Master thesis

The Plasticity Approach to the Weighting of Echo State Network Experts

June 15, 2020

Lucas Burger, Universität Konstanz
01/843636, Lucas.Burger@uni-konstanz.de

Contents

1 Introduction

Not only in light of the recent outbreak of the Corona virus pandemic (COVID-19), presumably starting in China and now spreading all over different continents, the world is in constant danger of falling apart. Even if this virus is a health risk to the human population, the reaction of financial markets has been immediate and severe. Not only the ever increasing number of infections but fears of a recession, the preparation for a stock market crash in the realms of the global financial crisis of 2008/2009 or the potential collapse of health care systems all around the world are dominating the news headlines these days. The strict restrictions of many governments in the form of curfews, working from home requirements and social distancing guidelines, influences listed companies and private companies equally. Those difficult conditions play out in stock markets as well as bond markets because of an increased risk of defaul or total bankruptcy. When governments file *Corona Bills* for the stimulus of their economies, \$2.2 trillion in the USA[1] or €165 billion in Germany[2], and central bank interventions become necessary, the currency markets also don't remain untouched. Finally, the combination with the oil price war between Russia and Saudi Arabia, which doesn't receive much attention beneath the corona virus headlines, weighing commodity markets, we are faced with uncertain times everywhere we look.

Not only the financial domain, with banks, insurance companies and investment funds, are at the frontline of constantly facing significant risks. Also other sectors a exposed to risks, such as production facilities relying on supply chains, retail stores constantly facing the forces of supply and demand, natural resource mining companies that rely on the discovery of new mineral deposits or utility companies providing basic infrastructure like water, sewage service, electricity or natural gas. Just to name a few.

While a thorough risk assessment becomes paramount in light of this imminent financial, health and supply crisis, it is also necessary in flourishing times like the previous year 2019, when the probability of an adverse event may likely be underestimated. Hence, proper risk manangement plays a very crucial role to soundly position and act in any of the markets mentioned above and any economic environment.

In this thesis we want to focus on the financial domain where the famously quoted trade-off between risk and expected reward, initially elaborated by Markowitz (1952), builds the foundation for acting in relation to and for modelling of financial returns. When striving for profit, Hedge Funds and high frequency trading have evaporated any predictablity of the expected reward - the first moment of the return distribu-

[1] https://www.washingtonpost.com/business/2020/03/25/trump-senate-coronavirus-economic-stimulus-2-trillion/, opened March 30th, 2020 10:25 pm.

[2] https://www.sueddeutsche.de/politik/coronavirus-covid-19-hilfspaket-wirtschaft-1.4857038, opened March 30th, 2020, 10:30 pm.

tion - exploiting market inefficiencies and invoking the efficient market hypothesis (Fama, 1970). While the definition of uncertainty in financial markets quickly escalates into the philosophical space, the volatility/variance - the second moment of the return distribution - is widely considered as a practical measure of risk and its forecasting has important implications for all participants, e.g. short-term traders, long-term investors and regulators.
While the first moment remains largely unpredictable, the second moment exhibits certain characteristics that can be exploited for the assessment and prediction of risk. Therefore, finding and using those patterns to make good forecasts of future volatilities is of great interest. This thesis is using reservoir computing which is naturally built to encounter and interpret patterns in any dataset, where the focus, as will become clear in a moment, lies predominantly on the time series data. Taking the stance of a model selection problem, we will use it trying to improve forecasts of daily realized volatilities.

1.1 Reservoir Computing

Reservoir Computing is a new emerging field of machine learning that employs large dynamical systems - called reservoirs - in order to perform certain tasks. The main motivation behind this approach is to map the (low dimensional) input to a higher dimensional representation and let the reservoir develop multiple patterns in its loosely coupled oscillators that can be exploited to perform a given task.
The earliest example of this concept can be found in Buonomano and Merzenich (1995) and has gained a lot of attention, especially over the last years with adaptions of reservoirs to different, more traditional machine learning models mostly related to neural networks. Originally and independently introduced by Jaeger (2003); Jaeger and Haas (2004) (under the name Echo State Network) and Maass et al. (2002) (under the name Liquid State Machine) the Echo State Network is one of many adaptations of Reservoir Computing, whose architecture resembles that of a recurrent neural network. Other approaches have been employed along the lines of shallow Neural Networks (Huang et al., 2006), Deep Neural Networks (Gallicchio and Micheli, 2017; Gallicchio et al., 2017), Convolutional Neural Networks (Ma et al., 2019), State Affine Systems (Grigoryeva and Ortega, 2018b) or Nonlinear Transient Computing (Crook, 2007).
Reservoir computing builds on the idea that a large excitable network (the reservoir) is presented with a signal and the connections - feed forward and recurrent - within the network produce excitable patterns that can be mapped to a (targeted) output. The advantage of reservoir computing over more traditional neural networks is the fact that the internal connections of the reservoir remain completely untrained after their random initialization. Hence, this enables larger networks going into multiple

1000 nodes[3]. The readout of the reservoir that produces an output is the only part of the system that is being trained. This step doesn't have to be sophisticated and most researchers stick to a linear regression in order to keep the simplicity and computational benefits of this approach. By narrowing down the training of the model to a linear regression, common problems of neural networks, namely vanishing gradients, local minima, high dimensionality and slow learning, are naturally tackled. Additionally, there is a vast literature on the estimation, inference and robustness of linear regression models that can be built on.
Given the dynamical nature of these networks, these models have mostly been used to work with time series data, i.e.

- wind power forecast (Xu et al., 2015)
- stock trading (Lin et al., 2011)
- prediction of dialysis in ICUs (Verplancke et al., 2010)
- useful life prediction of machines (Rigamonti et al., 2018)
- language modelling (Rachez and Hagiwara, 2014)

Requiring much less and more straightfoward training, the reservoir computing approach still suffers the choice of some hyperparameters. Apart from the training of the readout, those still include some parameters of neural networks like the number of hidden nodes, the appropriate scaling and shift of the input signal, a bias or the activation function, but also introduces some new hyperparameters in relation to the network's connections. This refers to the choice of a distribution for the initialization of connections, the connectivity in the recurrent connections of the network or the appropriate scaling of those internal connections to ensure network stability, all of which we are going into more detail later.
The search for a (task dependent) optimal set of hyperparameters weights heavily on the otherwise fast and easy training of reservoir computing models. Because of their iterative nature, the categorical choice for some or the non availability of analytical solutions for other hyperparameters, the employer of such a model is mostly left with a try-and-error approach. Sophisticated methods beyond mere grid search for the selection of those parameters are available, however,
the search can be circumvented by different approaches including

1. improving stability (Verzelli et al. (2019) or Jarvis et al. (2010))
2. array of reservoirs (Grigoryeva et al., 2016)
3. committee methods (Stanek, 2011)

[3]Compared to maybe a few hundert for traditional neural networks, which already stretches the bounds of numerical feasability.

where the second and the third employ a set of multiple reservoir computers each having a different configuration of hyperparameters. In this thesis we also want take the approach of multiple reservoir computer, namely Echo State Networks, and will consider them a set of experts for the preditive task daily realized volatilities.
While the properties and capabilities of reservoir systems have been studied in the artificial space, recent advances into the physical implementation of reservoir computing have gained a lot of attention building on electronic, photonic, spintronic, mechanical or even real biological reservoir computers (Tanaka et al., 2019).

1.2 Notation

Throughout this thesis we will, unless otherwise stated, use the following notation. Any deviations should be clear from the context.

x, y, u	Scalar values
$\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}$	Vectors, eg in $\mathbb{R}^n$
X, Σ, Γ	Matrices, e.g. in $\mathbb{R}^{n \times n}$
X'	Transpose of the matrix X
$[...; ...]$	Vertical concatenation of arrays, e.g. $X = [\mathbf{x}_1; ...; \mathbf{x}_n]$
$\mathbb{I}_n$	Idenity matrix of size $n \times n$
$\text{diag}(\mathbf{x})$	Maps $\mathbf{x} \in \mathbb{R}^n$ onto the diagonal of a matrix in $\mathbb{R}^{n \times n}$
$\mathbf{x} \odot \mathbf{y}$	Hadamard product, elementwise multiplication

1.3 Organisation

The remainder of this thesis is organised as follows: Section 2 will give a short introduction into the application of linear regression in the offline as well as the online setting. Section 3 will introduce Echo State Networks as one variant of the reservoir computing paradigm. This model relies on linear regression as training of the model, which motivates the separate introduction of linear regression in section 2. We will also dive into ways of tuning the reservoir of an Echo State Network beyond its random initialization. Section 4 then presents mixture of experts models as means of employing a committee of (machine learning) models to combine them into a single prediction. Building on the improvements made to Echo State Networks from section 3, we propose a novel way of weighting a set ESN experts by the plasticity of the network activation. This approach is mainly motivated by the fact that the experts' weigthing is updated prior to the prediction, which contrasts the standard loss induced weighting of experts. Section 5 will outline the application of the proposed models to real world financial data in the form of the daily realized volatility

of the IBM stock price traded on the New York stock exchange. It will also present the Heterogeneous Autoregressive model of Realized Volatility (HAR) model (Corsi, 2009) as a common benchmark for the prediction of realized volatility. Section 6 presents the empirical results, section 7 concludes and proposes further directions of research for the novel approach of *The Plasticity Approach to the Weighting of Echo State Network Experts*.

2 Linear Regression

Linear regression can be considered the most basic machine learning model which existed far before the term *machine learning* or *artificial intelligence* have been used. We will use linear regression in conjunction with Echo State Networks which motivates this short introduction into this matter. Section 2.1 will introduce the offline learning methods for linear regression and section 2.2 will present methods to estimate a linear regression model in an online fashion. Especially the latter approach suits the time series focus and iterative nature of reservoir computing.

2.1 Offline Learning

Asssume we have $k \in \mathbb{N}$ independent variables, denoted by $\mathbf{x} = (x_1, ..., x_k) \in \mathbb{R}^k$ and called regressors, and one independent variable which is referred to by $y \in \mathbb{R}$. We want to assess inference from $\mathbf{x}$ on y and want to find and quantify their relationship. Assume, we have $n \in \mathbb{N}$ observations of pairs $(\mathbf{x}_i, y_i)$, $i = 1, ..., n$ and gather those in matrices $X = [\mathbf{x}_1; ...; \mathbf{x}_n] \in \mathbb{R}^{n \times k}$ and $Y \in \mathbb{R}^n$. We want to find a linear relationship

$$Y = \alpha + X\boldsymbol{\beta} + \varepsilon \tag{2.1}$$

where $\alpha \in \mathbb{R}$ is an intercept, $\boldsymbol{\beta} \in \mathbb{R}^k$ is the vector of slopes and ε is an i.i.d. error term that follows a multivariate normal distribution, $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma_\varepsilon^2 \mathbb{I}_n)$. In this procedure, the fit is not perfect and we commit an error $e = Y - X\boldsymbol{\beta}$ as a realization of ε, which is called residual and is subject to a loss function $L : \mathbb{R}^2 \to \mathbb{R}$. The estimate $\hat{\boldsymbol{\beta}}$ of the slope $\boldsymbol{\beta}$ for the common *Mean Squared Error* $L : (\hat{y}, y) \mapsto (\hat{y} - y)^2$ is the ordinary least squares

$$\hat{\boldsymbol{\beta}} = (X'X)^{-1} X'Y = \underset{\boldsymbol{\beta} \in \mathbb{R}^k}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i \boldsymbol{\beta})^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^k}{\operatorname{argmin}} \sum_{i=1}^{n} e_i^2 \tag{2.2}$$

that minimizes the sum of squared residuals. Depending on the dimensionality of the regression model and the relationship of observations n and number of regressors k, regularization can become necessary. Regularization describes the procedure of restricting the estimation of a linear model to prevent overfitting (when $k > n$) and provide good generalization results on unseen data, e.g. when we want to predict y

based on a new observation of $\mathbf{x}$. One possibility for regularization is called Lasso Regression with

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{N\times L}}{\operatorname{argmin}} L(X\boldsymbol{\beta}, Y) + \lambda \left\|\boldsymbol{\beta}\right\|_1 \tag{2.3}$$

which has no closed form solution like 2.2 and has to be solved iteratively. Depending on the condition of the matrix $X'X$ in (2.2), it can also become crucial to regularize the matrix in order to prevent numerical instability when inverting $X'X$. This can be achieved by Ridge Regression

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{N\times L}}{\operatorname{argmin}} L(X\boldsymbol{\beta}, Y) + \lambda \left\|\boldsymbol{\beta}\right\|_2 \\ &= \left(X'X + \lambda \mathbb{I}_N\right)^{-1} X'Y \end{aligned} \tag{2.4}$$

which has a closed form solution and prevents numerical instability of the inversion if $\lambda > 0$ is chosen sufficiently large, which depends on the numerical accuracy and capability.

In order to achieve a good generalization of the predictive performance, the regularization parameter λ in (2.3) or (2.4) has to be chosen carefully, i.e. by k-fold cross validation which we will further elaborate on in section 5.4.

Another approach which finds application in the estimation of a model with heteroskedastic error terms, i.e. where different observations bear different levels of uncertainty, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2_{\varepsilon,i})$ depending on the index i, includes a weighting factor ρ_i to assign a different weight to the error committed for different observations. This approach is called weighted ordinary least squares. The error function then becomes

$$L = \sum_{i=1}^{n} \rho_i \left(y_i - \mathbf{x}_i\beta\right)^2 = \sum_{i=1}^{n} \rho_i e_i^2 \tag{2.5}$$

One way of weighting those observations and to account for the heteroskedasticity would be to account for the level of uncertainty and choose the weights as the inverse of the variance of the measurement, e.g. $\rho_i = \frac{1}{\sigma^2_{\varepsilon,i}}$. Another argument to bring forth in favor of weighted least squares would be in the setting of time series data. The fact, that older data may be less relevant for the prediction of future data than more recent observations, one can progressively punish older observations. Depending on the application an the amount of data, for example, this could be using $\rho_i = \rho^i$, for $0 < \rho < 1$, and older observations would receive an exponentially lower weight[4].

2.2 Online Learning

The term *online learning* is referring to a setup, where the training of any model is performed incrementally as opposed to the estimation procedures presented in

[4]Depending on the ordering of the data, this update can also be $\rho_i = \rho^{n-i}$ where $n = \#I$ is the number of observations.

section **??**. Especially in a time series setup, where by the nature of the data new information becomes available in a sequential order, online learning methods make an update of the best predictor for future data possible without the need to re-train the model on the whole dataset. In other areas where vast amounts of data are available for the training of machine learning, online learing methods become crucial because of computational limitations. As reservoir computing is predominantly concerned with time series data, a combination of an Echo State Network with sequential updates of the model parameters builds a powerful tool.
The recursive least squares algorithm (RLS) considers the ordinary least squares problem of equation (2.2). Assume, we have an indexed set of covariates $\mathbf{x}$ and target values y: $\{(\mathbf{x}_i, y_i) \in \mathbb{R}^{n+1} : n \in \mathbb{N}, i \in I\}$, for an index set I, and as in section (**??**) want to find the common relationship $y = \mathbf{x}\beta + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. The offline approach of section **??** would gather all covariates $\mathbf{x}$ into a regressor matrix X and the target values y into a vector Y. Instead of using (2.2) to estimate the relationship, the RLS algorithm updates

$$\begin{aligned} \Sigma_i &= \mathbf{x}_i'\mathbf{x}_i + \Sigma_{i-1} && (2.6) \\ \hat{\boldsymbol{\beta}}_i &= \hat{\boldsymbol{\beta}}_{i-1} - \Sigma_i^{-1}\mathbf{x}_i' \left(\mathbf{x}_i\hat{\boldsymbol{\beta}}_{i-1} - y_i\right) && (2.7) \end{aligned}$$

Where $\Sigma_0 := \mathbf{0} \in \mathbb{R}^{k\times k}$, $\hat{\boldsymbol{\beta}}_0 := \mathbf{0} \in \mathbb{R}^k$ and Σ_i is a running version of the expression $X'X$ from (2.2). In order to prevent matrix inversion of Σ_i by using the matrix inversion lemma[5] we can reformulate (2.6)-(2.7) and initialize an auxiliary $\Gamma_0 = \mathbb{I}_n \in \mathbb{R}^{n\times n}$. The update of $\hat{\beta}$ then changes to

$$\begin{aligned} \Gamma_i &= \Gamma_{i-1} - \frac{\Gamma_{i-1}\mathbf{x}_i\mathbf{x}_i'\Gamma_{i-1}}{1 + \mathbf{x}_i\Gamma_{i-1}\mathbf{x}_i'} && (2.8) \\ \hat{\boldsymbol{\beta}}_i &= \hat{\boldsymbol{\beta}}_{i-1} - \Gamma_i\mathbf{x}_i' \left(\mathbf{x}_i\hat{\boldsymbol{\beta}}_{i-1} - y_i\right) && (2.9) \end{aligned}$$

Kushner and Yin (2003) shows that this iteration is equivalent to the OLS estimate (2.2). The proof also includes the relationship $\Gamma_i = \Sigma_i^{-1} := (X'X)^{-1}$. The regularized version along the lines of ridge regression (2.4) with regularization strength $\lambda_r > 0$ can be achieved by setting $\Gamma_0 = (1 + \lambda_r\mathbb{I}_n)^{-1}$ and the iterations yield $\Gamma_i = (\Sigma_i + \lambda_r\mathbb{I}_n)^{-1}$ which is one part of equation (2.4).
The recursive least squares can be further extended for the inclusion of a weighting factor $0 < \rho < 1$ as in the weighted ordinary least squares approach (2.5). Based on

$$\Sigma_i = \mathbf{x}_i'\mathbf{x}_i + \rho\Sigma_{i-1} \tag{2.10}$$

To introduce an exponential weighting factor ρ, the recursive update of (2.8)-(2.9)

[5] More specifically, the Sherman-Morrison Formula, see appendix A.1.1.

becomes

$$\Gamma_i = \Gamma_{i-1} - \frac{\Gamma_{i-1}\mathbf{x}_i\mathbf{x}_i'\Gamma_{i-1}}{\rho + \mathbf{x}_i\Gamma_{i-1}\mathbf{x}_i'} \tag{2.11}$$

$$\hat{\boldsymbol{\beta}}_i = \hat{\boldsymbol{\beta}}_{i-1} - \Gamma_i\mathbf{x}_i'\left(\mathbf{x}_i\hat{\boldsymbol{\beta}}_{i-1} - y_i\right) \tag{2.12}$$

which is equivalent to the exponential weighting of (2.5), namely $\rho_i = \rho^i$.

3 Echo State Networks

Neural networks mimic the human brain in its information processing and can be distinguished into two classes: feedforward neural networks (FFNs) and recurrent neural networks (RNNs). As the names suggest, the feedforward neural networks process information in a sequential structure. They may consist of multiple layers who enable complex nonlinear calculations. Recurrent neural networks may contain cyclical paths and are able to remember past information. When the input has an autoregressive structure, the RNN can remember past information and is much more performant than a feedforward neural network. However, the optimization of these networks is computationally expensive and can become numerically critical. Gradients can vanish or the optimization can become stuck in local minima which makes the optimization of networks inefficient (Bengio et al., 1994).
Echo State Networks naturally tackle the problem of vanishing gradients or local minima. They have the structure of a recurrent neural network, but are much larger and the internal connections of the neural network remain fixed during the training which reflects the concept of reservoir computing.
Grigoryeva and Ortega (2018a) show the universality of Echo State Networks for discrete-time fading memory filters with uniformly bounded inputs. Gonon and Ortega (2018) extend the universality to stochastic semi-infinite inputs[6].
In the following, we want to focus on the Echo State Network which is desirable because of its simplicity and fast training. It is also the model that has been worked with the most.

3.1 Architecture

An Echo State Network, as mentioned earlier, resembles a recurrent neural network, especially in its architecture. It is also sometimes referred to as an extension of recurrent neural networks, but this is difficult so say. Basically, it only is a different approach to the training and usage of recurrent neural networks, keeping most of the network untrained. Let $K \in \mathbb{N}$ be the number of inputs, $N \in \mathbb{N}$ be the number of neurons and $L \in \mathbb{N}$ be the number of outputs. Let $\mathbf{x}_t = (x_1^t, ..., x_N^t) \in \mathbb{R}^N$ be the

[6]Universality is referring to arbitrary close approximation of functionals in $L^p(\Omega, \mathcal{F}, \mathbb{P})$ on a probability space Ω with filtration $\mathcal{F}$ and probability measure $\mathbb{P}$.

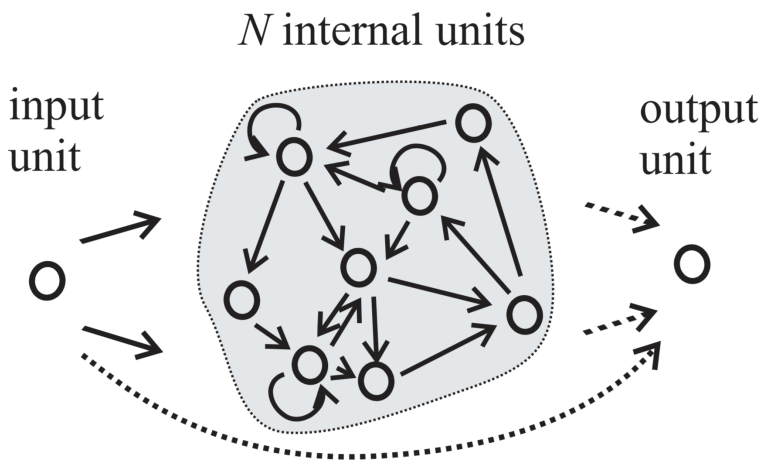

Figure 3.1: The basic structure of an ESN. Figure taken from Jaeger (2003). Dotted lines represent trainable connections, whereas solid arrows are fixed connections.

row vector of network activations at time $t \in \mathbb{Z}$ and $f : \mathbb{R} \to \mathbb{R}$ be an elementwise applied activation function. All connections within the system will be denoted by matrices. We will denote by $W_{in} \in \mathbb{R}^{N \times K}$ the connections from the input to the reservoir, by $W \in \mathbb{R}^{N \times N}$ we will denote the internal connections and $W_{out} \in \mathbb{R}^{L \times N}$ will represent the hidden-to-output connections. The network will also be equipped with a bias $\mathbf{b} = b \cdot W_{bias} \in \mathbb{R}^N$. Let $(u_t)_{t \in \mathbb{Z}}$ be an input signal which is discretized in time. Then, the evolution of the network - producing an output value y_t - can be characterized as follows:

$$\mathbf{x}_t = f\left(W\mathbf{x}_{t-1} + W_{in}u_t + \mathbf{b}\right) \tag{3.1}$$

$$y_t = \mathbf{x}_t W_{out} \tag{3.2}$$

where the bias $\mathbf{b} = b \cdot W_{bias}$ for a fixed and random matrix $W_{bias} \in \mathbb{R}^N$ and $b \in \mathbb{R}$. Usually, (3.2) is presented as $y_t = h(\mathbf{x}_t)$ for any map $h : \mathbb{R}^n \to \mathbb{R}$ but we will focus on the linear case. The readout h can also incorporate the input but for the ease of notation we will work with none of the two. The expression W_{out} is to be seen as $\boldsymbol{\beta}$ from section 2. The convincing argument of Echo State Networks, which also translates to other reservoir computing models, is the fact that all of its architecture except for W_{out} is left untrained and doesn't require any training. The *training* of the hidden-to-output connections, by choice of h as linear function, finally boils down to the estimation of a linear regression from section 2, where W_{out} is now referring to the parameter vector $\boldsymbol{\beta}$ from that section. Figure (3.1) presents the basic ESN architecture.

