulin, S., Palyanov, A., Hokanson, J., … Larson, S. (2014). Openworm: an open-science approach to modeling caenorhabditis elegans. *Frontiers in computational neuroscience*, *8*, 137.

Tse, Y. K. (1991). Stock returns volatility in the tokyo stock exchange. *Japan and the World Economy*, *3*(3), 285–298.

Verousis, T., & Ap Gwilym, O. (2010). An improved algorithm for cleaning ultra high-frequency data. *Journal of Derivatives & Hedge Funds*, *15*, 323–340.

VIX Volatility Index - Historical Chart. (2025, April). Retrieved from `https://www.macrotrends.net/2603/vix-volatility-index-historical-chart` ([Online; accessed 3. Apr. 2025])

Vorbach, C., Hasani, R., Amini, A., Lechner, M., & Rus, D. (2021). Causal navigation by continuous-time neural networks. *Advances in Neural Information Processing Systems*, *34*, 12425–12440.

Whaley, R. E. (2009). Understanding the vix. *Journal of Portfolio Management*, *35*(3), 98–105.

Wicks, S. R., Roehrig, C. J., & Rankin, C. H. (1996). A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, *16*(12), 4017–4031.

Yu, C., Fang, Y., Li, Z., Zhang, B., & Zhao, X. (2014). Non-parametric estimation

of high-frequency spot volatility for brownian semimartingale with jumps. *Journal of Time Series Analysis*, *35*(6), 572–591.

Zhang, H., Wang, Z., & Liu, D. (2014). A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(7), 1229–1262.

Zhang, L., Mykland, P. A., & Aït-Sahalia, Y. (2005). A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association*, *100*(472), 1394–1411.

Zhou, B. (1996). High-frequency data and volatility in foreign-exchange rates. *Journal of Business & Economic Statistics*, *14*(1), 45–52.

Zu, Y., & Boswijk, H. P. (2014). Estimating spot volatility with high-frequency financial data. *Journal of Econometrics*, *181*(2), 117–135.

A Derivation of Constants for Optimizations

The three constants $\Phi_{11}, \Phi_{22}, \Phi_{12}$ are in general defined as

$$\Phi_{ij} = \int_0^1 \phi_i(s)\phi_j(s)\, ds$$

where $\phi_1(s) = \int_s^1 g'(u)g'(u-s)\, du$, and $\phi_2(s) = \int_s^1 g(u)g(u-s)\, du$. Recall that the function of interest $g(x)$ is

$$\begin{aligned} g(x) &= 2x \wedge 1 - x \\ &= \min(2x, 1-x) \\ &= \begin{cases} 2x & \text{if } x \leq \frac{1}{3} \\ 1-x & \text{otherwise} \end{cases} \end{aligned}$$

and its derivative is

$$g'(x) = \begin{cases} 2 & \text{if } x \leq \frac{1}{3} \\ -1 & \text{otherwise} \end{cases}$$

A.1 First Constant $\Phi_{1,1}$

The integral for $\Phi_{1,1}$ is

$$\Phi_{1,1} = \int_0^1 \phi_1(s) \cdot \phi_1(s)\, ds = \int_0^1 \left(\int_s^1 g'(u)g'(u-s)\, du \right)^2 ds$$

The integral has to be split up into multiple parts which are too long to write at the same time. When $s \in \left[0, \frac{1}{3}\right]$, $\phi_1(s)$ becomes

$$\begin{aligned}
\phi_1(s) &= \int_s^1 g'(u)g'(u-s)\,du \\
&= \int_s^{1/3} g'(u)g'(u-s)\,du + \int_{1/3}^1 g'(u)g'(u-s)\,du \\
&= 4\left(\frac{1}{3} - s\right) - \int_{1/3}^{s+1/3} g'(u-s)\,du - \int_{s+1/3}^1 g'(u-s)\,du \\
&= \frac{4}{3} - 4s - 2\left(s + \frac{1}{3} - \frac{1}{3}\right) + \left(1 - s - \frac{1}{3}\right) \\
&= \frac{4}{3} - 4s - 2s + \frac{2}{3} - s \\
&= 2 - 7s
\end{aligned}$$

When $s \in \left[\frac{1}{3}, \frac{2}{3}\right]$, $\phi_1(s)$ becomes

$$\begin{aligned}
\phi_1(s) &= \int_s^1 g'(u)g'(u-s)\,du \\
&= -\int_s^{s+1/3} g'(u-s)\,du - \int_{s+1/3}^1 g'(u-s)\,du \\
&= -2\left(s + \frac{1}{3} - s\right) + \left(1 - s - \frac{1}{3}\right) \\
&= -\frac{2}{3} + \frac{2}{3} - s \\
&= -s
\end{aligned}$$