Most of the time, the network is being adapted to a specific time series and is trained to perform 1-step ahead predictions, so $u = By$[7].

[7]B is the backshift operator is defined as $B : (\mathbb{R}^n)^{\mathbb{Z}} \to (\mathbb{R}^n)^{\mathbb{Z}}$, $(z_t)_{t \in \mathbb{Z}} \mapsto (z_{t-1})_{t \in \mathbb{Z}}$.

Other versions of the basic model can include feedback connections from the output back to the reservoir or the application of leaky-integrator neurons (Jaeger et al., 2007). For $\alpha \in [0,1]$ and $W_{fb} \in \mathbb{R}^{N \times L}$ the dynamics of the system (3.1) accordingly change to

$$\tilde{\mathbf{x}}_t = W\mathbf{x}_{t-1} + W_{in}u_t + \mathbf{b} + W_{fb}y_{t-1} \tag{3.3}$$

$$\mathbf{x}_t = f\left(\alpha\mathbf{x}_{t-1} + (1-\alpha)\tilde{\mathbf{x}}_t\right) \tag{3.4}$$

$$y_t = \mathbf{x}_t W_{out} \tag{3.5}$$

The leak parameter α can be interpreted as a parametrical way of keeping information within the reservoir nodes. For example, a value of $\alpha = 0.2$ would make the neuron keep 20% of its current information and only allow the newly arriving value of $\tilde{x}(t)$ to complement this value with a weight of 80%. Leaky-integrator neurons would theoretically be possible for the remainder of this thesis but have not been used.

Feedback connections pose a challenge to the training of an Echo State Network because the output connections W_{out} have to be solved iteratively (because of the need for an output value y_{t-1} in the update 3.3). This could be achieved by an online learning approach (see section 2.2). However, if the input is chosen as $u = By$ to produce one-step-ahead forecasts of y, the target output y_t is the new input to the network by design.

3.2 Training

In order to map the network state to a desired output, W_{out} in (3.5) can be trained using a linear regression model from section 2. Hence, the network is being fed the input signal and activations x_t are gathered in a matrix $\tilde{X} = [\mathbf{x}_0'; \mathbf{x}_1'; ...; \mathbf{x}_T'] \in \mathbb{R}^{T \times N}$. This regressor matrix can be further enhanced by the addition of an intercept or by the the network input u so that $X = \left(1_N, u, \tilde{X}\right) \in \mathbb{R}^{T \times (2+N)}$. One could also employ more sophisticated (machine learning) models to map the network activations and the desired output. However, the Echo State Network convinces by its simple and fast training. Additionally, the theory of (high dimensional) linear regression has been studied extensively.

In the training of W_{out} one has to keep in mind, that the network state has been initialized with $\mathbf{x}_0 = \mathbf{0} \in \mathbb{R}^N$. To wash out the effect of the initial state and to prevent misleading training results, an initial transient τ of observations has to be dropped. The echo state property which will be introduced in section 3.4 will guarantee, that the effect of (any) initialization fades after several updates of the network. This transient can be chosen as a fixed number, or - depending on the size of the dataset - can be set to a fraction of the whole dataset, i.e. $\tau = 0.05 \cdot T$. The regressor matrix is truncated accordingly.

3.3 Hyperparameters

Training of Echo State Networks is simple and fast. Most of the connections within the network are initialized randomly and will not be trained to adapt the system to a given task. However, the initialization of those connections has - as almost all machine learning models do - some hyperparameters that need to be chosen in advance. The choice of (some of) these parameters plays a crucial role in the performance of the model and therefore need special attention. Analytic derivations of optimal parameters are very difficult because of the dynamic and iterative nature of the reservoir. For some there is no analytical solution and other methods such as basic grid search, particle swarm optimization (Basterrech et al., 2015) or gradient descent (Jaeger et al., 2007) have to be employed. This weighs heavily on the training of time of Echo State Networks. On the other hand, this is also the case for other machine learning models. In the following, we want to present the hyperparameters that are specific to Echo State Networks and come as a tradeoff for not having to train internal connections. We will go into more detail on the methodology of choosing hyperparameters in section 5.4.

3.3.1 Underyling Probability Distribution of Connections

In order to initialize the fixed connections within the dynamical system (i.e. W_{in}, W, b), one has to choose some probability distribution[8]. Typical choices include the discrete bi-valued[9] $\mathcal{U}\{-1, 1\}$, the uniform $\mathcal{U}(-1, 1)$ or the normal distribution $\mathcal{N}(0, \sigma^2)$ for some $\sigma > 0$. Even though, Wu et al. (2018) show that the choice of the distribution has an effect on performance, the probability distribution as a hyperparameter doesn't receive much attention and researchers rarely shed any light on their specific choice. Other sources state, that different distributions give almost the same performance because the other hyperparameters are of much higher importance (Lukoševičius, 2012). After all, it is worth noting that the distribution of internal weights bears some freedom of choice and may possibly be worth exploring for some tasks.

3.3.2 Sparsity

Sparsity is referring to the number of non-zero elements in the internal connections W. Jaeger and Haas (2004) use a 1% connectivity to establish a richly structured reservoir of excitable dynamics. Apart from the richness of the reservoir, the computational cost of updating the reservoir is significantly reduced when the connectivity is chosen in relation to the number of total connections. By choosing the sparsity as

[8] Typically, the same distribution is chosen for all random initializations.

[9] The discrete bi-valued distribution of weights has a positive chance of creating identical neurons but makes analysis of the reservoir dynamics much easier.

a fraction of total connections by $s = 10/N^2$ the cost of updating the network only grows linearly with the size of the reservoir instead of quadratically.

3.3.3 Spectral Radius

The spectral radius of the internal connections W is the most important hyperparameter for the Echo State Network as it is mainly responsible for the echo state property and the memory capacity of the reservoir. The larger the spectral radius, the more dominant is the echo state (the first part of the update 3.1) of the reservoir compared to the new input. So a higher spectral radius means that the reservoir has longer memory and a smaller spectral radius renders the reservoir predominantly a respresentation of the most recent input (Lukoševičius, 2012). Depending on the task at hand, one can use some intuituion on the range of plausible spectral radii.

3.4 Echo State Property

The echo state property (ESP) is a necessary condition for the network to be able to produce exploitable patterns. Loosely speaking, the echo state property states that the network state is asymptotically independent of its initial state (Jaeger, 2003). In other words, the network will forget its initial state and will only depend on a finite set of inputs u_t from a compact set and will forget its initialization, typically $\mathbf{0} \in R^N$. The literature offers different approaches to ensure the echo state property. Orginially, Jaeger (2001) stated that choosing the internal connections W such that the spectral radius $\rho(W) = \max\{\lambda : \lambda \text{ eigenvalue of} W\}$ is less than unity, would ensure the echo state property for all inputs u. This lead to a widespread misconception that the spectral radius always *has* to be smaller than unity. However, depending on the input, larger spectral radii can also be possible without counteracting the echo state property (Yildiz et al., 2012). Yildiz et al. (2012) also propose to look at the spectral radius of the corresponding positive matrix $\rho(|W|)$ and to ensure that its spectral radius is less than unity. Ozturk et al. (2007) propose a procedure to maximize the average state entropy by a spectrum of eigenvalues in the complex plane. Strauß et al. (2012) state that the echo state property should be based on the singular values of the connections W instead of its spectral radius. They propose to choose the singular values to be smaller than 1 to ensure the echo state property. Additionally, the activation function has to be Lipschitz continuous with constant 1[10]. They obtain a sparse and orthogonal matrix W whose singular values and eigenvalues are known to ensure the echo state property. As there is now direct way to ensure the echo state property for a given input sequence, and some of the above restrictions limiting the performance of the network, the choice of hyperparameters and ensuring for the echo state property falls back to a try-and-error

[10]Which is the case for *tanh*.

methodology. It is desirable to bring the network dynamics towards the *edge of chaos*, which means to push the dynamics close towards the border between a stable and an unstable regime (L. Büsing, 2010; Robert Lengenstein, 2007) to improve computational outcome.
The choice of the hyperparameters, especially the spectral radius, motivates the following section which targets the tuning of the reservoir beyond its random initialization.

3.5 Tuning of the Reservoir Dynamics

The motivation for the inclusion of this section stems from the nature of another reservoir computing model, namely the State Affine Systems (Grigoryeva and Ortega, 2018b), which changes its dynamics given newly presented input. An Echo State Network is indeed dynamical system, however, the connections within the network are fixed and don't change. They can be interpreted as a restricted and slightly modified State Affine System, where its dynamics, namely W or W_{in}, are functions of the input. This motivated the search for ways to modify the Echo State Network to react differently to different inputs, adapt its dynamics or in general to further improve the reservoir after initialization.
Different endeavours have been undertaken to tune the reservoir beyond the random initialization and the selection of hyperparameters. As mentioned earlier, the most import hyperparameter is the spectral radius[11]. Triesch (2005) uses an unsupervised intrinsic learning approach to change the neurons excitability beyond the its random intialization. Using a continuous activation model, he tries to bring the output of each neuron as close as possible to an exponential distribution.
In order to tune the neuron's output, he transforms the original activation function $f : \mathbb{R} \to \mathbb{R}$, $\mathbf{x} \mapsto \frac{1}{1+\exp(-\mathbf{x})}$ (fermi activation function) using

$$f_y(y) = f(\boldsymbol{\alpha} \odot \mathbf{x} + \boldsymbol{\beta}) \tag{3.6}$$

with tunable parameters $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^N$. To match the notation of Triesch (2005) and Schrauwen et al. (2008), y in this instance is not be confused with the network output but represents the post-activation network state. We use vector notations for $\mathbf{x}, \boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as each element of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is related to a neuron in the network. The modified network update not only tunes the spectral radius but also incorporates the bias $\mathbf{b}$. The effective update of the network then becomes

$$\mathbf{x}_t = f\left(\operatorname{diag}(\boldsymbol{\alpha}) W \mathbf{x}_{t-1} + \operatorname{diag}(\boldsymbol{\alpha}) W_{in} u_t + \boldsymbol{\alpha} \odot \mathbf{b} + \boldsymbol{\beta}\right) \tag{3.7}$$

Using a gradient descent method, he minimizes the Kullback-Leibler divergence of

[11] Other hyperparameters are important as well, but the spectral radius has the largest effect on the performance of the network.

the transformed output y and the desired output distribution.

$$
\begin{aligned}
D_{KL}\left(f_y \,||\, f_{exp}\right) &= \int f_y(y) \log\left(\frac{f_y(y)}{\frac{1}{\mu}\exp\left(\frac{-y}{\mu}\right)}\right) \mathrm{d}y && (3.8)\\
&= \int f_y(y) \log\left(f_y(y)\right) \mathrm{d}y - \int f_y(y) \left(-\frac{y}{\mu} - \log\left(\mu\right)\right) \mathrm{d}y && (3.9)\\
&= -H(y) + \frac{1}{\mu}\mathbb{E}\left[y\right] + \log\left(\mu\right) && (3.10)
\end{aligned}
$$

Where $H(y)$ is the entropy of y, and $\mathbb{E}\left[\cdot\right]$ is the expectation. In other words, he tries to make the output y be as close as possible to $\frac{1}{\mu}\exp\left(-y/\mu\right)$. The last term $\log\left(\mu\right)$ is constant and the second term $\frac{1}{\mu}\mathbb{E}\left[y\right]$ is equal to 1 if y follows an exponential distribution with parameter μ. This shows, that the exponential distribution is the maximum entropy distribution (Triesch, 2005). Therefore the Kullback-Leibler divergence is minimized, when the entropy is maximized[12]. Using the gradient with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, he comes up with the following stochastic gradient descent rule for $\boldsymbol{\alpha} := \boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}$ and $\boldsymbol{\beta} := \boldsymbol{\beta} + \Delta\boldsymbol{\beta}$ with

$$
\begin{aligned}
\Delta\boldsymbol{\beta} &= \eta\left(1 - \left(2 + \frac{1}{\mu}\right) y + \frac{1}{\mu} y^2\right) && (3.11)\\
\Delta\boldsymbol{\alpha} &= \eta\left(\frac{1}{\boldsymbol{\alpha}} + x - \left(2 + \frac{1}{\mu}\right) \mathbf{x}y + \frac{1}{\mu}\mathbf{x}y^2\right) && (3.12)\\
&= \frac{\eta}{\boldsymbol{\alpha}} + \Delta\boldsymbol{\beta}x && (3.13)
\end{aligned}
$$

where $0 < \eta < 1$ is a small learning rate.

Different extensions of Triesch (2005) have been proposed, including Boedecker et al. (2009) who match the neurons' output to a Laplace distribution. Another extension is provided by Schrauwen et al. (2008) who target a gaussian output distribution $\mathcal{N}\left(\mu, \sigma^2\right)$ using the hyperbolic tangent as activation function which is the maximum entropy with support $[-\infty, \infty]$. They follow the argumentation along the lines of Triesch (2005) and come up with similar updates of the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, namely[13]:

$$
\begin{aligned}
\Delta\boldsymbol{\beta} &= -\eta\left(2y + \frac{1}{\sigma^2}\left(1 - y^2\right) \odot \left(y - \mu\right)\right) && (3.14)\\
\Delta\boldsymbol{\alpha} &= \frac{\eta}{\boldsymbol{\alpha}} + \Delta\boldsymbol{\beta} \odot \mathbf{x} && (3.15)
\end{aligned}
$$

(Schrauwen et al., 2008) recognize, that the boundedness of the activation function influences the output distribution of the neurons and in turn imposes a cer-

[12] This implies the direct connection of the fermi activation function and the exponential distribution.

[13] They use an equivalent notation for (3.14): $\Delta\beta = -\eta\left(-\frac{\mu}{\sigma^2} + \frac{\mathbf{y}}{\sigma^2}\left(2\sigma^2 + 1 - \mathbf{y}^2 + \mu\mathbf{y}\right)\right)$.

tain constraint on the moments of the infinite normal distribution. They analyse the behaviour of their online update and the resulting empirical distributions and find that for certain choices of μ and σ the boundedness significantly truncates the distribution. However, this problem can be easily solved by introducing another transformation of the activation function which is one of the contributions of this thesis. Given $c > 0$ we transform the activation function

$$f_y(y) = c \cdot \tanh\left(\frac{\boldsymbol{\alpha} \odot \mathbf{x} + \boldsymbol{\beta}}{c}\right) \tag{3.16}$$

such that the activation functions maps to the interval $[-c, c]$. Choosing c depending on the parameters of the normal distribution, i.e. the 99.5% quantile or any arbitrary value larger than that, the probability mass that is being truncated by the activation function can be chosen arbitrarily small. The online learning rule for α and β straightforwardly becomes:

$$\Delta\boldsymbol{\beta} = -\eta\left(\frac{2y}{c^2} + \frac{1}{\sigma^2}\left(1 - \frac{\mathbf{y}^2}{c^2}\right)(\mathbf{y} - \mu)\right) \tag{3.17}$$

$$\Delta\boldsymbol{\alpha} = \frac{\eta}{\alpha} + \Delta\beta\mathbf{x} \tag{3.18}$$

which is equivalent to (3.14)-(3.15) for $c = 1$. This enables us to choose any value of $\sigma > 0$[14]. The choice of μ is still limited but can be neglected. This is because network activations centered around 0 or centered around $\mu \neq 0$ provide the same information and the linear readout can take care of any necessary shift. Figure 3.5 presents the effect of the adaptation of the reservoir towards a target output distribution. The fit is clearly improved and the introduction of the scaling parameter c is justified as we find activation values outside $[-1, 1]$.

Furthermore, Koprinkova-Hristova and Palm (2011) show that the pre-training of an Echo State Network using intrinsic plasticity improves the network stability. They find a close relationship between IP pre-training and stabilization of the reservoir dynamics, such that even initially unstable reservoirs become stable by pushing reservoir activations towards the predefined mean μ of the normal distribution. They derive the Kullback-Leibler divergence as

$$D_{KL} = -H(y) + \frac{1}{2\sigma^2}\mathbb{E}\left[(y - \mu)^2\right] + \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) \tag{3.19}$$

which shows the compromise between entropy maximization and minimization of the distance between μ (typically chosen as 0) and y.

[14]Other activation functions like the *arcsinh* (Kim and Adali, 2001) or a rectified natural logarithm (Liu et al., 2019) have been considered to achieve the goal of variable boundedness of the activation function but given the solution presented above, have not been further explored.

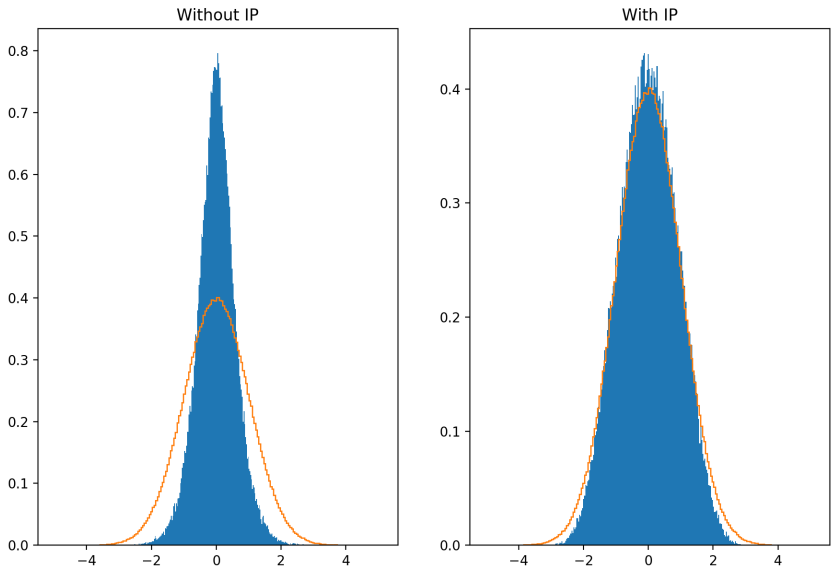

Figure 3.2: Network activations of a standard Echo State Network of size $N = 1000$ with initial spectral radius of $\rho = 0.95$. Its activation function, as can be seen by the range of activation values, has been adjusted to allow for a wider range of values. The fit of the last 100 network activations is clearly improved by the transformation. Additionally, the activation values are larger than 1 which supports the additional of the scaling parameter c. Adaptation of the reservoir has been performed for 100 epochs feeding the Mackey-Glass timeseries of length 5000 and the scaling of the activation function was chosen as $c = 5$.

4 Mixtures of Experts Models

Mixture of experts models cover a wide range of mixture models where multiple expert candidates compete for best performance. Consider yourself a forecaster and you have a certain number of experts or candidates for advice around you and you want to combine or mix their advice into one prediction. The mixture components can be modelled by covariates or functions of the individual candidate's performance. Having its origin in the machine learning literature, they do appear in many guises (Gormley and Frühwirth-Schnatter, 2018). The main goal is to predict a unknown sequence $y_1, y_2, ...$ of independent and identically distributed outcome variables. Subject to a loss function, a natural forecasting strategy would be to combine the experts' predictions $\hat{y}_t^k$, $t = 1, 2, ...$ in a weighted average to get the aggregate forecaster's predictions $\hat{y}_1, \hat{y}_2, ...$, where k is referring to one of the experts, while keeping the *cumulative regret* (or simply *regret*) with respect to each expert small. The performance of each expert is evaluated subject to a loss function $L : \mathbb{R}^2 \to \mathbb{R}_{>0}$, $(\hat{y}_t, y_t) \mapsto L(\hat{y}_t, y_t)$ which enables the definition of the regret as (Cesa-Bianchi and Lugosi, 2006)

$$R_n^k = \sum_{t=1}^{n} \big(L(\hat{y}_t, y_t) - L(\hat{y}_t^k, y_t)\big) = \hat{L}_n - L_{k,n} \tag{4.1}$$

where $\hat{L}_n$ is the forecaster's cumulative loss and $L_{k,n}$ is the cumulative loss of expert k. Accodingly, the instant regret to expert k can be defined by $r_{k,t} = L(\hat{y}_t, y_t) - L(\hat{y}_t^k, y_t)$. One can interpret this quantity as the regret the forecaster feels for not having listened to the advice of expert k in the prediction of y_t. The main motivation for the employment of multiple experts over a single candidate is the expectation that an ensemble outperforms a single expert (*The wisdom of crowds*). In this sense, the regret should vanish for an increasing number n of observations. That is

$$\frac{1}{n}\left(\hat{L}_n - \min_k L_{k,n}\right) \to 0 \tag{4.2}$$

for $n \to \infty$. The performance and the ensurance of this goal can only be evaluated in hindsight but regret bounds do exist for different algorithms.

Hence, assume we have finite number $K \in \mathbb{N}$ of possible experts or components and a weighting $\mathbf{w}_t = \left(w_t^1, ..., w_t^K\right) \in [0, 1]^K$ $(t = 1, 2, ...)$ at time t of those experts such that $\sum_{k=1}^K w_t^k = 1$ for all t. Each expert k $(k = 1, ..., K)$ produces a forecast, which in its most general form can be described by a density $f_k(\cdot \,|\, \theta_k)$ depending on a parameter $\theta_k(\mathbf{x}_t)$ which can be based on a set of covariates $\mathbf{x}_t$ (Gormley and Frühwirth-Schnatter, 2018). The goal is to sequentially find a good weighting $\mathbf{w}_t$ of the experts to combine their predictions. On the basis of the experts' predictions, a final prediction is computed as

$$p(y_t \mid \mathbf{x}_t) = \sum_{k=1}^{K} w_k f_k(y_t \mid \theta_k(\mathbf{x}_t)) \tag{4.3}$$

This model, by allowing $\mathbf{w}_t$ and $\theta_k(\mathbf{x}_t)$ to vary in different ways, allows for a wide range of applications, one of which we want to highlight in the following section. One common way to model $\mathbf{w}_t$ and $\theta_k(x_t)$ is to choose a point forecast $\hat{y}_t^k$ instead of a density $f_k(y_t \mid \theta_k(x_t))$. The forecaster chooses a weighting $\mathbf{w}_t$ of the experts and combines their individual point forecasts $\hat{y}_t^k$ by

$$\hat{y}_t = \sum_{k=1}^{K} \mathbf{w}_t^k \hat{y}_t^k \tag{4.4}$$

and - by observing the true value y_t - suffers a non-negative loss $L(\hat{y}_t, y_t)$. Now the performance of each expert can be judged based on the loss function L or equivalently on each expert's regret R_n^k from (4.1).

4.1 Loss Induced Weighting of Experts

It seems intuitive to update the weighting of the experts based on their performance, such that an expert with a high regret receives a higher weight because it would have been advisable to 'listen' to him and an expert with a comparably lower regret receives a lower weight. This can be achieved by different approaches (Cesa-Bianchi and Lugosi, 2006)

$$\text{Linear:} \quad w_t^k = \frac{R_t^k}{\sum_{i=1}^{K} R_t^i} \tag{4.5}$$

$$\text{Polynomial:} \quad w_t^k = \frac{2(R_t^k)_+^{p-1}}{\left(\sum_{i=1}^{K} (R_t^i)^p\right)^{(p-2)/p}} \tag{4.6}$$

$$\text{Exponential:} \quad w_t^k = \frac{\exp\left(\eta R_t^k\right)}{\sum_{i=1}^{K} \exp\left(\eta R_t^i\right)} \tag{4.7}$$

to differently weight the regret where the fractions are needed to normalize the weights to ensure $\sum_{k=1}^{K} w_t^k = 1$. $(\,\cdot\,)_+$ is referring to the positive part, so $(x)_+ = \max\{0, x\}$. The exponential update introduces a learning rate $\eta > 0$ which bears a drawback of the exponential weighting because of the introduction of a hyperparameter that may be subject to optimization. Nevertheless, we will focus on this approach because of its nice properties and the similarity to the plasticity weighting which we will introduce in section 4.2.3. One major advantage of the exponentially weighted approach is the fact that it can be equivalently formulated as an

incremental update of the weights by inserting (4.1)

$$
\begin{aligned}
w_t^k &= \frac{\exp\left(\eta R_t^k\right)}{\sum_{i=1}^K \exp\left(\eta R_t^i\right)} && (4.8)\\
&= \frac{\exp\left(\eta \sum_{\tau=1}^t \left(L(\hat{y}_\tau, y_\tau) - L(\hat{y}_\tau^k, y_\tau)\right)\right)}{\sum_{i=1}^K \exp\left(\eta \sum_{\tau=1}^t \left(L(\hat{y}_\tau, y_\tau) - L(\hat{y}_\tau^i, y_\tau)\right)\right)} && (4.9)\\
&= \frac{w_{t-1}^k \exp\left(\eta \left(L(\hat{y}_t, y_t) - L(\hat{y}_t^k, y_t)\right)\right)}{\sum_{i=1}^K w_{t-1}^i \exp\left(\eta \left(L(\hat{y}_t, y_t) - L(\hat{y}_t^i, y_t)\right)\right)} && (4.10)\\
&= \frac{w_{t-1}^k \exp\left(-\eta L(\hat{y}_t^k, y_t)\right)}{\sum_{i=1}^K w_{t-1}^i \exp\left(-\eta L(\hat{y}_t^i, y_t)\right)} && (4.11)
\end{aligned}
$$

This update of weights only requires the last performance of the experts as opposed to the linear or polynomial update that depend on the regret and therefore on past predictions $\hat{y}_s^k$, for $s < t-1$ and $k = 1, ..., K$. Different choices of η can lead to different performance results and one can circumvent its optimization by knowledge of the forecasting horizon which can help to choose η in advance.
One objective of the forecaster is to choose the learning rate of any algorithm for w_t such that the regret is low. Cesa-Bianchi and Lugosi (2006) prove different performance bounds for the weight updates mentioned above. The exponential update for a convex loss function $L : \mathbb{R}^2 \to [0, 1]$ is proven to be

$$
\hat{L}_n - \min_{k=1,...,K} L_{k,T} \leq \frac{\log(K)}{\eta} + \frac{T\eta}{8} \tag{4.12}
$$

for any $T > 0$ referring to the number of observations and any $\eta > 0$. Minimizing this upper bound with respect to η one can choose $\eta = \sqrt{8\log(K)/T}$ which results in the minimal upper bound of $\sqrt{(T/2)\log(K)}$. This procedure, however, requires prior knowledge of the forecasting horizon T. If this is not known in advance, Cesa-Bianchi and Lugosi (2006) further propose a time dependent learning rate η_t that has a regret bound that holds uniformly over time. By choosing

$$
\eta_t = \sqrt{8 \ln K / t} \tag{4.13}
$$

which comes back to an argument of resetting the optimal learning rate in (4.12) after every prediction and considering only 1-step ahead prediction horizons, the regret bound becomes .....

4.2 Pre-Prediction Update of Weights

One of the main drawbacks of the loss induced expert weighting introduced in section 4.1 is the fact that the weighting of the experts can be adjusted *after* the loss has been suffered, i.e. after the prediction. To our knowledge, no research in this specific setting has been undertaken so far to update the weights of a set of Echo

State Networks *prior* to the prediction step. Coming up with measures to judge an expert's expected performance seems difficult but different sources lead to main part of this thesis which tackles this problem.

4.2.1 Gaussian Mixture Autoregressive Model

Kalliovirta et al. (2015) present a Gaussian Mixture Autoregressive Model (GMAR) where they use a particular choice of the mixing weights for a set of $K \in \mathbb{N}$ experts, each of them being an auxiliary, stationary AR(p) process. The stationary solution of such an AR(p) process specifies a mean value and an autocovariance structure on the time series. The mixing of those processes (experts) is based on the likelihood that the last p observations of the target time series have been generated by any of those processes. Set $\mathbf{y}_{t-1} = (y_{t-1}, ..., y_{y-p}) \in \mathbb{R}^p$. They use the multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}^k, \Sigma^k)$, $\mu^k \in \mathbb{R}^p$ and $\Sigma^k \in \mathbb{R}^{p \times p}$ for $k = 1, ..., K$, with density

$$f^k(\mathbf{y}_{t-1}) = \frac{1}{\sqrt{(2\pi)^k |\Sigma^k|}} \exp\left(-\frac{1}{2} \left(\mathbf{y}_{t-1} - \boldsymbol{\mu}^k\right)' (\Sigma^k)^{-1} \left(\mathbf{y}_{t-1} - \boldsymbol{\mu}^k\right) \right) \tag{4.14}$$

Subject to a mean weight $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_K) \in \mathbb{R}^k$, they define the mixing weights $\alpha_{k,t}$ as

$$\alpha_{k,t} = \frac{\alpha_k f^k(\mathbf{y}_{t-1})}{\sum_{i=1}^{K} \alpha_i f^i(\mathbf{y}_{t-1})} \tag{4.15}$$

This method of weight update is - as within the expert setting - performed before the predition of the next value as one has uses the observations up to y_{t-1} in order to predict y_t which makes the process $\mathbf{y}_t$ Markovian.

4.2.2 Echo State Incremental Gaussian Mixture Network

Another, very rich, line of research is (Engel and Heinen, 2010; Heinen et al., 2011; Heinen, 2011). In one of their publications, Pinto et al. (2011) propose a novel algorithm for incremental temporal pattern processing called the Echo State Incremental Gaussian Mixture Network (ESIGMN). It combines Echo State Networks with the Incremental Gaussian Mixture Network (IGMN) from Heinen et al. (2011).
The Incremental Gaussian Mixture Network is composed of multiple sets of virtual neurons (referred to as cortical regions), all of which have size $M \in \mathbb{N}$ and an association region. Figure (4.2.2) from their paper visualizes the model.
The IGMN is a gaussian mixture model with an agressive learning process, meaning the training data only has to be presented once. Based on $K \in \mathbb{N}$ distribution components (these are the *cortical regions*) the network can adapt to the information from a new data point. One can think of the cortical regions as individual neural networks. Presented with a new input u[15], a running mean μ_k and running covari-

[15]The notation has been adjusted to match our setting so far.

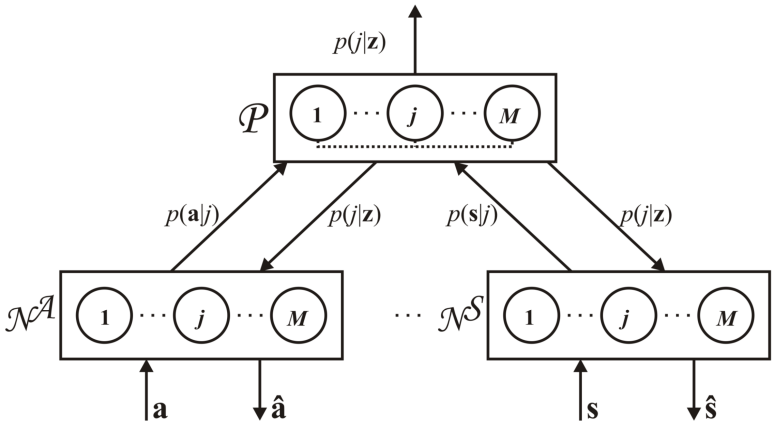

Figure 4.1: Basic architecture of the Incremental Gaussian Mixture Network from Heinen et al. (2011). $\mathcal{N}^{\mathcal{A}}$ to $\mathcal{N}^{\mathcal{S}}$ refer to the coritcal regions and $\mathcal{P}$ is the association region. The cortical regions are only connected via the association region. New inputs are denoted by **a**-**s**.

ance Σ_k of network activations for each network are maintained[16]. Based on these, the likelihood of each component k is can be calculated:

$$p\left(u \mid k\right) = \frac{1}{(2\pi)^{M/2}\sqrt{|\Sigma_k|}} \exp\left(-\frac{1}{2}(u-\mu_k)'\Sigma_k^{-1}(u-\mu_k)\right) \tag{4.16}$$

where M is the size of component k. Given a prior distribution $p(k)$ which is initialized as $p(k) = \frac{1}{K}$ $(k = 1, ..., M)$ posterior probabilites can be calculated for each component

$$p(k \mid u) = \frac{p\left(x \mid k\right) p(j)}{\sum_{k=1}^{K} p\left(x \mid k\right) p(k)} \tag{4.17}$$

The prior distribution $p(k)$ is also updated based on (4.17). The recostruction of a target signal by the network is then produced as a weighted average of the individual regions' predictions

$$\hat{y} = \sum_{k=1}^{K} p(k \mid u)\hat{y}_m \tag{4.18}$$

Where $\hat{y}_m$ is the prediction of the m-th component based on a Gaussian Process Regression. We will refrain from going into more detail, the interested reader is referred to their paper.

The further enhancement of Pinto et al. (2011) to this idea is the preprocessing of the input u by a single Echo State Network and the application of the Incremental Gaussian Mixture Network after the input has been transformed into a higher-dimensional representation.

[16]For details on the exact update of μ_k and Σ_k, the reader is referred to their paper.

4.2.3 Plasticity based Weighting

Both of the approaches outlined above are working with a likelihood based methodology to weight the predictions of a set of models (experts). As opposed to the loss induced weighting of experts, they choose the likelihood as a measure that is not related to the actual prediction error or loss, that the components suffer indivually. Accordingly, their model designs favour such an approach.
To transfer this idea of a likelihood based weighting, the combination of Schrauwen et al. (2008), Kalliovirta et al. (2015) and Pinto et al. (2011) inspired a weighting of experts of echo state networks that is updated prior to the prediction step and is not necessarily dependent on a loss function L.
Given $K \in \mathbb{N}$ experts, each being an Echo State Network, we define $\boldsymbol{\sigma} := (\sigma_1, ..., \sigma_K)$, $\boldsymbol{\mu} := (\mu_1, ..., \mu_K) \in \mathbb{R}^K$. Assume we have tuned each of the networks by the approach of Schrauwen et al. (2008) with a target normal distribution $\mathcal{N}(\mu_k, \sigma_k^2)$ for $k = 1, ..., K$. Associated to these experts is a weight vector $\mathbf{w}_t = \left(w_t^1, w_t^2, ..., w_t^K\right)$ where the index t corresponds to a point in time. After the network has been presented a new input and has been update to reflect the state $\mathbf{x}_t$, the update of $\mathbf{w}_t$ is based on the likelihood. Accordingly, weights are updated in the following way

$$\begin{aligned} \tilde{w}_t^k &= p(x_t^k \,|\, \mu_k, \sigma_k^2) \\ &= w_{t-1}^k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{\left(x_t^k - \mu_k\right)' \left(x_t^k - \mu_k\right)}{2\sigma_k^2} \right) \qquad (4.19) \\ &\quad \text{for } k = 1, ..., K \\ \mathbf{w}_t &= \frac{\tilde{\mathbf{w}}_t}{\sum_{k=1}^K \tilde{w}_t^k} \qquad (4.20) \end{aligned}$$

This makes intuitive sense because the k-th Echo State Neworks has been tuned to follow a normal distribution with mean μ_k and variance σ_k^2 and, thereby, to achieve an information-maximization. Ranking the networks by their states' likelihood is equivalent to using the Kullback-Leibler divergence that has originally been used to move the network towards the targeted distribution. As argued in section 3.5, μ_k can be set to 0 and $p(\cdot \,|\, \mu, \sigma_k^2)$ refers to the density of a normal distribution with the given parameters[17]. Let $N \in \mathbb{N}$ be the number of total neurons aggregated over all experts and $n = \frac{N}{K}$ be the size of one expert, define W_{in}^k, W^k, W_{out}^k and $\mathbf{b}^k$ like in section 3.1 and initialize $\mathbf{x}_0^k = \mathbf{0} \in \mathbb{R}^n$. We can outline the algorithm for the plasticity weighted experts of Echo State Networks

[17] As will further be elaborated on in section 5, numerical accuracy in the calculation of (4.19) can become critical, when the size N of neurons in each expert gets large.

$$\mathbf{x}_t^k = f\left(W^k \mathbf{x}_{t-1}^k + W_{in}^k u_t + \mathbf{b}^k\right) \tag{4.21}$$

$$\tilde{w}_t^k = w_{t-1}^k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{\left(\mathbf{x}_t^k - \mu_k\right)' \left(\mathbf{x}_t^k - \mu_k\right)}{2\sigma_k^2}\right) \tag{4.22}$$

$$\text{for } k = 1, ..., K$$

$$\mathbf{w}_t = \frac{\tilde{\mathbf{w}}_t}{\sum_{k=1}^K \tilde{w}_t^k} \tag{4.23}$$

$$\hat{y}_t = \sum_{k=1}^K w_t^k \mathbf{x}_t^k W_{out}^k \tag{4.24}$$

For the ease of notation, the addition of an intercept or the input u_t into (4.24) has been neglected but surely is possible. Combining this with an online training of the output connections W_{out}^k (see section 2.2) can produce a one-shot prediction cycle then can be run and updated perpetually.

4.2.4 Equivalence of Exponential and Plasticity Weighting

We want to show the equivalence of the loss induced weighting and the proposed plasticity based weighting of experts and try to present the relationship of the two under certain conditions. These assumptions are mostly based on a good adaptation of the reservoir dynamics towards a targeted distribution such that we can infer some statistical features of the network activation which would otherwise not be possible without the pre-training of the reservoir dynamics. The equivalence of the proposed plasticity approach and the exponential weighting of experts can be shown under the following assumptions:

1. The loss function L for the loss induced weighting is chosen to be the *Mean Squared Error* which is $L(x, y) = (x - y)^2$.

2. The output model doesn't include an intercept or the input u. This means, the output of the model is exclusively based on the activation values of the network's neurons.

Using assumption (1) the update of weights for the loss induced weighting (4.11) and the plasticity (4.19) approach already look similar:

$$\tilde{w}_t^k = \tilde{w}_{t-1}^k \exp\left(-\eta \left(\hat{y}_t - y_t\right)^2\right) \tag{4.25}$$

$$\hat{\mathbf{w}}_t^k = \hat{w}_{t-1}^k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{\left(\mathbf{x}_t^k - \mu_k\right)' \left(\mathbf{x}_t^k - \mu_k\right)}{2\sigma_k^2}\right) \tag{4.26}$$

where $\tilde{w}$ and $\hat{w}$ refer to the expert and the plasticity approach, respectively. Choosing $\eta > 0$ separately for each expert k as $\eta_k = \frac{1}{2\sigma_k^2}$ and targeting all of the neuron's

output to $\mu_k = 0$ for all k, the argument of the exponential function is aligned and the plasticity update becomes

$$\hat{w}_t^k = \hat{w}_{t-1}^k \frac{1}{\sqrt{\pi \eta_k}} \exp\left(-\eta_k \left(\mathbf{x}_t^k\right)^2\right) \tag{4.27}$$

where $\left(\mathbf{x}_t^k\right)^2 = \mathbf{x}_t^k \left(\mathbf{x}_t^k\right)'$ is the inner product of $\mathbf{x}_t^k$. In order to finally compare the two approaches, one has to take the stance of a pre-prediction update of the loss induced weighting by the expected error. Kerridge (1967) argues that

$$\hat{y}_t^k - y_t = z_k \sigma_{\varepsilon_k} \left[\left(\mathbf{x}_t^k - \bar{\mathbf{x}}^k\right)' S_k^{-1} \left(\mathbf{x}_t^k - \bar{\mathbf{x}}^k\right) + 1 + \frac{1}{n}\right]^{1/2} \tag{4.28}$$

where S_k is the sample covariance of network activations, a bar denotes the average of past network activations, σ_{ε_k} is the prediction error of the linear model corresponding to expert k and $z_k \sim \mathcal{N}(0,1)$ for $k = 1, ..., K$, all up to time $t-1$. Under ideal conditions, we add the following assumptions:

3. $\bar{\mathbf{x}}^k = 0$ because we have tuned the network activations to follow a normal distribution with mean 0.

4. $S_k = \sigma_k^2 1_n$ is a diagonal matrix with σ_k^2 on its diagonal.

Both of these assumptions are in line with the methodology of the plasticity weighting. Therefore, (**??**) simplifies to

$$\hat{y}_t^k - y_t = z_k \sigma_{\varepsilon_k} \left[\frac{1}{\sigma_k^2} \left(\mathbf{x}_t^k\right)^2 + 1 + \frac{1}{n}\right]^{1/2} \tag{4.29}$$

which implies

$$\left(\hat{y}_t^k - y_t\right)^2 = z_k^2 \sigma_{\varepsilon_k}^2 \left[\frac{1}{\sigma_k^2} \left(\mathbf{x}_t^k\right)^2 + 1 + \frac{1}{n}\right] \tag{4.30}$$

Finally, we can conclude

$$\begin{aligned}
\mathbb{E}\left[\left(\hat{y}_t^k - y_t\right)^2\right] &= \mathbb{E}\left[z_k^2\right] \sigma_{\varepsilon_k}^2 \left[\frac{1}{\sigma_k^2} \left(\mathbf{x}_t^k\right)^2 + 1 + \frac{1}{n}\right] && (4.31)\\
&= \sigma_{\varepsilon_k}^2 \left[\frac{1}{\sigma_k^2} \left(\mathbf{x}_t^k\right)^2 + 1 + \frac{1}{n}\right] && (4.32)\\
&= \frac{\sigma_{\varepsilon_k}^2}{\sigma_k^2} \left(\mathbf{x}_t^k\right)^2 + \sigma_{\varepsilon_k}^2 \left(1 + \frac{1}{n}\right) && (4.33)
\end{aligned}$$

because $z_k^2 \sim \chi_1^2$ and $\mathbb{E}\left[z_k^2\right] = 1$. So we find a direct relationship between the expected loss $\mathbb{E}\left[\left(\hat{y}_t^k - y_t\right)^2\right]$ and the network activation $\mathbf{x}_t^k$.

Substituting the loss in (4.25) for its expected value, and using (4.33) yields

$$\begin{aligned}
\tilde{w}_t^k &= \tilde{w}_{t-1}^k \exp\left(-\eta \mathbb{E}\left[\left(\hat{y}_t^k - y_t\right)^2\right]\right) && (4.34)\\
&= \tilde{w}_{t-1}^k \exp\left(-\eta\left(\frac{\sigma_{\varepsilon_k}^2}{\sigma_k^2}\left(\mathbf{x}_t^k\right)^2 + \sigma_{\varepsilon_k}^2\left(1+\frac{1}{n}\right)\right)\right) && (4.35)\\
&= \tilde{w}_{t-1}^k \exp\left(-\eta\frac{\sigma_{\varepsilon_k}^2}{\sigma_k^2}\left(\mathbf{x}_t^k\right)^2\right) \underbrace{\exp\left(-\eta\sigma_{\varepsilon_k}^2\left(1+\frac{1}{n}\right)\right)}_{=:\zeta} && (4.36)
\end{aligned}$$

with a constant factor ζ that doesn't influence the effective weighting. So the updates of weights can be regarded as equivalent up to some difference in the learning rate. Even this difference can be overcome by choosing η of the experts approach appropriately as $\eta = \frac{\sigma_{\varepsilon_k}^2}{2}$ for the equivalence of (4.22) and (4.36) up to some constant factor.

5 Application

The statistical characteristics of financial asset returns play a crucial role in today's economy. Risk assessment has become ever increasingly important given not only the recent history of financial markets[18]. The financial crisis of 2007/2008, the burst of the 'dot com' bubble in the year 2000 and many other examples have shown that the financial market can cause major distress in banks, companies and for individual investors. The modeling of the first moment of the return distribution is important but for risk assessment, the most critical feature of the return distribution is its second moment, the variance, which has spurred a wide literature on forecasting and modeling of asset return characteristics. The following builds the foundation for the application of our model(s) to real world data.

5.1 Mathematical Foundations

The return process of a financial asset can be discribed by

$$r_{t+h} = P_{t+h} - P_t = \log\left(\tilde{P}_{t+h}\right) - \log\left(\tilde{P}_t\right) \tag{5.1}$$

where $\tilde{P}_t$ is the price process of the asset. In the seminal work of Andersen et al. (2001) and Andersen et al. (2003) they define the one-dimensional stochastic (Itô) process

$$P_{t+h} - P_t = \int_t^{t+h} \mu_s \,\mathrm{d}s + \int_t^{t+h} \sigma_s \,\mathrm{d}B_s \tag{5.2}$$

with a standard Brownian motion B_t, μ_s is a locally bounded and predictable dift process and σ_s is the right-continuous and left-limit volatility. Given a sequence of

[18] US-China trade war, corona virus outbreak, ...

partitions $(D_m)_{m\in\mathbb{N}}$ with $D_m = \{0 = t_1 < ... < t_m = T\}$ with $\sup_{1\leq i\leq m} |t_i - t_{i-1}| \to 0$ for $m \to \infty$, the quadratic variation $\langle P\rangle_t$ of r_{t+h} is defined as

$$\langle P\rangle_t = \lim_{m\to\infty} \sum_{\substack{t_i\in D_m \\ t_i<t}} \left(P_{\min(t_{i+1},t)} - P_{t_i}\right)^2 \tag{5.3}$$

which exists in probability. Billingsley (2008) shows that

$$\langle P\rangle_t = \int_0^T \sigma_s^2 \,\mathrm{d}s \tag{5.4}$$

which is known as integrated variance.

5.2 Realized Volatility

Barndorff-Nielsen and Shephard (2004) were the first to introduce a consistent estimator for the integrated variance (5.4)[19] under the assumption of no market microstructure noise[20]. Given the log-price process introduced earlier, they propose the Realized Volatility[21] $RV(0,t)$ as an estimator of the square root of the integrated variance by

$$RV(0,t) = \sqrt{\sum_{j=1}^{J} \left(P_{t_j} - P_{t_{j-1}}\right)^2} = \sqrt{\sum_{j=1}^{J} r_{t_j}^2} \tag{5.5}$$

given a partition $D_J = \{0 = t_0 < t_1 < ... < t_J = t\}$ for the period $[0,t]$. The defintion (5.3) implies the consistency of $RV(0,t)$, more precisely

$$\operatorname{p\,lim}_{J\to\infty} RV(0,t)^2 = \int_0^t \sigma_s^2 \,\mathrm{d}s \tag{5.6}$$

The assumption of no market microstructure doesn't hold in reality due to multiple reasons. Firstly, the price of an asset is not continous but discrete with bounces between the bid-price and the ask-price. Secondly, the observation of prices is not continous but trades and as a consequence the new price happen irregularly. However, by design the estimator is positive and a compromise has to be found between market microstructure noise and consistency of the estimator. Andersen et al. (2011) give a detailed analysis of the effects of market microstructure noise on the estimation of realized volatility.

[19] They even propose an estimator for integrated covariance.