Finally, when $s \in \left[\frac{2}{3}, 1\right]$, $\phi_1(s)$ becomes

$$\phi_1(s) = \int_s^1 g'(u)g'(u-s)\,du = -2(1-s) = 2s - 2$$

When we put it all together

$$\begin{aligned}
\Phi_{1,1} &= \int_0^1 \phi_1(s)^2\,ds \\
&= \int_0^{1/3} (2-7s)^2\,ds + \int_{1/3}^{2/3} (-s)^2\,ds + \int_{2/3}^1 (2s-2)^2\,ds \\
&= \int_0^{1/3} (4 - 28s + 49s^2)\,ds + \int_{1/3}^{2/3} s^2\,ds + \int_{2/3}^1 (4s^2 - 8s + 4)\,ds \\
&= \left[4s - 14s^2 + \frac{49}{3}s^3\right]_0^{\frac{1}{3}} + \left[\frac{1}{3}s^3\right]_{\frac{1}{3}}^{\frac{2}{3}} + \left[\frac{4}{3}s^3 - 4s^2 + 4s\right]_{\frac{2}{3}}^1 \\
&= \frac{31}{81} + \frac{7}{81} + \frac{4}{81} \\
&= \frac{14}{27}
\end{aligned}$$

A.2 Second Constant $\Phi_{2,2}$

The integral for $\Phi_{2,2}$ is

$$\Phi_{2,2} = \int_0^1 \phi_2(s) \cdot \phi_2(s)\, ds = \int_0^1 \left(\int_s^1 g(u)g(u-s)\, du \right)^2 ds$$

This integral too needs to be split up into multiple parts that cannot be written out at the same time. The same strategy to solve the integral in subsection A.1 is used. When $s \in \left[0, \frac{1}{3}\right]$, $\phi_2(s)$ becomes

$$\begin{aligned}
\phi_2(s) &= \int_s^1 g(u)g(u-s)\, du \\
&= \int_s^{1/3} g(u)g(u-s)\, du + \int_{1/3}^{s+1/3} g(u)g(u-s)\, du + \int_{s+1/3}^1 g(u)g(u-s)\, du \\
&= \int_s^{1/3} 4u(u-s)\, du + \\
&\quad \int_{1/3}^{s+1/3} 2(u-s)(1-u)\, du + \\
&\quad \int_{s+1/3}^1 (1-u)(1-u+s)\, du \\
&= \int_s^{1/3} 4u^2 - 4su\, du + \\
&\quad \int_{1/3}^{s+1/3} 2u - 2s - 2u^2 + 2su)\, du + \\
&\quad \int_{s+1/3}^1 1 - 2u + s + u^2 - su\, du \\
&= \left[\frac{4}{3}u^3 - 2su^2 \right]_s^{\frac{1}{3}} + \\
&\quad \left[u^2 - 2su - \frac{2}{3}u^3 + su^2 \right]_{\frac{1}{3}}^{s+\frac{1}{3}} + \\
&\quad \left[u - u^2 + su + \frac{1}{3}u^3 - \frac{1}{2}su^2 \right]_{s+\frac{1}{3}}^1 \\
&= \frac{4}{81} - \frac{2}{9}s + \frac{2}{3}s^3 + \frac{1}{3}s^3 - s^2 + \frac{4}{9}s + \frac{1}{6}s^3 - \frac{2}{9}s + \frac{8}{81} \\
&= \frac{7}{6}s^3 - s^2 + \frac{4}{27}
\end{aligned}$$

When $s \in \left[\frac{1}{3}, \frac{2}{3}\right]$, $\phi_2(s)$ becomes

$$\begin{aligned}
\phi_2(s) &= \int_s^1 g(u)g(u-s)\,du \\
&= \int_s^{s+1/3} g(u)g(u-s)\,du + \int_{s+1/3}^1 g(u)g(u-s)\,du \\
&= \int_s^{s+1/3} 2(u-s)(1-u)\,du + \int_{s+1/3}^1 (1-u)(1-u-s)\,du \\
&= \int_s^{s+1/3} 2u - 2s - 2u^2 + 2su\,du + \int_{s+1/3}^1 1 - 2u - s + u^2 + su\,du \\
&= \left[u^2 - 2su - \frac{2}{3}u^3 + su^2\right]_s^{s+\frac{1}{3}} + \left[u - u^2 - su + \frac{1}{3}u^3 + \frac{1}{2}su^2\right]_{s+\frac{1}{3}}^1 \\
&= -\frac{1}{9}s + \frac{7}{81} - \frac{5}{6}s^3 + \frac{4}{3}s^2 - \frac{2}{3}s + \frac{8}{81} \\
&= -\frac{5}{6}s^3 + \frac{4}{3}s^2 - \frac{7}{9}s + \frac{5}{27}
\end{aligned}$$