[20] Market microstructure is concerned with the formation of the price and the 'noise' around it which arises from timing of transactions, flow and disclosure of information and the individual behaviour of market participants.

[21] The original papers were concerned with the Realized Variance but we will denote by RV the Realized Volatility, so the square root of the Realized Variance.

5.3 Heterogeneous Autoregressive Model

One of the most prominent models for the forecasting of realized volatility is the *Heterogeneous Autoregressive model of Realized Volatility* (HAR) by Corsi (2009). He proposes an additive cascade model of volatility based on different time periods. It is a linear model and convices by its simplicity and its ability to reproducing the main empirical phenomena of financial returns (long memory, fat tails and self-similarity). It is often considered as a benchmark for the prediction of realized volatility and has proven itself to be hard to beat. Corsi (2009) extends the daily realized volatility introduced in section 5.2 (now denoted by $RV_t^{(d)}$) to different time horizons longer than one day. By taking a simple average of the daily quantities he presents for example

$$RV_t^{(w)} = \frac{1}{5}\left(RV_t^{(d)} + RV_{t-1}^{(d)} + \ldots + RV_{t-4}^{(d)}\right) \tag{5.7}$$

to define a weekly realized volatility. Accordingly, he uses a average over the last 22 days to define a monthly realized volatility $RV_t^{(m)} = \frac{1}{22}\sum_{i=0}^{21} RV_{t-i}^{(d)}$. The superscripts (d), (w) and (m) refer to the time periods and subscript t denotes the time. Corsi (2009) mentions that the aggregate volatility over long time frames, as presented here, doesn't exactly represent the realized volatility over othe the specified time interval. The differences, he states, are immaterial and enable the interpretation of the model as a restricted AR(22) model of daily realized volatility. The specific choice of those time horizons can be motivated by dissimilarities in the approaches participants in financial markets. Endowments or pension funds - trading much less frequently - target a longer time horizon of their investments[22], medium-term investors may rebalance their portfolios on a weekly basis, have different risk profiles and analyse markets on shorter timeframes, and day traders (as the name suggest), market makers or dealers look at daily timeframes. The overall emerging pattern is a volatility cascade from monthly (low) to daily (high) frequencies that builds the basis for a model with three heterogeneous volatility components. Obviously, more components (quarterly or even yearly) could be added. Corsi (2009) defines the *latent partial volatility* $\tilde{\sigma}_t^{(\cdot)}$ for three volatility components: one day (d), one week (w) and one month (m). He argues that each component has an 'almost AR(1)' structure and that the shorter-term processes are influenced by the longer-term components to finally present:

[22]Mostly even longer than one month.

$$\tilde{\sigma}^{(m)}_{t+1m} = c^{(m)} + \phi^{(m)} RV^{(m)}_t + \tilde{\omega}^{(m)}_{t+1m} \tag{5.8}$$
$$\tilde{\sigma}^{(w)}_{t+1w} = c^{(w)} + \phi^{(w)} RV^{(w)}_t + \gamma^{(w)} \mathbb{E}\left[\tilde{\sigma}^{(m)}_{t+1m}\right] + \tilde{\omega}^{(w)}_{t+1w} \tag{5.9}$$
$$\tilde{\sigma}^{(d)}_{t+1d} = c^{(d)} + \phi^{(d)} RV^{(d)}_t + \gamma^{(d)} \mathbb{E}\left[\tilde{\sigma}^{(w)}_{t+1w}\right] + \tilde{\omega}^{(d)}_{t+1d} \tag{5.10}$$
$$\tag{5.11}$$

where $RV^{(m)}$, $RV^{(w)}$ and $RV^{(d)}$ are the (ex post) observed realized volatilities. The innovations $\tilde{\omega}^{(m)}_{t+1m}$, $\tilde{\omega}^{(w)}_{t+1w}$ and $\tilde{\omega}^{(d)}_{t+1d}$ are independent and appropriately left truncated error terms to ensure positiveness of the left-hand sides. Substitution of the partial processes of longer-term components into the daily component finally yields

$$RV^{(d)}_{t+1d} = c + \beta^{(d)} RV^{(d)}_t + \beta^{(w)} RV^{(w)}_t + \beta^{(m)} RV^{(m)}_t + \omega_{t+1d} \tag{5.12}$$

A major drawback of (5.12) is the fact, that the model doesn't guarantee positiveness of the RV forecast. In order to remedy that, Corsi (2009) suggests to work with the log-transformation of the realized volatility which practitioners almost always do.

5.4 Methodology

The HAR model by Corsi (2009) has the advantage that the model doesn't have any hyperparameters. One could use regularization techniques that have been presented in section 2, however due to the low dimensionality (in the general case, it only uses 3 regressors and an intercept) the possibility of overfitting decreases as the amount of available data increases. With Echo State Networks that have multiple hyperparameters, the question arises on how to optimize those in order get decent predictive performance. There are many strategies and

5.4.1 Hyperparameter Optimization

The choice of hyperparameters is a crucial step in the training of many machine learning models and has to be performed with due diligence. In order to train our models and validate their hyperparameters, we are going to use k-fold cross-validation which has already been mentioned in section 2. Cross-validation describes the procedure of training a model on a subset of the available training data, which we will refer to as *estimation set*, and validate its performance on another set of the data called *validation set*, which hasn't been presented to the model in the estimation step. This simulates the situation when non-available data will be presented to the model in the future. Based on the error in the validation set, the hyperparameters of the model can be chosen in a reasonable way. Many procedures have been presented that help to get a good representation of the performance of a model on unseen

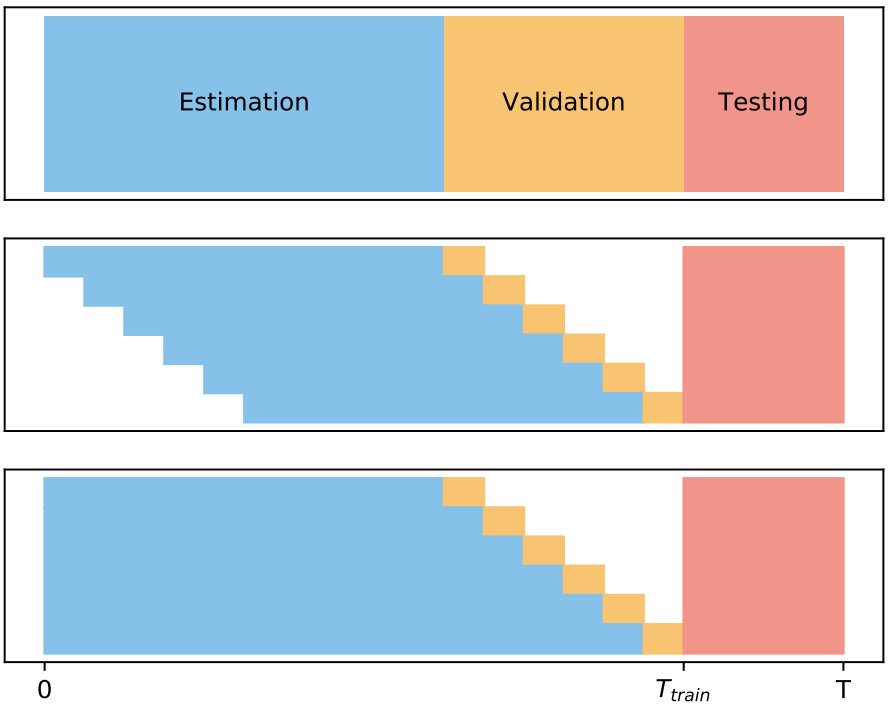

Figure 5.1: The top panel describes the split of the avaliable data into an estimation and validation (which together is the training set) and a testing set, that is used to compare different models. The split was chosen 50%, 30% and 20%, respectively. The center panel presents the rolling window approach and the expanding window approach is in the lower panel, both for a 6-fold cross-validation technique that corresponds to a 5% validation for each validation and a total of 30% for the whole validation set. In the rolling window approach the size of the training set stays constant whereas in the expanding window approach, the effective training sample size increases.

data, and we want to focus on two of those, namely a rolling window approach or an expanding window approach, which are presented in figure (5.1).
There are other approaches of cross-validation like random sampling or leave-k-out cross validation but those do not take into account the temporal nature of the data (Hastie et al., 2001). It would be unreasonable to estimate a temporal model on a given timeframe and validate its performance on data that preceeds the training set because there are dependencies over time and it would not be a realistic setting for making predictions into the future. The rolling window and expanding window approach, however, take the time series setting into account and the training set temporally always preceeds the validation set. In the rolling window approach, the size of the estimation and validation set stays the same, because both are shifted over time. The expanding window on the other hand only has a constant validation set but the estimation set is increasing. This poses the challenge of inconsistent

validation errors, because increasing the training has an impact on the quality of the estimation and therefore influence the validation error[23]. After cross validation has been performed and a set of hyperparameters has been fixed, the model can then be re-estimated on the whole training set to achieve the best estimation possible.
For the case of Echo State networks the hyperparameter optimization is performed in two steps because of the temporal and iterative nature of the network. When W_{out} is estimated 'offline', the network activations x_t have to be gathered in a matrix and this can only be done after the hyperparameters of the network, namely the spectral radius ρ and the bias $\mathbf{b}$, have been fixed. Based on the network activations, the regularization strength λ of for example a ridge regression can be chosen accordingly. The approach to the search for the best hyperparameters can be performed by simple grid search over a predefined set of hyperparameters. Alternatively, there are many computational packages that provide more sophisticated approaches like particle swarm optimization (Basterrech et al., 2015) or gradient descent (Jaeger et al., 2007) as already mentioned in section 3.3. For our empirical application we are going to use a *gradient boosted regression tree* as part of a powerful Python packaged called *scikit-optimize*[24]. The theory of regression trees and how to boost them are provided in Hastie et al. (2001).

5.5 Forecasting

The models - both Echo State Networks and HAR - are trained to perform 1-step ahead predictions, this means $u_t = By_t$ for a times series $(y_t)_{t\in\mathbb{Z}}$. In order to come up with mutli-step ahead forecasts of horizon $h > 1$, we let the model produce a 1-step ahead forcast and feed the value back to the network as a new input. Assume, we have trained our model using data $(u_t)_{t\leq T_{\text{train}}}$ and we produce a 1-step ahead forecast $\hat{y}_{T_{\text{train}}+1}$. By concatenation and without retraining the model, we let it produce another 1-step ahead forecast based on $(..., u_{T-1}, u_T, \hat{u}_{T_{\text{train}}+1})$. In an iterative way, this can produce an h-step ahead forecast $\hat{u}_{T+h}$. One phenomenon that hasn't been researched extensively is the fact that injecting noise into the system while performing a multi-step prediction can improve performance. Jaeger (2002) mentions that the injection of noise while updating the network can improve network stability. This can also be used in multi-step predictions. During the training of an Echo State Network, or more specifically in the training of W_{out}, you commit an error in the estimation. Using the residuals, which we will refer to as $e_t = y_t - W_{out}x_t$, $t = 1, ..., T_{\text{train}}$, of the estimation of what boils down to a linear regression, one can improve the performance by adding random draws of $\{e_1, ..., e_{T_{\text{train}}}\}$. So instead of using the 1-step ahead prediction $\hat{y}_{T_{\text{train}}+1}$, you disturb the prediction and feed

[23]This can come in both forms, better estimation which would decrease the validation error or overfitting which would increase the validation error.

[24]The package hasn't been maintained for quite some time, but has been taken over in January 2020. https://scikit-optimize.github.io/stable/

$\tilde{u}_{T+1} = \hat{u}_{T+1} + e_j$ to the network for a random draw $j \in \{1, ..., T_{\text{train}}\}$. Finally, taking a monte carlo approach by producing multiple paths of $(\hat{y}_{T_{\text{train}}+1}, ..., \hat{y}_{T_{\text{train}}+h})$ and averaging over those paths, the performance can be improved.

5.5.1 Fixed Training Set

We will not let the models predict the whole testing set in one go by the methods mentioned above, our testing set will be of size 1000, but instead are going to take an iterative approach. As for the linear regression in the HAR model and the estimation of W_{out} in the Echo State Networks, we will use fixed training set (consisting of estimation and validation) to estimate a cross-validated set of hyperparameters and regression parameters and those will not be changed over the course of the prediction. Given a forecasting horizon $h \in \{1, 2, 5, 10\}$, we will start by letting the models predict $(\hat{y}_{T_{\text{train}}+1}, ..., \hat{y}_{T_{\text{train}}+h})$ as a set of iterated 1-step ahead forecasts if $h > 1$. We will then update the models by presenting them the true values $(y_{T_{\text{train}}+1}, ..., y_{T_{\text{train}}+h})$. Using the true data we will update the network state (for ESNs) and, in case of the experts, their weighting based on the associated loss over the period $[T_{\text{train}} + 1, T_{\text{train}} + h]$. For the HAR model, we will update the regressors. For the plasticity, we are in fact able to update the weighting in a multistep prediction because the network state and its likelihood can be measured, independent of whether the true data or self-produced prediction has been fed or the network has been presented the true data. Once, the true data has been presented to the *plasticity experts*, the network's reaction to its own predictions is rectified by the true data. Accordingly, the procedure then continues to produce forecasts $(\hat{y}_{T_{\text{train}}+h+1}, ..., \hat{y}_{T_{\text{train}}+2h})$. This way, we can predict up to = 1000. For both approaches we consider the mean error and the mean likelihood over the previous h steps to update the weights in the multi-step prediction settings.

5.5.2 Rolling Training Set

In contrast to the fixed training set approach, an alternative way of producing forecasts of the testing set is to re-estimate the models after every h-step ahead prediction. Since the optimization of hyperparameters weighs heavily on the training time, you usually stick to one set of hyperparameters that has been selected based on the training set. This makes a diligent selection of those parameters even more important. Keeping the hyperparameters, we will however update the regression coefficients, namely W_{out} for ESNs and $\boldsymbol{\beta}$ for the HAR model. This way, we are able to account for regime changes in the time series, which are assumed to have no material effect on the set of hyperparameters. For this approach, one can choose either the rolling window approach or the expanding window approach, now in relation to forecasting instead of hyperparameters optimization, as presented in figure 5.1.

6 Empirical Results

For the empirical application of the proposed model, we investigate the evolution of the IBM stock price. The data comes from the TAQ database of the New York Stock Exchange in the period from January 2nd, 2001 to December 30th, 2018. The price is sampled in intervals of 5 minutes and contains 4444 trading days, excluding holidays and weekends. The time series for our application will be the daily realized volatility introduced in section 5.2. In order to compare the models that come from the Reservoir Computing field with the HAR model by Corsi (2009), we take the log transformation of the daily realized volatility. So our time series becomes

$$u_t = \log\left(\sqrt{RVD_t}\right) = \log\left(\sqrt{\sum_{j=1}^{78} r_{t_j}^2}\right) \tag{6.1}$$

where r_{t_j} is referring to the j-th 5-minute return on day t. The core trading hours on the New York Stock Exchange are from 9:30 am to 4:00 pm[25], which amounts to 6.5 hours or 87 intervals of 5 minutes.

Figure 6.1 presents the evolution of the realized volatility of IBM in the given timeframe. The left panel is referring to the original series of σ_t and the right panel is its log transformation $\log(\sigma_t)$.

Especially in the early years of the 21st century, the aftermath of the burtst of the 'dot com' bubble are still present in the timeseries which IBM as a technology company was probably more affected than other companies as well as the effect of the global financial crisis of 2007/2008 where this specific company was less affected by than in the early years of the century. Calmer periods of low volatility can be seen in the years between 2004 and 2006 and between late 2012 and late 2015 which underlines the presence of clustering.

Apart from the idiosyncratic risks associated to IBM, the evolution shows the typical patterns of volatility in financial time series, namely clustering and high persistence. Clustering is describing the pheonomenon of periods of high volatility being followed by periods of low volatility. High persistence is referring to a high level of autocorraltion which is presented in figure 6.2. The blue shaded area covers the range of significant autocorrelation and is touched for the first time at approximately the 140-th lag. This corresponds to more than 6 months.

In what follows, we will refer to the loss induces experts as *loss experts* and to the plasticity weighted experts as *plasticity experts*.

[25] https://www.nyse.com/markets/hours-calendars

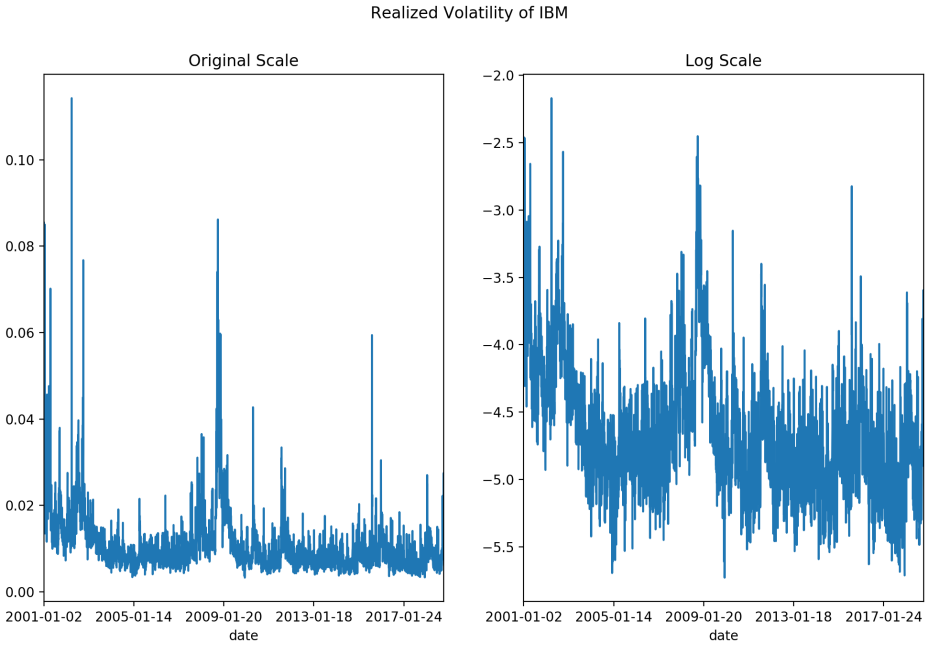

Figure 6.1: Realized volatility of IBM for the period from January 2nd, 2001 to December 30th, 2018. The left hand side presents the realized volatility in its original scale, the right hand side presents the log transformation of the time series.

6.1 Analysis of Network Behaviour

In this section we want to shed some more light of the network behaviour for reservoirs that have been pre-trained using the intrinsic plasticity approach from section 3.5 using the IBM realized volatility as input u_t. Figure 6.3 presents the network activations for a givent time period and compares two networks that have been pre-trained towards different normal distributions. Networks of size $n = 200$ were used and the network has been presented the whole time series of realized volatilities for 100 epochs of adaptation. Given the *tanh* activation function as well as the scaling of the input series to $[-0.8, 0.8]$ support the centering of network activations around 0 which is in line with setting the mean $\mu = 0$ that has been argued about in section 4. The network behaviour for the left panel, which has been targeted towards $\mathcal{N}(0, 0.16)$, barely manages to put the neurons' output inside the 95% confidence bounds of the respective normal distribution. In periods of relatively low volatility, e.g. between step 10 and 20 or around step 90, the network is able to fit the normal distribution relatively well. This compares to the network in the right panel, where the network has been trained to match a normal distribution $\mathcal{N}(0, 1)$. Here the adaptation barely manages to fit the neurons activations outside the confidence bounds[26].

Additionally, figure 6.4 presents the evolution of the networks likelihood in relation-

[26]Given the size of the reservoir, approximately $0.05 \cdot 200 = 10$ activations should lie outside the confidence bounds

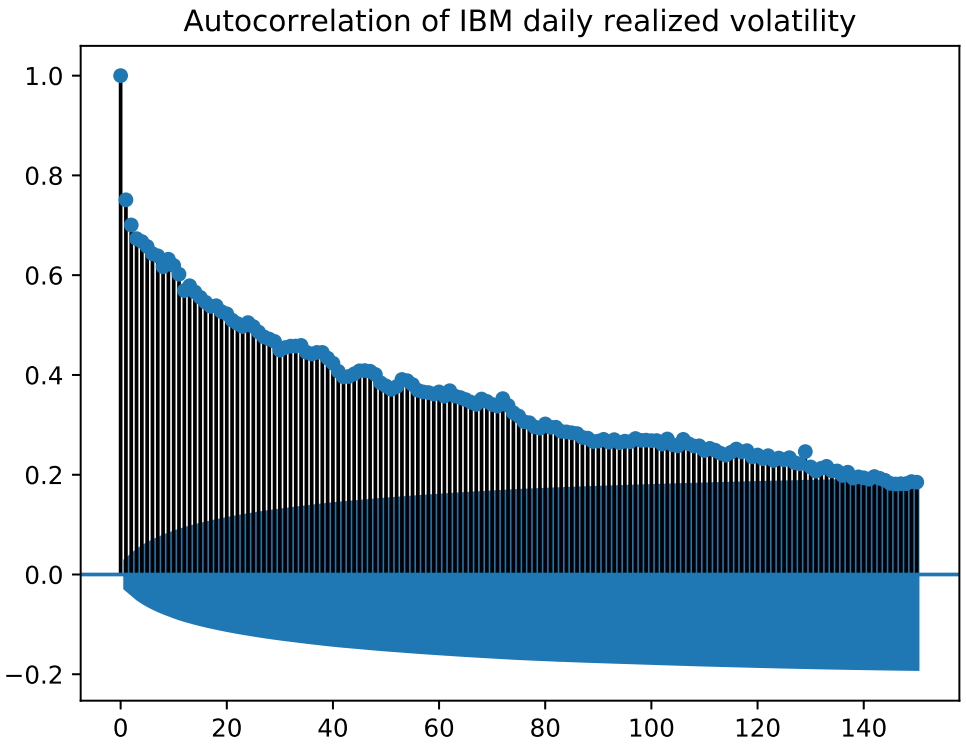

Figure 6.2: Autocorrelation of daily realized volatility of IBM for the period from January 2nd, 2010 to December 30th, 2018. The blue shaded area are the bounds for significance which is touched for the first time at approximately the 140-th lag which shows the long persistence in the series.

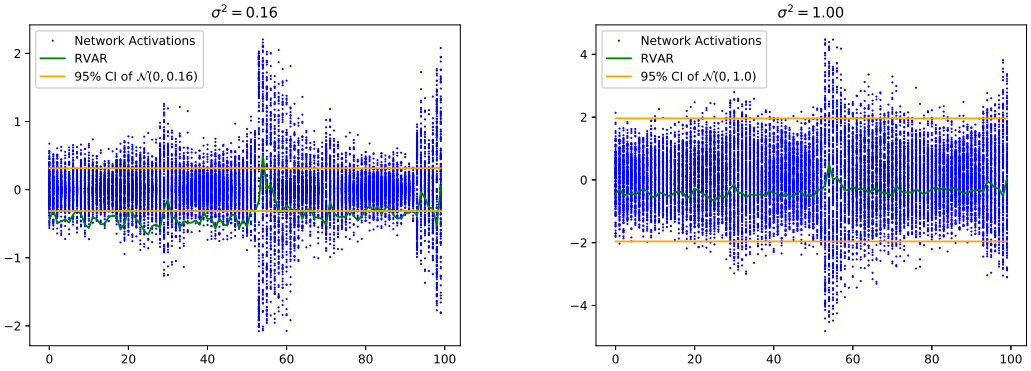

Figure 6.3: Comparison of the activations for two different plasticity tuned networks. One has been tuned for $\sigma = \frac{1}{\sqrt{2\pi}} = 0.16$ which we will use later in the empirical application. As a comparison, a network with $\sigma = 1.0$ is presented. The horizontal lines are the 95% confidence bounds to give an idea of the range of a normal distribution $\mathcal{N}(0, \sigma^2)$ with the corresponding parameters. The time series presented is from 10th of July to 30th of October 2015. It has been rescaled to the interval $[-0.8, 0.8]$ and was chosen for representative purposes only. Comparing the two, one can see that the extreme values of the timeseries around step 55 make the network with $\sigma = 0.16$ very unlikely given the amount of activations outside the confidence bounds. The effect of this move on the network with $\sigma = 1.0$ is much less significant when considering the comparably fewer activations outside the bounds. Intuitively, this makes sense because the higher volatility move in the time series would favor the network which allows for and targets a wider range of activations.

	initial transient	training	validation	testing
ESN	444	2700	300	1000
HAR	3144			

Table 6.1: Split of the timeseries into different parts during the empirical application. The echo state network approaches have an initial transient to wash out the initialization of the network state which - for the convenience of the other sets - was chosen as 444. As the HAR model doesn't need this transient, those additional days are attributed to the training set which results in a larger training set of 3144 days.

ship to the timeseries of a tuned network, in this case targeted to $\mathcal{N}(0, 1)$. The likelihood, as the series before presenting it to the network, has been rescaled to the interval $[-0.8, 0.8]$. The relationship between the series and the likelihood is clearly negative, the correlation coefficient is -0.59. Whenever the series is spiking, the likelihood is negatively impacted. This is in line with the analysis of figure 6.3 which shows the same relationship of spikes and their impact on the likelihood. Given that the networks have been adapted to a series that rarely exhibits spikes, it is to be expected that all networks, independent of the targeted output distribution, show a negative correlation between the likelihood and the series itself. This, however, compares directly to the *loss experts* approach where each expert is suffering an individual loss and their weighting is based on the relative loss and not the absolute loss. Therefore, the *plasticity experts* are based on their relative likelihood and some network may have a stronger (negative) impact on the likelihood compared to another networks. We expect the networks with an output distribution of higher volatility to favor instances where the time series experiences spikes and output distributions with comparatively lower volatility to favor more supressed regimes of daily realized volatility.

6.2 Predictive Performance

This section presents the empirical forecasting performance of the plasiticity approach in comparison to benchmark tasks including the HAR model by Corsi (2009), the exponential weighted experts of Echo State Network and a single Echo State Network with equivalent size and hyperparameter optimization that has been performed for the spectral radius ρ and bias $\mathbf{b}$.

The timeseries has been divided according to table 6.2. The initial transient is referring to the wash-out period of the Echo State Networks to reduce the effect of the 0-initialization of the network state which effectively decreases the training size. This, coming as a tradeoff compared to the HAR model, decreased the amount of data for the estimation because the the validation set was kept the same for the models. The size of the test set, , was set to 1000 which corresponds to roughly 4 years of daily data.

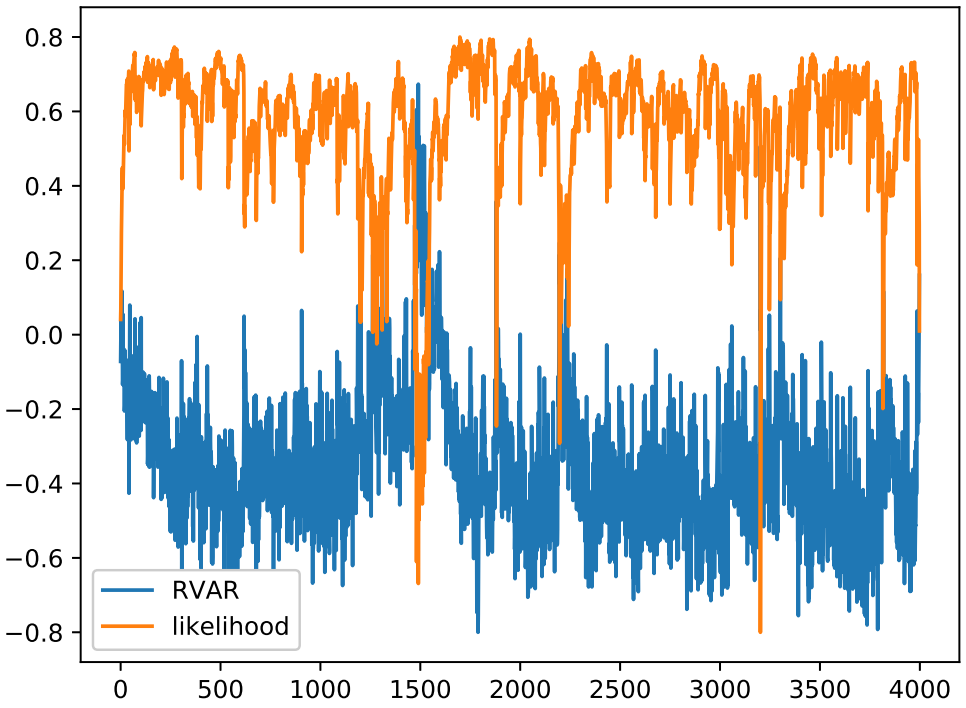

Figure 6.4: The network activation and the corresponding likelihood of the activation for the pre-trained network with a target $\sigma = 1.0$. The correlation of the two series is -0.59. The likelihood has been rescaled to the same interval of $[-0.8, 0.8]$ as the series itself. One can clearly see the negative relationship of likelihood and the spikes in the series. Under *normal* conditions, this means in times of low volatility, the likelihood is larger than at times where there is a spike in the series. This is to be expected because spikes are comparably rare and the network has mostly adapted to these low volatility regimes. The edge of the *plasticity experts*, as with the *loss experts*, is expected to be found in the relative likelihood and the networks' reaction to the spikes in the series and their resulting likelihoods.

6.2.1 Model Specifications

Applying the new model, we will compare the following specifications and comparators:

- HAR model by Corsi (2009)
- ESN experts with loss functions:
  - $L(\hat{u}_t, u_t) = (\hat{u}_t - u_t)^2$
  - $L(\hat{u}_t, u_t) = (\exp(\hat{u}_t) - \exp(u_t))^2$
  - $L(\hat{u}_t, u_t) = \frac{\exp(u_t)^2}{\exp(\hat{u}_t)^2} - \frac{u_t^2}{\hat{u}_t^2} - 1$

  in the exponential update of weights 4.11
- Plasticity weighted experts with
  - $\sigma_k = \frac{1}{\sqrt{2\pi}}$ for all $k = 1, ..., K$
  - An equally spaced grid of $\sigma_k \in \left[0.8\,(2\pi)^{-\frac{1}{2}}, 1.2\,(2\pi)^{-\frac{1}{2}}\right]$ around the constant value $\frac{1}{\sqrt{2\pi}}$
- Single ESN of comparable size where the hyperparemeter optimzation of the spectral radius ρ and bias b and their validation has been performed using tge same error functions as the experts approach

For the loss functions of the loss experts, one has to keep in mind the transformation $u_t = \log\left(\sqrt{\sigma_t^2}\right)$ of the timeseries by the square root and the natural logarithm. Hence, the three loss functions correspond to the differnt scales of the series, (i) the logarithm of the volatility $\log(\sigma)$, (ii) the volatility σ and (iii) the variance σ^2. The choice of $\sigma_k \equiv \frac{1}{\sqrt{2\pi}}$ is motivated by the plasticity update (4.19), so that the additional factor of $\frac{1}{\sqrt{2\pi\sigma_k^2}} = 1$. All the specifications are trained using the online learning of equations (2.11) - (2.12). The choice of the range of spectral radii for the loss experts was motivated by the correspondence to the effective spectral radius of the plasticity experts which are in the range of $[0.8, 1.2]$. This is also supported by the fact that the single ESN with optimized hyperparameters was using a spectral radius of approximately $\rho \in [1.20, 1.25]$, depending on the loss function. Even though, we showed some relationship between the two approaches in section 4.2.4, a fair way of comparison still has to be found.
The following results are based on $K = 10$ experts, total number of neurons $N = 1000$, bias equal to 0.2. In order to compare the models it is crucial, that the random number generator is fixed for all the applications. When this is neglected, by the nature of the reservoir computing models and their random initalization, the comparison of predictive performance may be due to the state of the random number generator when initializing the random connections and can actually not be

		spectral radius	bias	regularization
Single	logMSE	1.0107	0.4507	20.7
	MSE	0.8142	0.3711	20.7
	QLIKE	0.8263	0.8414	6.2
Loss	logMSE	$[0.2, 2.0]$	0.2	$[0.5, 1.8]$
	MSE			$[0.0003, 1.8]$
	QLIKE			$[0.0003, 0.5]$
Plast	Constant	$[0.32, 1.51]$	$[-31.2, 73.4]$	$[0.5, 1.8]$
	Grid	$[0.31, 1.36]$	$[-33.8, 67.0]$	$[0.5, 1.8]$

Table 6.2: Selected hyperparameters of the models based on the training dataset. For the plasticity experts, the spectral radius and the bias are referring to the effective quantities of the network dynamics after tuning them by the plasticity approach.

attributed to the model design. This way, it can be guaranteed that the differences in performance can be attributed to the different weighting approaches and model specifications. Beyond the random initialization, the networks of the *plasticity experts* have been adapted to their targeted output distributions using 50 epochs of presenting them the whole training data set.
The original scale of daily realized volatility was $[0.0032, 0.1142]$ which was log transformed to $[-5.7300, -2.1691]$. For the reservoir computing models, it was finally rescaled to $[-0.8, 0.8]$.
The selected hyperparameters are presented in table 6.2.1. The highest and lowest values of the spectral radius and bias have been realized for the same networks, which means that there are some extreme networks and some less extremely modified network dynamics. The strong regularization in the Single ESN is to be expected given the larger size of $N = 1000$ neurons, compared to the experts that are only of size $N = 100$, but the regularization of the QLIKE is comparably low at 6.2 but may be attributed to the different combination of a lower spectral radius and a higher bias.

6.2.2 Fixed Training Performance

This section presents the out-of-sample predictive performance of the models referring to section 5.5.1 where the models have only been trained on the training set, which is up to $T_{\text{train}} = 3444$, and have not been re-estimated for the predictions.
The figures 6.2.2 to 6.11 present the predictiven performance of the different models and, in case of the expert methods, the distribution of their weightings over the course of the out-of-sample prediction horizon. The *loss experts* show a very consistent behaviour for all the error measurements in regards of their weights distribution. There are minor differences but the overall picture of the evolution of the weights stays the same for all apporaches which comes back to the fact that the underlying

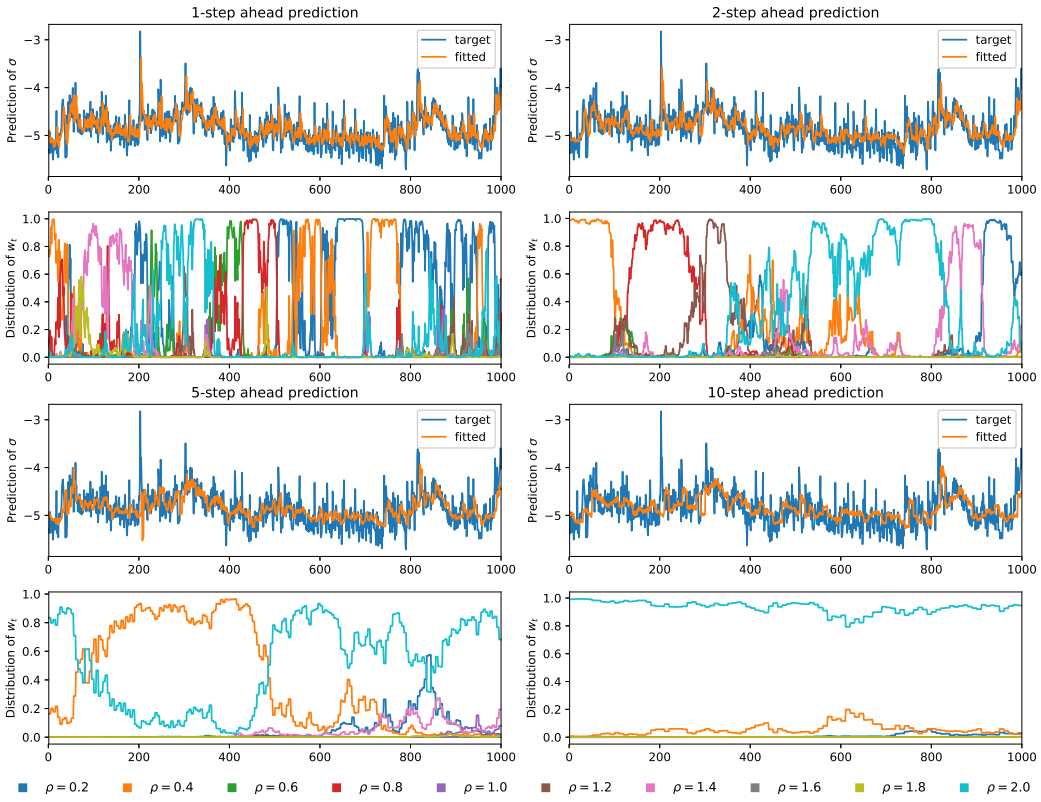

Figure 6.5: This is the *loss experts* approach with exponential weighting of the experts based on the mean squared error of the log transformation $\log(\sigma)$. The values of spectral radii are equally space in the interval $\rho \in [0.2, 2.0]$. The networks have only been trained once on the training set as outlined in section 5.5.1.

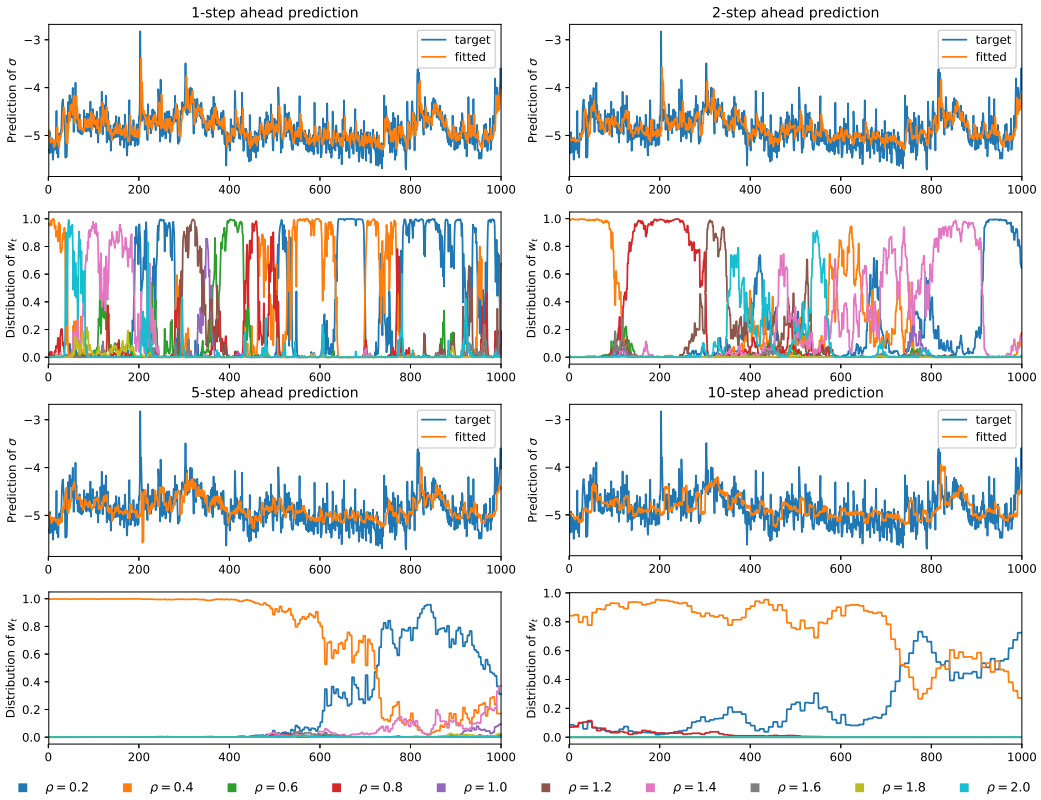

Figure 6.6: This is the *loss experts* approach with exponential weighting of the experts based on the mean squared error of the original time series, so σ. The values of spectral radii are equally space in the interval $\rho \in [0.2, 2.0]$. The networks have only been trained once on the training set as outlined in section 5.5.1.

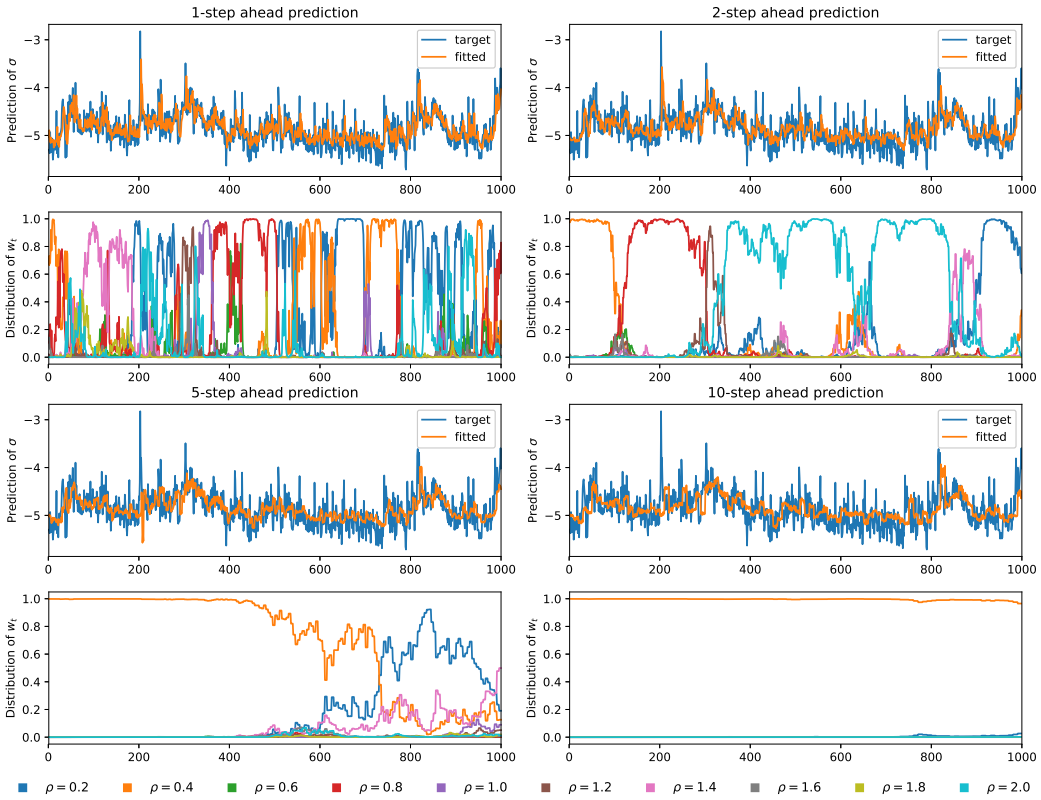

Figure 6.7: This is the *loss experts* approach with exponential weighting of the experts based on the QLIKE loss function, so in the scale of σ^2. The values of spectral radii are equally space in the interval $\rho \in [0.2, 2.0]$. The networks have only been trained once on the training set as outlined in section 5.5.1.

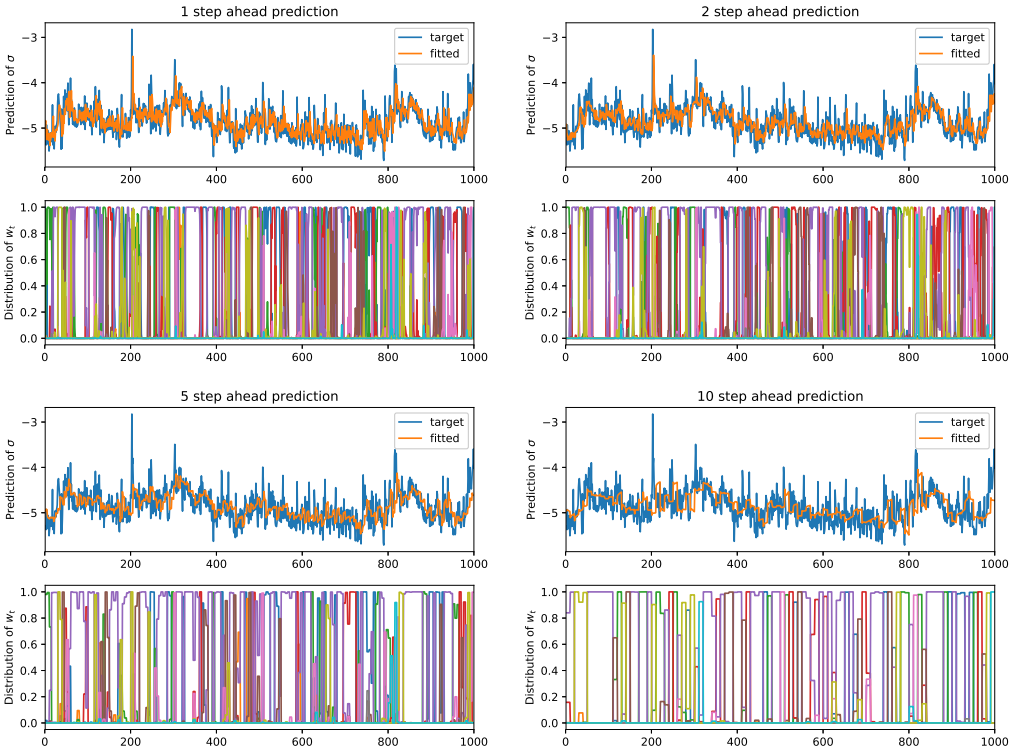

Figure 6.8: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using a constant $\sigma = \frac{1}{\sqrt{2\pi}}$ for all networks. The networks have only been trained once on the training set as outlined in section 5.5.1.