When $s \in \left[\frac{2}{3}, 1\right]$, $\phi_2(s)$ becomes

$$\begin{aligned}
\phi_2(s) &= \int_s^1 g(u)g(u-s)\,du \\
&= \int_s^1 2(1-u)(u-s)\,du \\
&= \int_s^1 2u - 2s - 2u^2 + 2su\,du \\
&= \left[u^2 - 2su - \frac{2}{3}u^3 + su^2\right]_s^1 \\
&= 1 - 2s - \frac{2}{3} + s - s^2 + 2s^2 + \frac{2}{3}s^3 - s^3 \\
&= -\frac{1}{3}s^3 + s^2 - s + \frac{1}{3}
\end{aligned}$$

When we put it all together

$$\begin{aligned}
\Phi_{2,2} &= \int_0^1 \phi_2(s)^2\, ds \\
&= \int_0^{1/3} \left(\frac{7}{6}s^3 - s^2 + \frac{4}{27}\right)^2 ds + \\
&\quad \int_{1/3}^{2/3} \left(-\frac{5}{6}s^3 + \frac{4}{3}s^2 - \frac{7}{9}s + \frac{5}{27}\right)^2 ds + \\
&\quad \int_{2/3}^{1} \left(-\frac{1}{3}s^3 + s^2 - s + \frac{1}{3}\right)^2 ds \\
&= \int_0^{1/3} \left(\frac{49}{36}s^6 - \frac{7}{3}s^5 + s^4 + \frac{28}{81}s^3 - \frac{8}{27}s^2 + \frac{16}{729}\right) ds + \\
&\quad \int_{1/3}^{2/3} \left(\frac{25}{36}s^6 - \frac{20}{9}s^5 + \frac{83}{27}s^4 - \frac{193}{81}s^3 + \frac{89}{81}s^2 - \frac{70}{243}s + \frac{25}{729}\right) ds + \\
&\quad \int_{2/3}^{1} \left(\frac{1}{9}s^6 - \frac{2}{3}s^5 + \frac{5}{3}s^4 - \frac{20}{9}s^3 + \frac{5}{3}s^2 - \frac{2}{3}s + \frac{1}{9}\right) ds \\
&= \left[\frac{49}{252}s^7 - \frac{7}{18}s^6 + \frac{28}{324}s^4 + \frac{1}{5}s^5 - \frac{8}{81}s^3 + \frac{16}{729}s\right]_0^{1/3} + \\
&\quad \left[\frac{25}{252}s^7 - \frac{10}{27}s^6 + \frac{83}{135}s^5 - \frac{193}{324}s^4 + \frac{89}{243}s^3 - \frac{35}{243}s^2 + \frac{25}{729}s\right]_{1/3}^{2/3} + \\
&\quad \left[\frac{1}{63}s^7 - \frac{1}{9}s^6 + \frac{1}{3}s^5 - \frac{5}{9}s^4 + \frac{5}{9}s^3 - \frac{1}{3}s^2 + \frac{1}{9}s\right]_{2/3}^{1} \\
&= \frac{2009}{393660} + \frac{173}{688905} + \frac{1}{137781} \\
&= \frac{985}{183708}
\end{aligned}$$

A.3 Third Constant $\Phi_{1,2}$

The integral for $\Phi_{1,2}$ is

$$\Phi_{1,2} = \int_0^1 \phi_1(s) \cdot \phi_2(s)\, ds = \int_0^1 \int_s^1 g'(u)g'(u-s)\, du \int_s^1 g(u)g(u-s)\, du\, ds$$

the form of the inner integral just follows those seen before with $\Phi_{1,1}$ and $\Phi_{2,2}$.