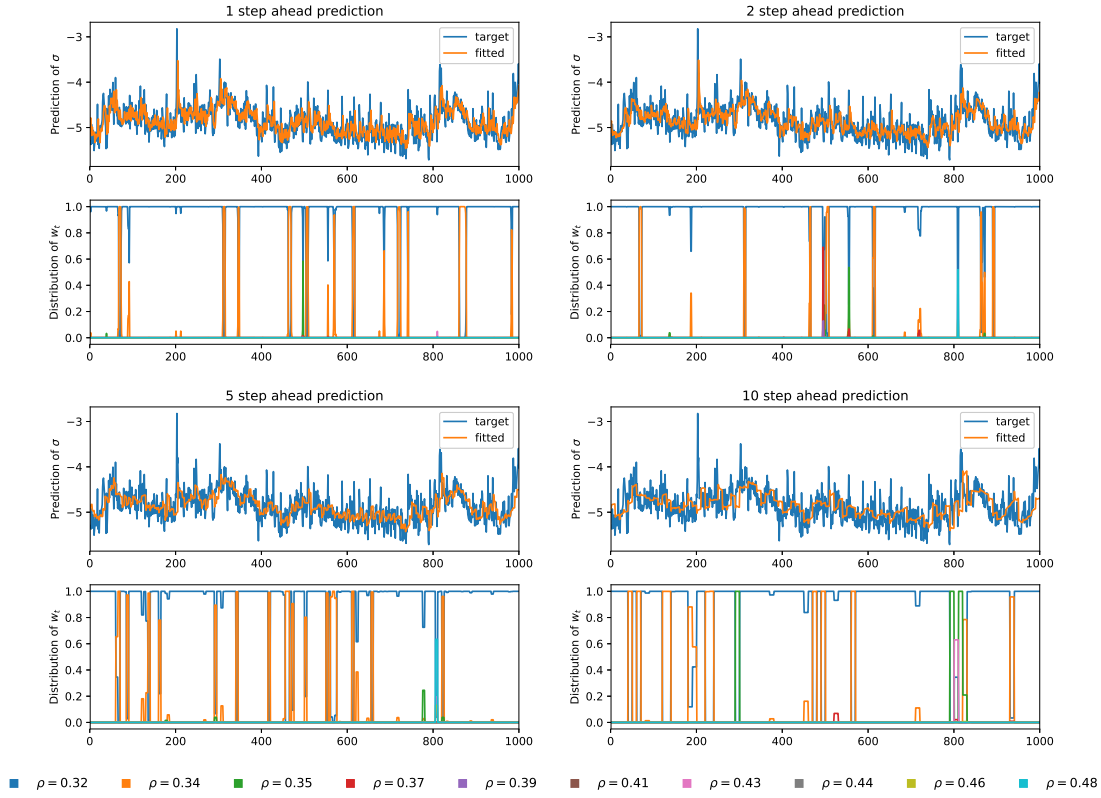

Figure 6.9: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using an equally spaced grid $\left[0.8\frac{1}{\sqrt{2\pi}}, 1.2\frac{1}{\sqrt{2\pi}}\right]$ around the constant value of $\sigma = \frac{1}{\sqrt{2\pi}}$. The networks have only been trained once on the training set as outlined in section 5.5.1.

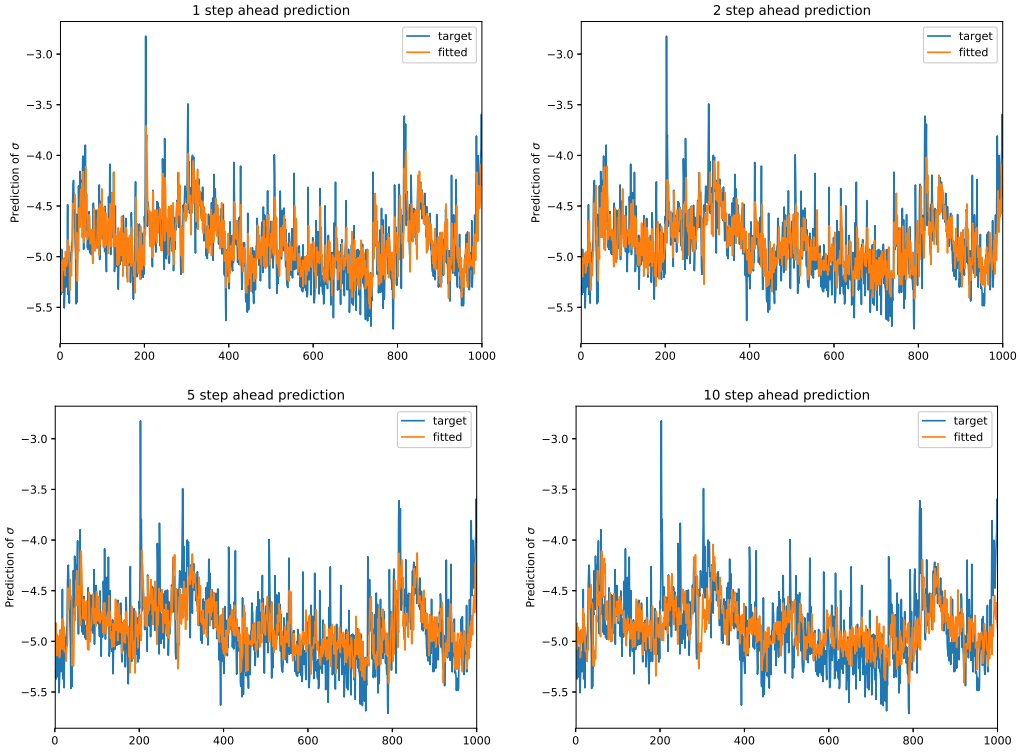

Figure 6.10: Single Echo State Network of equivalent size to the experts approach, namely $N = 1000$ internal neurons. The hyperparameters that have been tuned for the logMSE are the spectral radius and the bias in the interval $\rho \in (0, 3]$ and bias $b \in [-1, 1]$ and using a gradient boosted regression tree from the scikit-optimize package. See the appendix for details on this package.

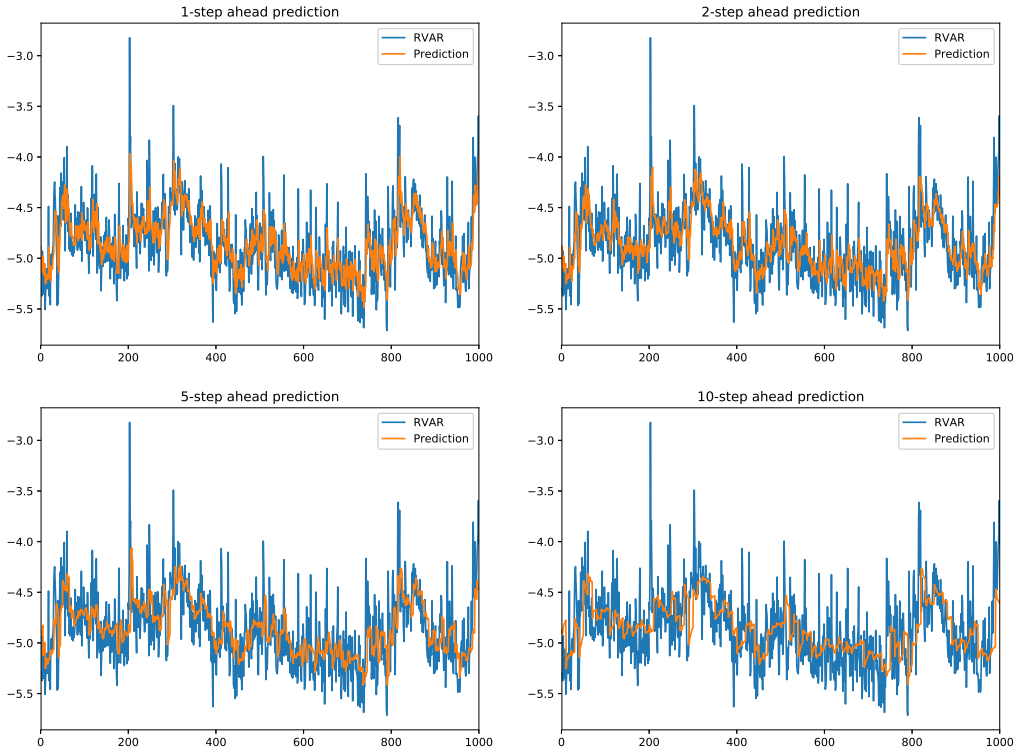

Figure 6.11: HAR model as proposed by Corsi (2009), e.g. with daily, weekly and monthly realized volatilities, that has been trained in an online fashion along the lines of recursive least squares (2.8)-(2.9).

networks are the same because we fixed the random number generator. Additionally, because the Mean Square Error approach to the weighting showed very little movement when only using equation (4.11), the errors for all models were rescaled to $[0, 1]$ before applying equation (4.11) to keep the comparison between the models. The 1-step ahead prediction is very volatile in terms of the weights which decreases with an increase in the prediction horizon. One can clearly see the dominance of the network with $\rho = 0.2$ in the period from step 500 to time step 800 where the series is trending down with little volatility. In the 2-step prediction, the network with the smallest spectral radius of 0.2 dominates towards the end of the out-of-sample period which is not expected given that the increase in the prediction step would be expected to favor networks with longer memory, so a higher spectral radius. The network with $\rho = 2.0$ is dominating in the mid to end section of the horizon after $\rho = 0.8$ and $\rho = 1.2$ have taken over after $\rho = 0.4$ started the out-of-sample prediction. The weights assigned to networks with smaller spectral radii could be explained by the fact, that the output connections W_{out} of the networks have not been retrained during the out-ouf-sample prediction and this may be indicative of a regime shift, where the long term dependencies in the series have changed and the prediction is predominantly based on the recent history of the series. On the contrary, there are the 5-step and 10-step ahead predictions. There the networks

with the largest spectral radius of $\rho = 2.0$ as gaining weight in the 5-step prediction and and is clearly dominating the 10-step prediction which means that the longer term dependencies are a dominant factor for the predictive ability. Looking at the *loss experts* in figure 6.2.2 the MSE in the original scale as loss function, the 1-step ahead prediction is still very volatile. The 2-step ahead prediction also shows a very similar picture except that the network with $\rho = 2.0$ is not gaining as much weight as in the log-scale MSE case. The 5-step and 10-step prediction differ significantly. For the original-scale MSE the networks with the lowest spectral radii of $\rho = 0.4$ and $\rho = 0.2$ receive the dominant weights in the beginning and end section of the prediction horizon, respectively. The QLIKE weighted experts in figure 6.2.2 show a combination of the previous two weighting approaches. Where, still, the 1-step ahead prediction shows its volatile and very similar picture, the 2-step ahead prediction resembles the log-scale MSE distribution of weights and the 5-step and 10-step prediction are more similar to the original-scale MSE weighted experts. The 10-step prediction is even more extrem in the sense, that the network with $\rho = 0.4$ receives a weight of 1 almost all of the time.

The *plasticity experts* with the same output distributions in figure 6.8 do present a very volatile and quickly adapting behaviour. The similarity between the four panels of figure 6.8 is expected given that after the prediction, the network evolution is corrected by the true series and this is done for all prediction steps. The only difference is the granularity of the weights update which can be seen very well comparing the 1-step and 10-step ahead prediction. The latter behaves like a step function. Additionally, comparing the weights to the experts approach, the distribution of component weights is much more extrem in the sense that the weights mostly alternate between 0 and 1 and in rare occasions have a true mixture of components. The similarity of all the panels leads to the conclusion that the empirical distribution and the resulting likelihood of a network's activation can change very quickly. The series and the distribution of the weights, unfortunately, don't seem to correspond in a certain way. The *plasticity experts* with a grid of output distributions in figure 6.2.2 on the other hand do present a very different distribution of output weights. Where with constant distributions, most of the networks were involved in the prediction at one point in time, the two experts with the lowest volatility distributions of $\sigma = 0.32$ and $\sigma = 0.34$ make up the resulting predictions. As with the constant plasticity experts the weights alternate between 0 and 1 and don't seem to coincide with the series in any remarkable pattern. In very few instances, the networks with $\sigma = 0.35$ and $\sigma = 0.37$ do receive a small weight which is not of long duration[27].

Plots for the predictive performance of the single ESN and the HAR model are provided in figures 6.2.2 and 6.11. The single ESN shows some very choppy behaviour which could indicate an overfitting during the training process. The tuned hyper-

[27]The values for σ don't seem equidistant but this is due to the rounding to two digits.

Fixed 1-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.08758738	1.05803697	0.24808956
Experts QLIKE	0.08760709	1.05505096	0.24823963
Experts logMSE	0.08758126	1.05689476	0.24847465
HAR	0.06847035	0.84973355	0.19371920
Plasticity Constant	0.08933186	1.04572311	0.25836014
Plasticity Grid	0.08640221	1.01276597	0.25078255
Single MSE	0.08334474	0.94009009	0.20983467
Single QLIKE	0.07926708	0.93331580	0.20988650
Single logMSE	0.08267177	0.95702170	0.22700093

Table 6.3: Errors for the prediction for the different models and the 1-step ahead predictions using a fixed training set as outlined in section 5.5.1.

Fixed 2-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.08986521	1.07277764	0.25348541
Experts QLIKE	0.09085416	1.09143206	0.25588969
Experts logMSE	0.09084808	1.08410217	0.25612836
HAR	0.07503286	0.91231669	0.21128034
Plasticity Constant	0.08938321	1.05612199	0.26294063
Plasticity Grid	0.08817040	1.03261120	0.25984234
Single MSE	0.08974965	1.00270113	0.22271637
Single QLIKE	0.08679926	0.99016069	0.23042241
Single logMSE	0.09181919	1.04655279	0.25124330

Table 6.4: Errors for the prediction for the different models and the 2-step ahead predictions using a fixed training set as outlined in section 5.5.1..

paremeters of the single ESN are $\rho = 1.2511$ and $b = 0.4406$ [28]. The HAR model is in line with expectations. The difference can also be attributed to the fact that the predictions are plotted in the log-scale of σ. The higher the prediction steps, the calmer the predictions become. Where in the panel of 1-step ahead predictions the forecasts do follow the true series quite strongly this effect decreases with the prediction step and the forecasts have a tendency towards the histrical mean of the series.

For the single ESN, only the prediction of the logMSE related measurements are presented because the chosen hyperparameters were almost the same as the other two error measurements. Out of the three, the logMSE was chosen as it is related to the original scale of the series, namely σ. The very volatile behaviour of the predictions leads to the concern of overfitting during the training of the network's output connections. This is supported by the fact that the regularization parameter of the ridge regression was chosen on the border of admissible values.

The tables 6.3 to 6.6 present the prediction error for the out-of-sample part of the series, this is the last 1000 observations, for the different prediction step sizes.

[28]The bias b is multiplied by W_{bias} to become $\mathbf{b} = b \cdot W_{bias}$.

Fixed 5-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.09987065	1.17159711	0.30540002
Experts QLIKE	0.09973242	1.16626035	0.30564940
Experts logMSE	0.09832657	1.14407191	0.30210400
HAR	0.08737418	1.03945024	0.26712508
Plasticity Constant	0.09897247	1.14413505	0.31834281
Plasticity Grid	0.09600595	1.11135717	0.31021388
Single MSE	0.10656475	1.19625577	0.29094670
Single QLIKE	0.09785688	1.11072290	0.28079199
Single logMSE	0.10401672	1.16531048	0.30932509

Table 6.5: Errors for the prediction for the different models and the 5-step ahead predictions using a fixed training set as outlined in section 5.5.1..

Fixed 10-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.10915740	1.25883661	0.33336069
Experts QLIKE	0.11004703	1.25491580	0.34062729
Experts logMSE	0.11021187	1.24497983	0.33841503
HAR	0.10138104	1.16342703	0.31118643
Plasticity Constant	0.11787165	1.31586371	0.36889776
Plasticity Grid	0.11188359	1.25658633	0.34886447
Single MSE	0.12216529	1.31975650	0.34693690
Single QLIKE	0.11271315	1.23937364	0.32947925
Single logMSE	0.11743093	1.31681358	0.35623439

Table 6.6: Errors for the prediction for the different models and the 10-step ahead predictions using a fixed training set as outlined in section 5.5.1..

The error measurements, namely logMSE, MSE and QLIKE are referring to the different scales of $\log(\sigma)$, σ and σ^2, respectively, which is in accordance with the hyperparameters' validation of the *loss experts* and the single ESN.
The HAR model by Corsi (2009) presents the best performance throughout all prediction steps for the logMSE. After the log transformation the volatility process becomes linear and it is to be exptected that the HAR model, which can be regarded as a restricted AR(22) model and therefore is linear, should outperform the other models in this matter. The better performance of the HAR model can also be seen in MSE and in the QLIKE error. This phenomenon can be explained by the fact that the HAR model is by its nature of an AR-process capturing changes in the regime of the time series where the reservoir computing models are only fed the time series itself and don't receive a preprocessed (in the sense of aggregation) of the signal. The only regime change that the HAR model is not able to adapt to is the intercept, but this is also the case for the other models and in comparison the HAR model beats all other models. This will also become clearer in the following section 6.2.3
Comparing the *plasticity experts* and the *loss experts* in the logMSE, the predictive performance of the former is better for the 1, 2 and 5-step ahead prediction and also for the 10 step prediction when using the grid of targeted output distributions. The 10-step ahead prediction horizon of the constant plasticities is worse compared to the experts of all loss functions. This tendency mostly continues for the MSE, where the *plasticity experts* outperform the *loss experts* up to the 5-step prediction but both *plasticity experts*, with constant output distributions and using a grid, are not able to beat the *loss experts* in the MSE.
A comparison between the two plasticity approaches is difficult as the the constant distribution seems to outperform in the logMSE and the MSE for the 5 and 10-step prediction, but the grid of distributions seems to be favourable in terms of logMSE and QLIKE for the 1 and 2-step prediction setting.
The *loss experts* consistently dominate in the QLIKE where, independent of the prediction step, they outperform all other model specifications.

6.2.3 Rolling Training Performance

This section presents the out-of-sample performance for the approach where the training set is shifted into the testing set. This procedure has been outlined in section 5.5.2.
Figures 6.2.3 to 6.14 present the performance of the performance of the *loss experts* using the rolling window training. The logMSE trained networks shows a very consistent picture. The weighting in the 1-step ahead predictions is very volatile where almost every network receives significant weight at some point over the testing horizon which leads to the interpretation that those predictions are an interplay of

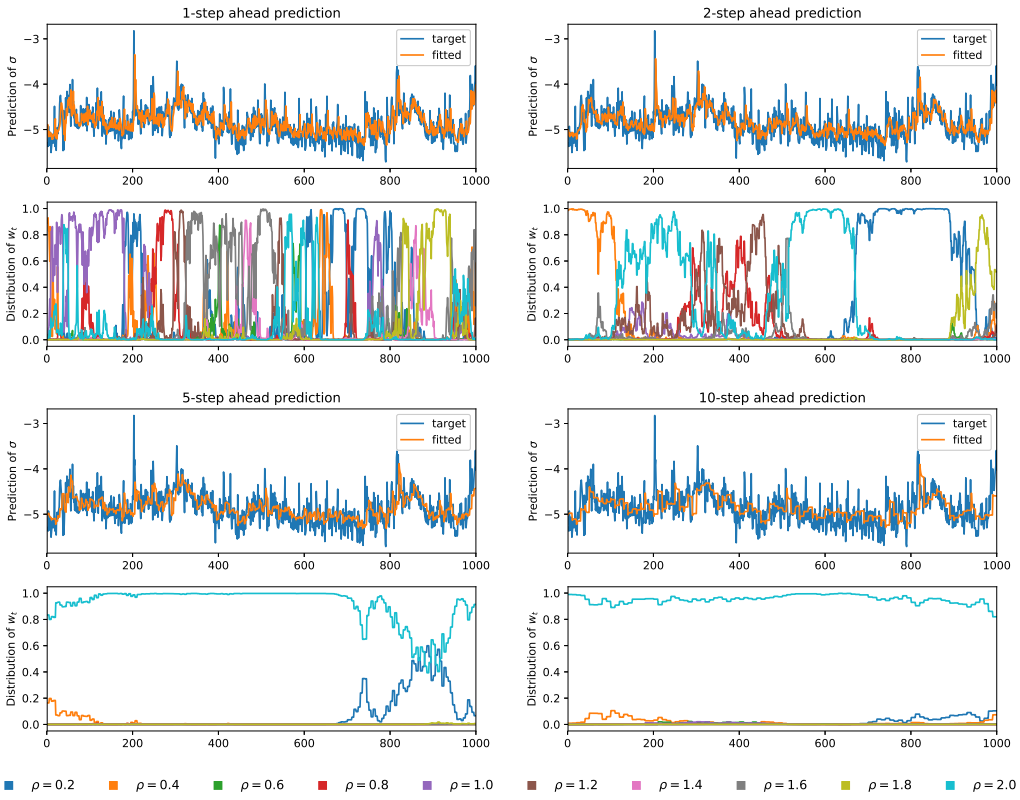

Figure 6.12: This is the *loss experts* approach with exponential weighting of the experts based on the mean squared error of the log transformation $\log(\sigma)$. The values of spectral radii are equally space in the interval $\rho \in [0.2, 2.0]$. The networks have been retrained in a rolling window fashion as outlined in section5.5.2.

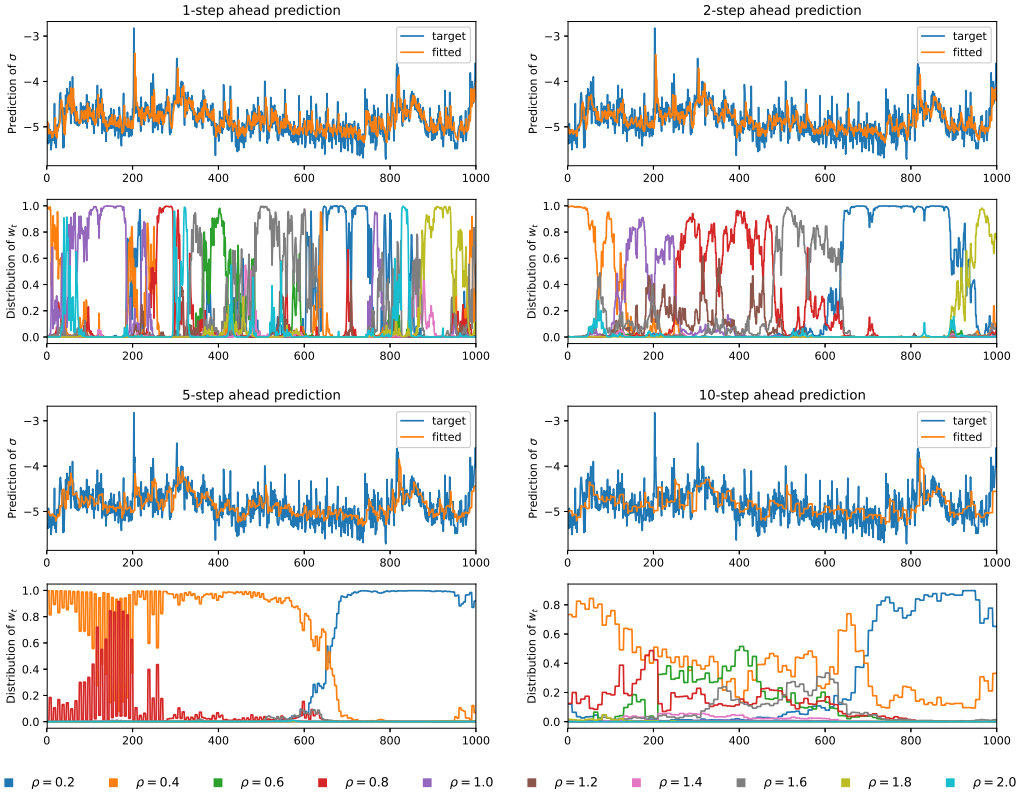

Figure 6.13: This is the *loss experts* approach with exponential weighting of the experts based on the mean squared error of the original time series, so σ. The values of spectral radii are equally space in the interval $\rho \in [0.2, 2.0]$. The networks have been retrained in a rolling window fashion as outlined in section5.5.2.