$$
\begin{aligned}
\Phi_{1,2} &= \int_0^1 \phi_1(s) \cdot \phi_2(s)\, ds \\
&= \int_0^{1/3} (2-7s) \cdot \left(\frac{7}{6}s^3 - s^2 + \frac{4}{27}\right) ds + \\
&\quad \int_{1/3}^{2/3} -s \cdot \left(-\frac{5}{6}s^3 + \frac{4}{3}s^2 - \frac{7}{9}s + \frac{5}{27}\right) ds + \\
&\quad \int_{2/3}^{1} (2s-2) \cdot \left(-\frac{1}{3}s^3 + s^2 - s + \frac{1}{3}\right) ds \\
&= \int_0^{1/3} \left(-\frac{49}{6}s^4 + \frac{28}{3}s^3 - 2s^2 - \frac{28}{27}s + \frac{8}{27}\right) ds + \\
&\quad \int_{1/3}^{2/3} \left(\frac{5}{6}s^4 - \frac{4}{3}s^3 + \frac{7}{9}s^2 - \frac{5}{27}s\right) ds + \\
&\quad \int_{2/3}^{1} \left(-\frac{2}{3}s^4 + \frac{8}{3}s^3 - 4s^2 + \frac{8}{3}s - \frac{2}{3}\right) ds \\
&= \left[-\frac{49}{30}s^5 + \frac{7}{3}s^4 - \frac{2}{3}s^3 - \frac{14}{27}s^2 + \frac{8}{27}s\right]_0^{\frac{1}{3}} + \\
&\quad \frac{1}{54}\left[9s^5 - 18s^4 + 14s^3 - 5s^2\right]_{\frac{1}{3}}^{\frac{2}{3}} + \\
&\quad \left[-\frac{2}{15}s^5 + \frac{2}{3}s^4 - \frac{4}{3}s^3 + \frac{4}{3}s^2 - \frac{2}{3}s + \frac{2}{15}\right]_{\frac{2}{3}}^{1} + \\
&= \frac{281}{7290} - \frac{1}{243} - \frac{2}{3645} \\
&= \frac{247}{7290}
\end{aligned}
$$

B Proof of Kernel Integral Equivalence

We have that

$$L(t) = \int_t^\infty K(u)du\mathbb{1}_{\{t>0\}} - \int_{-\infty}^t K(u)du\mathbb{1}_{\{t\leq 0\}}$$

in the case that $t > 0$

$$L(t) = \int_t^\infty K(u)du = \int_t^\infty \frac{1}{2}e^{-u}du = \frac{1}{2}e^{-t}$$

in the case that $t \leq 0$

$$L(t) = -\int_{-\infty}^t K(u)du = -\int_{-\infty}^t \frac{1}{2}e^{u}du = -\frac{1}{2}e^{t}$$

meaning $L(t) = \frac{1}{2}e^{|-t|} \cdot \text{sign}(t)$ except at $t = 0$ where $L(t) = -\frac{1}{2}$. From this follows that for all $t \neq 0$

$$L(t)^2 = \left(\frac{1}{2}e^{-|t|}\right)^2 = \frac{1}{4}e^{-2|t|} = K(t)^2$$

and at $t = 0$ we have that $L(t)^2 = \frac{1}{4} = K(t)^2$ once again. This means that

$$\int_{-\infty}^{\infty} L(u)^2 \, du = \int_{-\infty}^{\infty} K(u)^2 \, du$$

this completes the proof.

C Hyperparameters

The following are the hyperparameters used in the training runs of the LTCs presented in this thesis. For the training specific hyperparameter the name of the corresponding argument in PyTorch can be found in parentheses. If some argument of a PyTorch object is not listed, that means that the value was not altered and the default value as provided by the package was used.

LTC$^{\mathbf{V}}$

Window: 10
Batch Size: 40
Max Number of Epochs: 250
CPU cores (devices): 1
Features: Daily Realized Volatility
Network Size (neurons): 15
Wiring: NCP
Include Time Deltas: True
Time Delta Scale: 20
ODE Unfolds per state update (ode_unfolds): 12
Gradient Clip Value (gradient_clip_val): 1

Callbacks:
EarlyStopping
Metric (monitor): Validation Loss
Tolerance (patience): 25

Optimizer:
Adam
Step (learning_rate): 1×10^{-4}
Regularization (weight_decay): 1×10^{-5}

Scheduler:
Cosine Annealing

Cycles (T_max): 10
Floor (min_lr): 1×10^{-6}
Interval (interval): Epoch
Frequency (frequency): 1

LTC$^{\mathbf{K}}$

Window: 10
Batch Size: 40
Max Number of Epochs: 250
CPU cores (devices): 1
Features: Daily Realized Volatility
Network Size (neurons): 15
Wiring: NCP
Include Time Deltas: false
Time Delta Scale: 10
ODE Unfolds per state update (ode_unfolds): 12
Gradient Clip Value (gradient_clip_val): 1

Callbacks:
EarlyStopping
Metric (monitor): Validation Loss
Tolerance (patience): 25

Optimizer:
Adam
Step (learning_rate): 1×10^{-4}
Regularization (weight_decay): 1×10^{-5}

Scheduler:
Cosine Annealing
Cycles (T_max): 10
Floor (min_lr): 1×10^{-6}
Interval (interval): Epoch
Frequency (frequency): 1