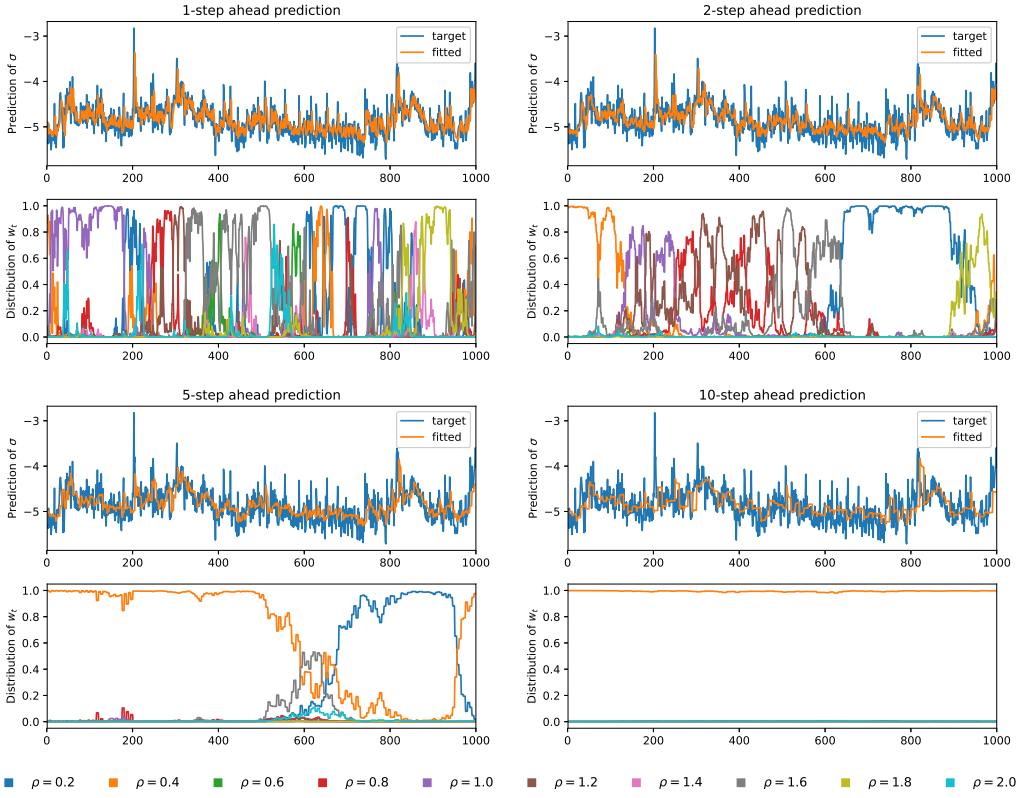

Figure 6.14: This is the *loss experts* approach with exponential weighting of the experts based on the QLIKE loss function, so in the scale of σ^2. The values of spectral radii are equally space in the interval $\rho \in [0.2, 2.0]$. The networks have been retrained in a rolling window fashion as outlined in section5.5.2.

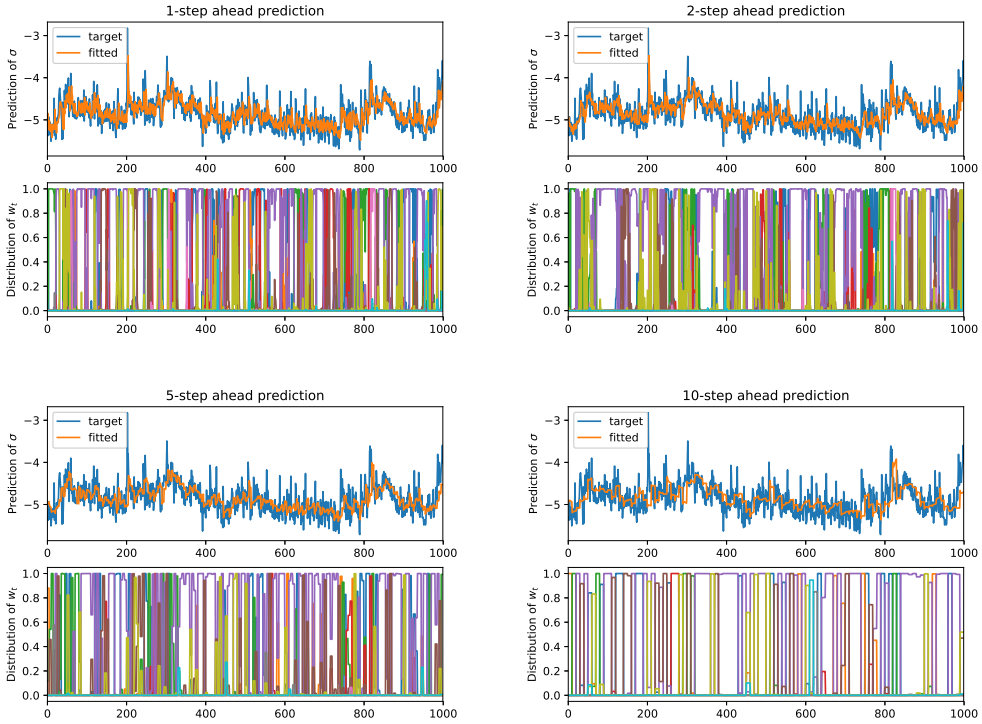

Figure 6.15: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using a constant $\sigma = \frac{1}{\sqrt{2\pi}}$ for all networks. The networks have been retrained in a rolling window fashion as outlined in section5.5.2.

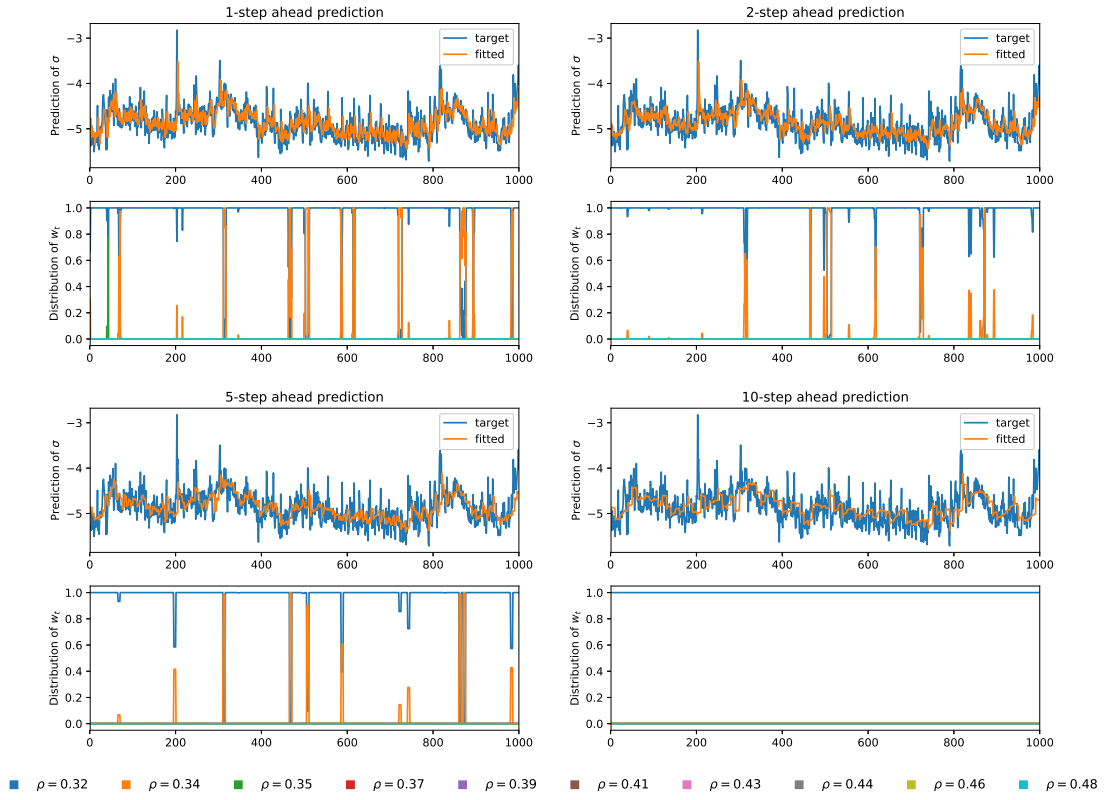

Figure 6.16: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using an equally spaced grid $\left[0.8\frac{1}{\sqrt{2\pi}}, 1.2\frac{1}{\sqrt{2\pi}}\right]$ around the constant value of $\sigma = \frac{1}{\sqrt{2\pi}}$. The networks have been retrained in a rolling window fashion as outlined in section5.5.2.

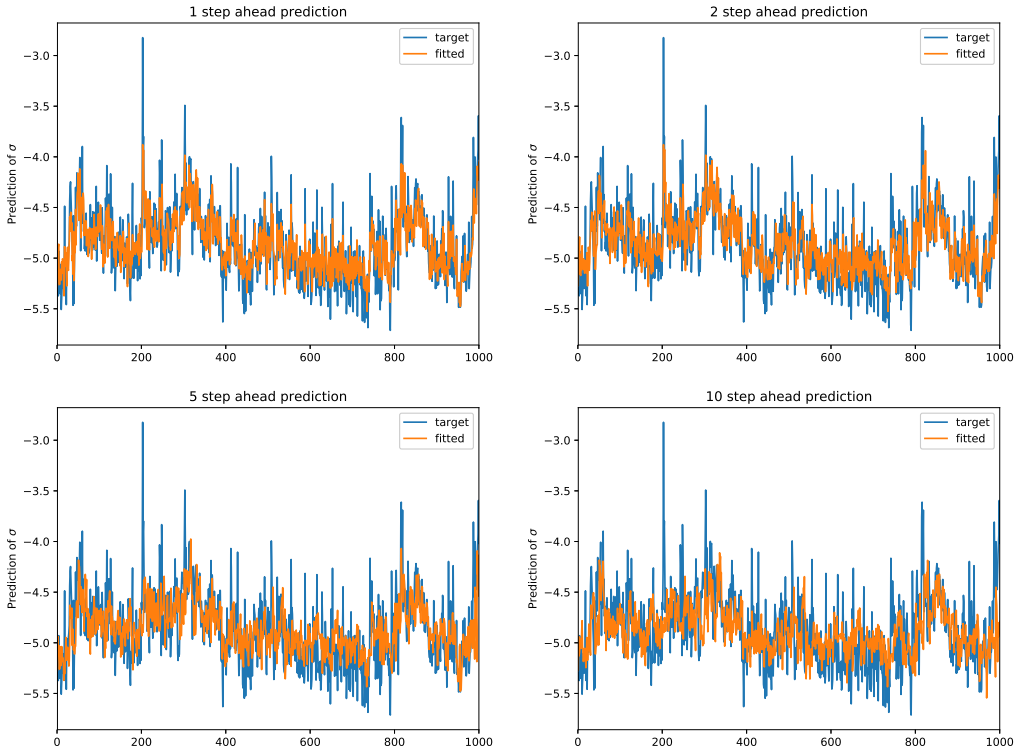

Figure 6.17: Single Echo State Network of equivalent size to the experts approach, namely $N = 1000$ internal neurons. The hyperparameters that have been tuned for the logMSE are the spectral radius and the bias in the interval $\rho \in (0, 3]$ and bias $b \in [-1, 1]$ and using a gradient boosted regression tree from the scikit-optimize package. See the appendix for details on this package. The networks have been retrained in a rolling window fashion as outlined in section5.5.2 but hyperparameters have only been optimized on the fixed training set.

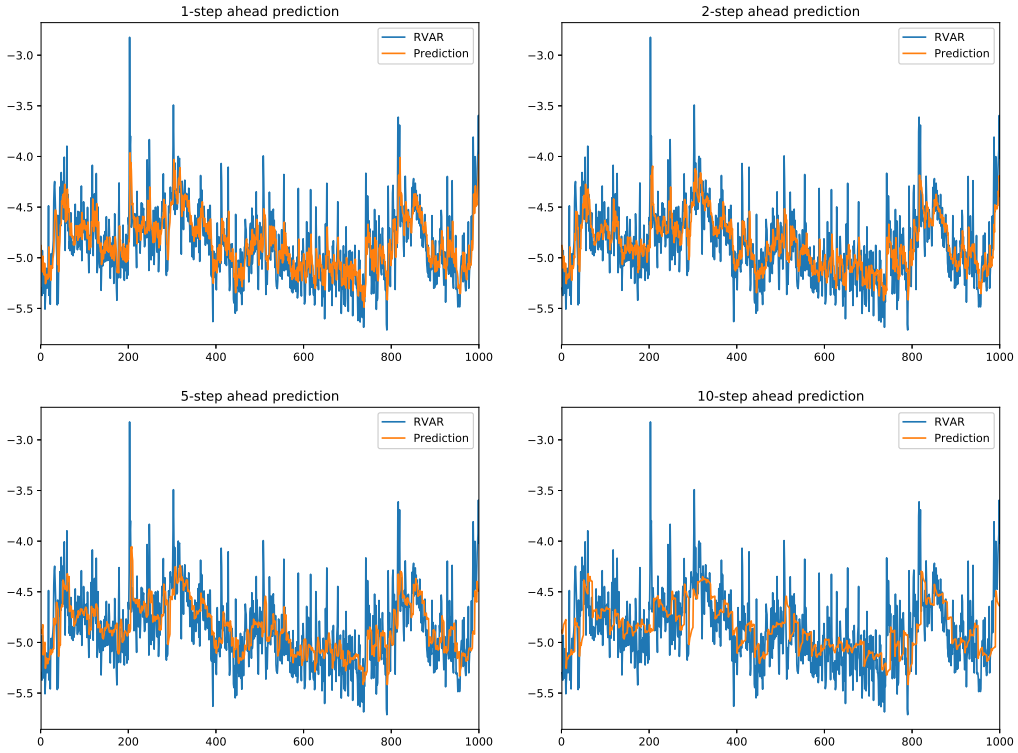

Figure 6.18: HAR model as proposed by Corsi (2009), e.g. with daily, weekly and monthly realized volatilities. In line with the Echo State Networks, the HAR model has been retrained in the same rolling window fashion as the reservoir commputing models. Given the size of the dataset, no hyperparameters (e.g. regularization) were needed.

short term moves (smaller ρ) and longer term trends (higher ρ). In the 2-step head prediction, the weights are not as widely distributed over the networks but mostly assigned to the networks with $\rho = 0.4$ in the very beginning, followed by $\rho = 2.0$ towards the middle and $\rho = 0.2$ dominating the end section of the horizon. $\rho = 1.8$ gains some weight in the very end. The 5-step and 10-step ahead predictions present a clear dominance of $\rho = 2.0$ over most of the horizon for the 5-step and over the whole horizon for the 10-step predictions. This is in line with expectations because a higher ρ signifies a longer memory of the network which is needed for longer out-of-sample predictions.

The broad picture of the weights' distribution is the same for the 1-step and 2-step ahead prediction in the *loss experts* with the MSE as error function. The 1-step ahead prediction shows a very similar picture, but slight differences are recognizable by just eyeballing. The 2-step ahead prediction as well shows a very similar picture, however, the weight of $\rho = 2.0$ that is present in the logMSE seems to have been replaced by the network with $\rho = 1.0$ that gains some weight in the beginning to mid section of the prediction.

Putting the *loss experts* with the QLIKE loss function in contrast with the other two, the results look more in line with the MSE than with the logMSE experts. This makes sense because the scales of σ and σ^2 are more alike compared to the $\log(\sigma)$. Surprisingly, the 1-step and 2-step ahead prediction, as with the MSE, still look similar in the sense, that we find a broad spectrum of networks receiving weight in the 1-step ahead prediction and for the 2-step ahead prediction the networks of $\rho = 0.4$, followed by a mixture of different networks, the network of $\rho = 0.2$ in the end section and $\rho = 1.8$ in the very end of the prediction horizon. For the 5-step ahead prediction, the network with $\rho = 0.4$ is dominating in the beginning and towards the end, the network with $\rho = 0.2$ is taking over. They again switch roles in the very and it ends with $\rho = 0.4$ as the dominant network. The 10-step ahead prediction is dominated by $\rho = 0.4$ throughout the whole prediction horizon.

The general picture that we have seen in the *loss experts* using MSE and QLIKE suggest that, if we are not in the log-scale, short term moves dominate the predictions of future volatility, which stands in contrast to the logMSE where the network with the highest value of ρ, namely the network with $\rho = 2.0$ tended to dominate the longer prediction horizons.

The *plasticity experts* with equal output distributions show a very similar weight distribution compared to the fixed training set. The distribution is still very volatile which makes a direct comparison difficult. For the networks with different output distributions, however, the distribution is even more concentrated towards the network with $\sigma = 0.32$. This is the case for all prediction steps. While the network with the second lowest $\sigma = 0.34$ is sporadically and for very short amounts of time gaining some weight in the 1-step ahead prediction, this happens fewer and fewer

Rolling 1-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.08802286	1.06410393	0.24888490
Experts QLIKE	0.08808350	1.06756442	0.24892281
Experts logMSE	0.08813920	1.06925160	0.24909907
HAR	0.06836762	0.84943039	0.19290737
Plasticity Constant	0.08742805	1.03377312	0.25916568
Plasticity Grid	0.08620444	1.02328620	0.25938944
Single MSE	0.04375821	0.52195032	0.09560064
Single QLIKE	0.03912077	0.50217962	0.08570485
Single logMSE	0.03968808	0.51286353	0.08754516

Table 6.7: Errors of the rolling window prediction for the different models and the 1-step ahead predictions.

Rolling 2-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.08923483	1.09138857	0.25424408
Experts QLIKE	0.08914354	1.08832016	0.25355556
Experts logMSE	0.08863778	1.07198041	0.25318178
HAR	0.07491863	0.91276585	0.21032252
Plasticity Constant	0.08822185	1.04517222	0.26429428
Plasticity Grid	0.08736104	1.04320714	0.26452248
Single MSE	0.06513711	0.70083327	0.15331451
Single QLIKE	0.05814992	0.65169373	0.13849141
Single logMSE	0.06232936	0.69091708	0.14581589

Table 6.8: Errors of the rolling window prediction for the different models and the 2-step ahead predictions.

as the prediction horizon increases. This doesn't seem to coincide with the series at all. Tests with a different grid of lower values was performed as well but didn't show significant improvements in performance or further insights into the resulting weighting. Analyses are provided in appendix A.3.

For the HAR model in figure 6.18 we can see a very similar behaviour compared to the fixed training set of figure 6.11.

In terms of prediction errors which are presented in the tables 6.7 to 6.10, the stellar performance of the HAR model compared to the reservoir computing models which we have seen in the previous section, decreases. For the rolling 1-step and 2-step ahead prediction, the single ESNs, independent of the associated error function, outperform the HAR model in all measurements. In the 5-step and 10-step ahead predictions, the comparison of the Single ESN and the HAR model depends on the error measurement as well as the hyperparameter tuning of the Single ESN.

Generally, the performance of the *loss experts*, the *plasticity experts* and the HAR model doesn't increase as significantly as for the single ESN. Additionally, their relative performance stays the same with the HAR model outperforming the expert approaches.

Rolling 5-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.09729214	1.14939778	0.30033757
Experts QLIKE	0.09713321	1.14502098	0.29860384
Experts logMSE	0.09732349	1.13903226	0.30362584
HAR	0.08724299	1.04101105	0.26582336
Plasticity Constant	0.09802419	1.14526361	0.31551162
Plasticity Grid	0.09590954	1.11738107	0.30815891
Single MSE	0.09332434	1.07093848	0.26010275
Single QLIKE	0.08905669	1.02041242	0.25948164
Single logMSE	0.09239568	1.04845941	0.27164816

Table 6.9: Errors of the rolling window prediction for the different models and the 5-step ahead predictions.

Rolling 10-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Experts MSE	0.11097113	1.27168492	0.34156865
Experts QLIKE	0.11154794	1.28125508	0.34201162
Experts logMSE	0.11092033	1.25180105	0.34742392
HAR	0.10088391	1.15992706	0.30959814
Plasticity Constant	0.11459149	1.30729493	0.36355034
Plasticity Grid	0.10688234	1.20814290	0.34068571
Single MSE	0.10229432	1.13029632	0.29080090
Single QLIKE	0.10515565	1.17085258	0.31481402
Single logMSE	0.11414491	1.26265779	0.35273585

Table 6.10: Errors of the rolling window prediction for the different models and the 10-step ahead predictions.

Comparing the *loss experts* and the *plasticity experts*, the grid of plasticity networks seems to outperform all *loss experts* in the logMSE and the MSE, but is worse in the QLIKE. The only exception of this pattern is the 10-step prediction, where the *plasticity experts* are slightly better even in the QLIKE. The constant plasticity networks is outperforming the *loss experts* in the logMSE and MSE for the 1-step and 2-step ahead prediction, for the 5-step ahead prediction it is worse than most *loss experts* in all error measurements with the execption of the MSE-tuned expert for with the MSE error measurement. In the 10-step ahead prediction the constant experts underperform compared to the *loss experts* in every aspect.
Comparing the fixed training and the rolling training approach, the errors of the *loss experts* increased for some cases and decreased for others. This leads to the conclusion that the expert methods were able to account for changes in the short and long-term dependencies in the series and re-estimating the output connections doesn't improve performance consistently. The *plasticity experts*, for the most cases, were able to improve the performance by some small margin. The HAR model was able to consistently improve for almost all combination of prediction step and error measurement. In some instances the original-scale MSE increased. The model that has by far benefitted the most from the rolling training was the single MSE that was able to improve its predictive performance significantly to now beat all other models.

7 Conclusion

In this thesis we introduced and investigated the novel approach of a plasticity based weighting of a set of experts which individually are Echo State Networks from the field of reservoir computing. This idea was build on different ideas like the Echo State Incremental Gaussian Mixture (Engel and Heinen, 2010; Heinen et al., 2011; Heinen, 2011; Pinto et al., 2011), the Gaussian Mixture Autoregressive Model (Kalliovirta et al., 2015) and the plasticity tuning of echo state network dynamics (Schrauwen et al., 2008). Especially the latter, where the authors tune the dynamics, namely the activation values of the reservoir, of an Echo State Network towards a normal distribution $\mathcal{N}(\mu, \sigma^2)$. Alltogether, theses individual ideas lead to the approach of weighting a set of ESN experts by the plasticity, this is the likelihood, of their network activation states which makes intuitive sense because it relates to information maximizatino of the network's activation values. This way enables an updating of the weights prior to the prediction step of the experts approach which is contrast to the more traditional weighting of a set of experts based on a loss function. For the weighting based on a loss function, the true values has to be observed and the weights can be adjusted *afterwards*. Under some assumptions that build on the successfull pre-tuning of network dynamics towards a targeted

normal distribution, we showed the close connection of the two approaches. More specifically, the plasticity weighting can be regarded as an update of the weights based on the expected loss of an expert.

In the application to the prediction of daily realized volatilities of the IBM stock price in the years between 2001 and 2018, we presented the predictive capabilities of this new weighting approach in comparison to the loss induced weighting of experts as well as a standard single Echo State Network of comparable size or the famous Heterogeneous Autoregressive Model (HAR) model. We examined two different prediction methodologies, namely a fixed training prediction and a rolling training prediction. Both, the loss induced experts and the plastiticy weighted experts were not able to outperform both of their comparators, where the single Echo State Network was able to outperform the HAR in the short term predictions. The fact that the weight distributions of the expert models from the two prediction methodologies didn't differ substantially but presented a very similar picture of weight distributions, led to the conclusion, that the expert methods were able to account for changes in the short and long-term dependencies in the series. Re-estimating the output connections even increased the prediction error in some cases. Focussing on the expert settings, the plasticity approach seemed to have slightly outperformed the loss induced weighting of experts. Given that this improvement was minor, further analysis, such as the model confidence set, could be conducted in this direction.

One direction of diving deeper into the plastiticy weighting of experts would be a more analysis based approach to choosing appropriate output distributions which may be task dependent or possible even would have general capabilities. Other directions may include the following: Firstly, a combination of the loss induced and the plasticity weighting of experts would be interesting way of extending the plasticity approach and to see whether the pre-prediction update of weighting coincides with the post-prediction update of weights based on a loss function. Secondly, plasticity weighted networks may be further enhanced by using a weighting scheme in the estimation of output weights that is based on the networks likelihood. This may account for heteroscedasticity in the network states and could potentially improve the estimation of output weights in Echo State Networks.

A Appendix

A.1 Mathematical Tools and Defintions

A.1.1 Sherman-Morrison Formula

The Sherman-Morrison formula is based on the Woodbury Formula:

Lemma A.1 (Woodbury Formula) *Let E and G be square invertible matrices of dimension $n_E \times n_E$ and $n_G \times n_G$ respectively. Let F and H be matrices of size $n_E \times n_G$ and $n_G \times n_E$ respectively, then the following identity holds:*

$$(E + FGH)^{-1} = E^{-1} - E^{-1}F(G^{-1} + HE^{-1}F)^{-1}HE^{-1} \tag{A.1}$$

The Sherman-Morrison formula is a special case of (A.1) where $G = 1$ and $F = u \in \mathbb{R}^{n_E}$ and $H = v \in \mathbb{R}^{n_E}$ which results in the equality

$$(E + uv')^{-1} = E^{-1} - \frac{E^{-1}uv'E^{-1}}{1 + v'E^{-1}u} \tag{A.2}$$

For the case of the recursive least squares this translates equation (2.6) into (2.8).

A.1.2 Standard Brownian Motion

A stochastic process $(B_t)_{t\geq 0}$ that satisfies

1. $B_0 = 0$ $\mathbb{P}$-*almost surely, e.g.* $\mathbb{P}(\omega \in \Omega : B(0, \omega) \neq 0) = 0$.
2. *The increments* $B_t - B_s$ *for* $t > s$ *follow a normal distribution* $\mathcal{N}(0, t - s)$.
3. *For all* $0 \leq t_1 < t_2 < ... < t_n$ *it holds that* $B_{t_2} - B_{t_1}, ..., B_{t_n} - B_{t_{n-1}}$ *are independent.*

is called Brownian Motion or Wiener Process.

A.2 Single ESN predictive performance

These are the predictive performances of a single ESN of equivalent size to the experts approach, namely $N = 1000$ internal neurons. The hyperparameters that have been tuned are the spectral radius and the bias in the interval $\rho \in (0, 3]$ and bias $b \in [-1, 1]$ using a gradient boosted regression tree from the scikit-optimize package. See the appendix for details.

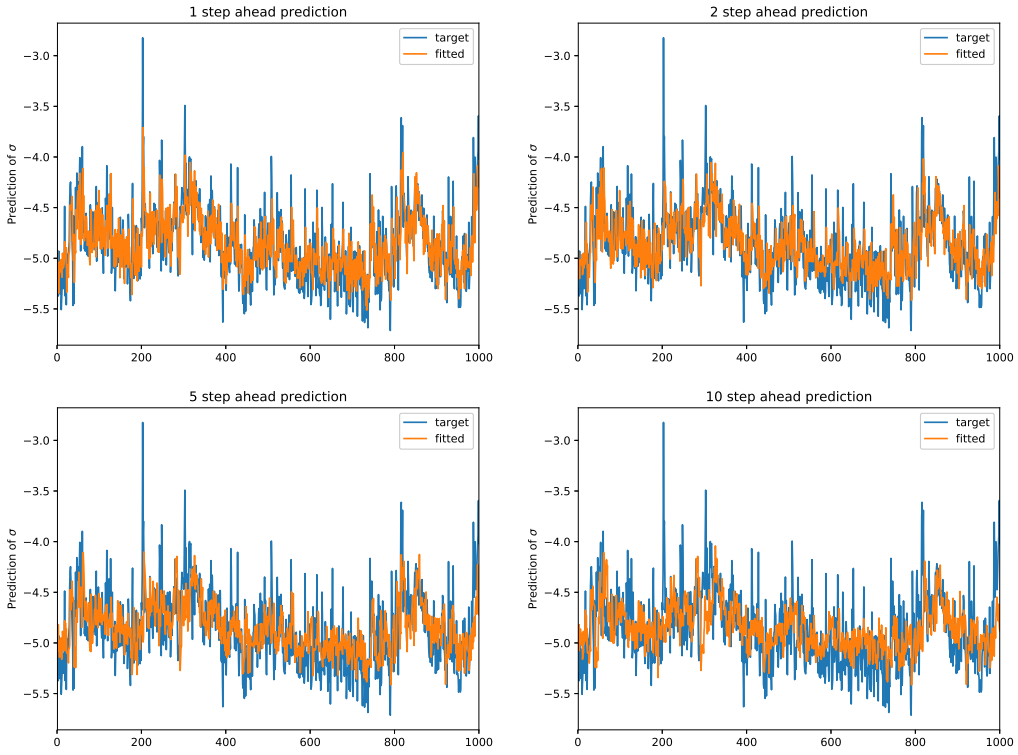

Figure A.1: Single Echo State Network using the logMSE as error function in the cross-validation.

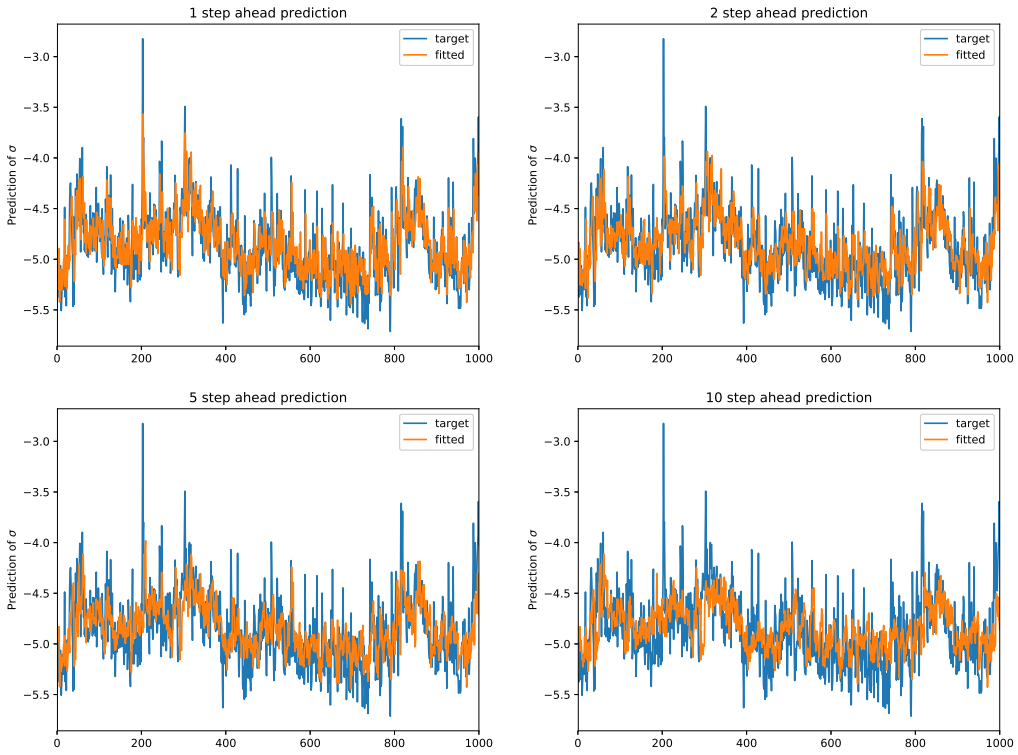

Figure A.2: Single Echo State Network using the QLIKE as error function in the cross-validation.

A.3 Grid of Smaller Sigma Values

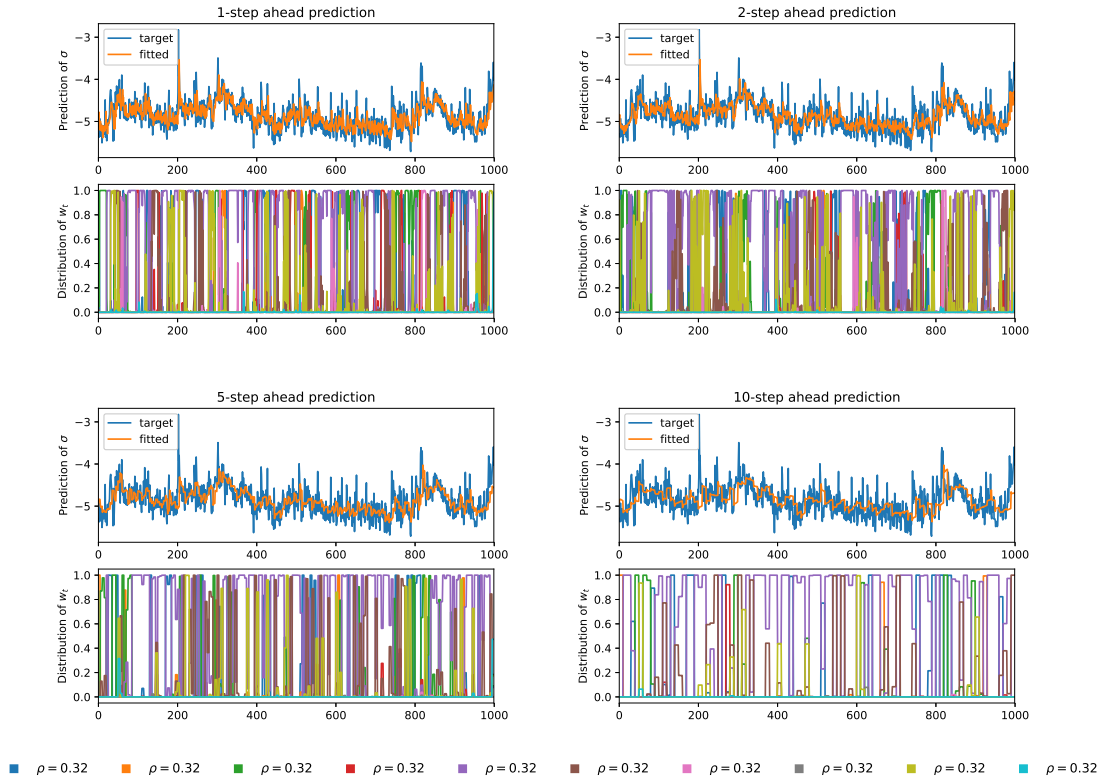

Figure A.3: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using a constant $\sigma = 0.32$ for all networks based on the dominate weight of this network in the grid approach of section 6.2.3.

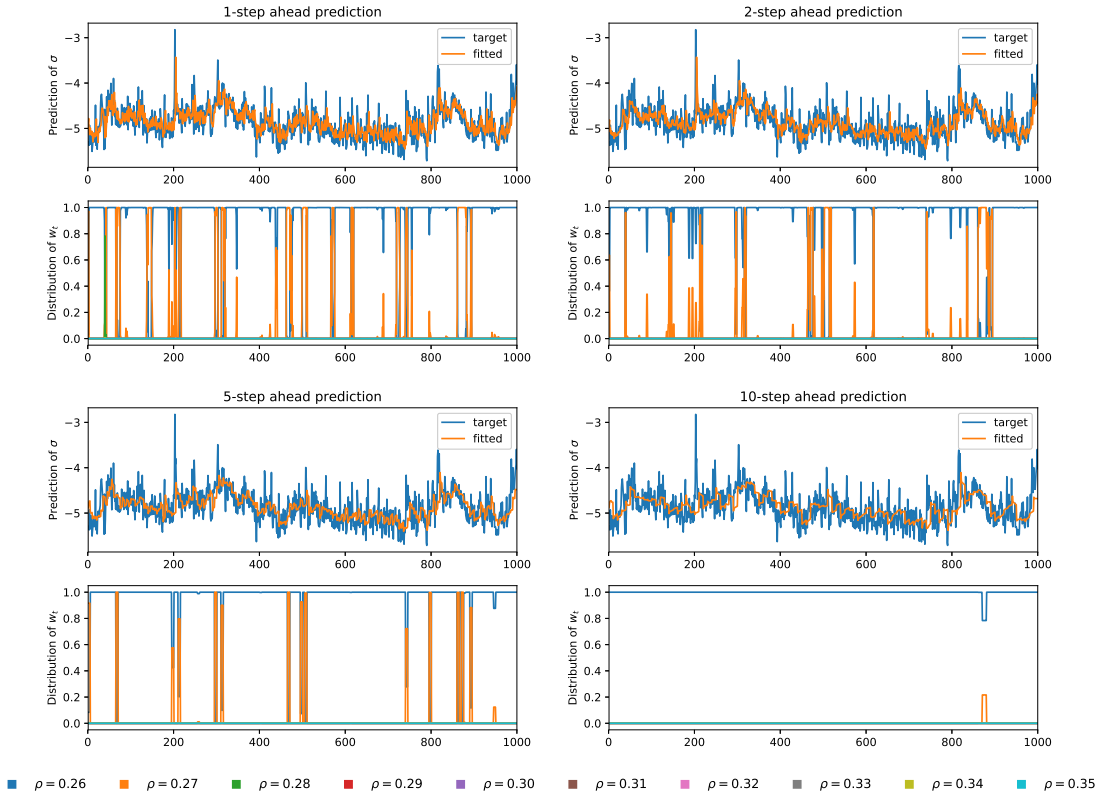

Figure A.4: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using an equally spaced grid $[0.26, 0.35]$ around the constant $\sigma = 0.32$ based on the dominate weight of this network in the grid approach of section 6.2.3.

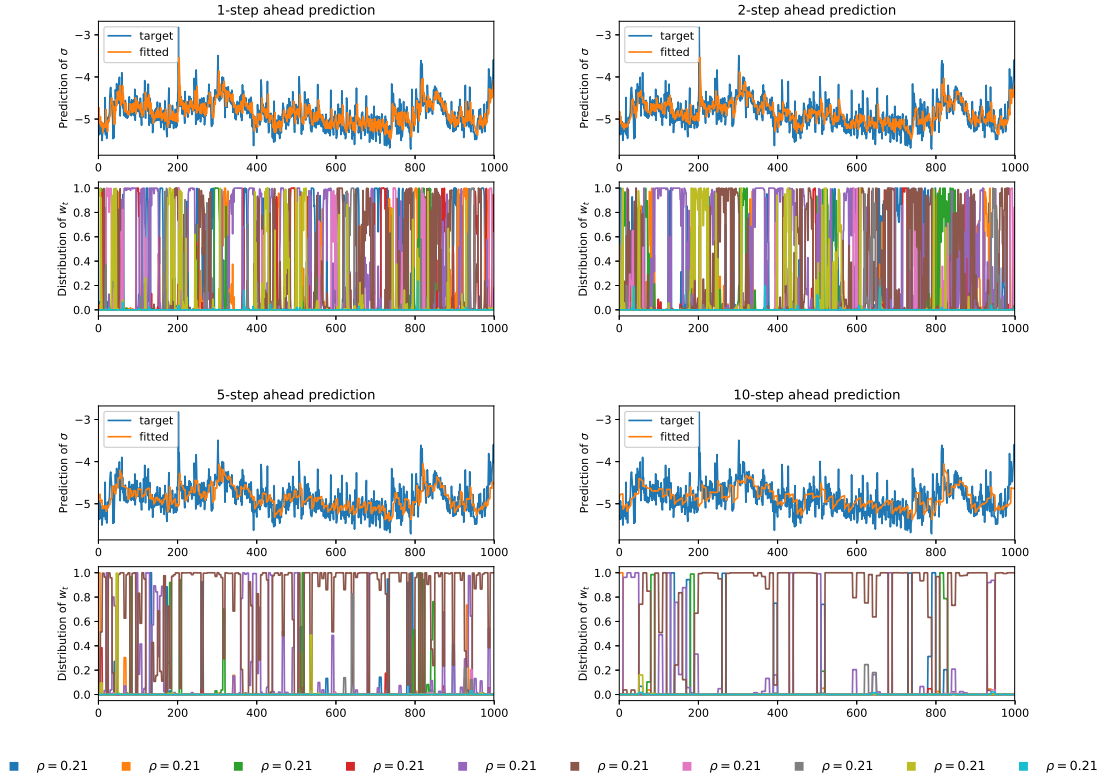

Figure A.5: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using a constant $\sigma = 0.21$ for all networks based variance of the underlying series of realized volatility.

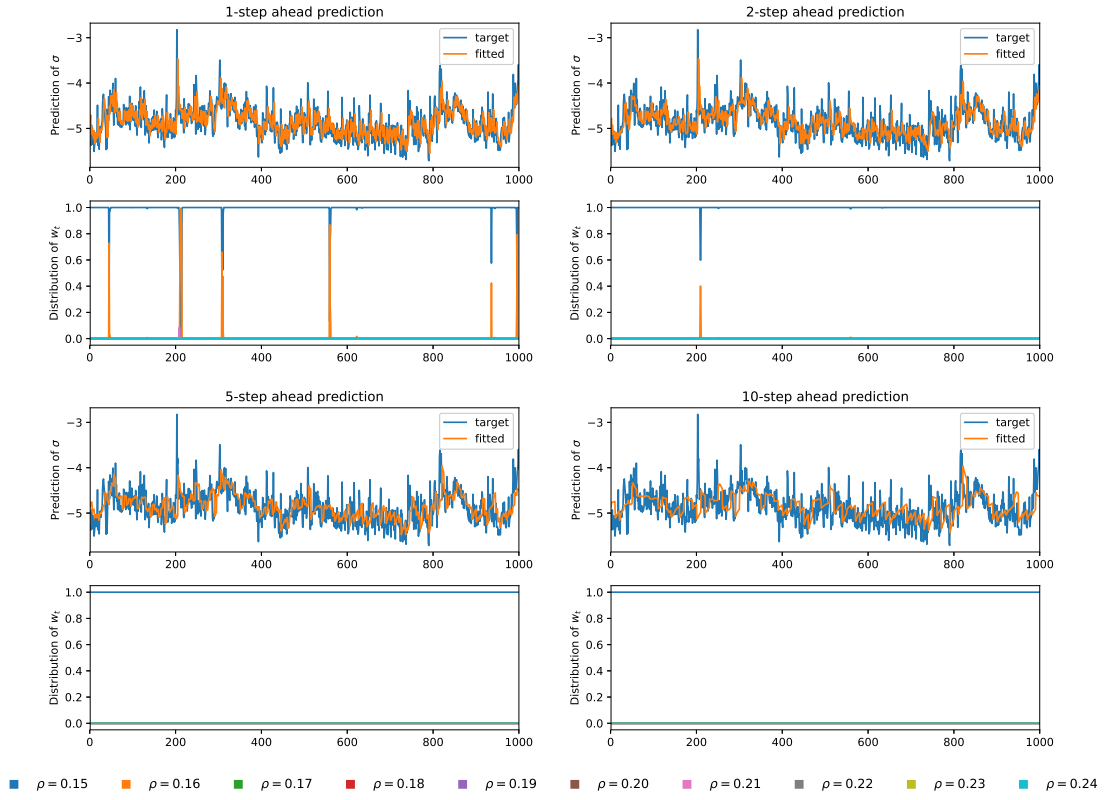

Figure A.6: This presents the predictive performance of the *plasticity experts* for 1, 2, 5 and 10 step ahead predictions. The targeted network activations are using an equally spaced grid $[0.15, 0.24]$ around the constant $\sigma = 0.21$ based variance of the underlying series of realized volatility.

Choosing the new alternative constant value of σ or the shifted grid, doesn't change the weight distribution of and their behaviour. For the constant value of σ the distributions are still very volatile. For the shifted grid of values for σ it turns out, that still the weights are distributed between the lowest σ network and the second lowest σ. This leads to the conclusion that likelihood of the network state is in fact reliant on the random weights of the internal connections and the plasticity tuning of the networks connections are not

Choosing the new constant value of $\sigma = 0.32$ doesn't improve the network performance significantly. Shifting the grid doesn't change much in the weight distribution. In the grid approach with a lower grid of values, still the network with the lowest σ receives the largest weight.

	Original	Middle	Low
Constant	$\frac{1}{\sqrt{2\pi}} \approx 0.40$	0.32	0.21
Grid	$\left[0.8\frac{1}{\sqrt{2\pi}}, 1.2\frac{1}{\sqrt{2\pi}}\right]$	$[0.26, 0.35]$	$[0.15, 0.24]$

Table A.1: Reference for the following tables

Rolling 1-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Plasticity Constant Middle	0.08726152	1.02852912	0.25950733
Plasticity Constant Low	0.08637962	1.01789215	0.25328345
Plasticity Constant	0.08742805	1.03377312	0.25916568
Plasticity Grid Low	0.08942995	1.03451887	0.25295346
Plasticity Grid Middle	0.08599342	1.02120263	0.25862333
Plasticity Grid	0.08620444	1.02328620	0.25938944

Table A.2: Errors for the rolling prediction for the different models and the 1-step ahead predictions.

Rolling 2-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Plasticity Constant Low	0.08736678	1.03018180	0.25832888
Plasticity Constant Middle	0.08816717	1.04446062	0.26580909
Plasticity Constant	0.08822185	1.04517222	0.26429428
Plasticity Grid Low	0.09136562	1.06222826	0.26008277
Plasticity Grid Middle	0.08762366	1.04400969	0.26553307
Plasticity Grid	0.08736104	1.04320714	0.26452248

Table A.3: Errors for the rolling prediction for the different models and the 2-step ahead predictions.

Rolling 5-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Plasticity Constant Low	0.09721939	1.13024216	0.31649472
Plasticity Constant Middle	0.09780792	1.13913434	0.31888618
Plasticity Constant	0.09802419	1.14526361	0.31551162
Plasticity Grid Low	0.10543143	1.18778672	0.34084742
Plasticity Grid Middle	0.09543957	1.11023250	0.30987966
Plasticity Grid	0.09590954	1.11738107	0.30815891

Table A.4: Errors for the rolling prediction for the different models and the 5-step ahead predictions.

Rolling 10-step ahead	logMSE	MSE ($\times 10^{-5}$)	QLIKE
Plasticity Constant Low	0.10915696	1.22575270	0.35292040
Plasticity Constant Middle	0.11126106	1.25985877	0.36134648
Plasticity Constant	0.11459149	1.30729493	0.36355034
Plasticity Grid Low	0.12052321	1.30609972	0.37401721
Plasticity Grid Middle	0.10630105	1.19898912	0.33948644
Plasticity Grid	0.10688234	1.20814290	0.34068571

Table A.5: Errors for the rolling prediction for the different models and the 10-step ahead predictions.

Changing the exact distribution of the targeted output distributions doesn't affect the performance very much. The best improvement could be achieved in teh 'Plas-

ticity Constant Low' for all different prediction steps when comparing to the original 'Plasticity Constant'. In terms of the grids of output distributions, the 'Plasticity Grid Middle' seems to improve the performance slightly. Alltogether there is a tendency that lower variance for the targeted output distributions may be preferrable to the one chosen in the empirical appplication. One possible explanation may be based on the variance which is around 0.21 for the whole dataset (training and testing). Another explanation may be, that the network is generally more capable of achieving output distributions with lower targeted variances. After all, given that the improvements are minor, a more rigurous analysis has to be conducted to support any specific choices of the output distributions.

A.4 Implementation

All implementations have been self written in Python where the main engine behind the code is *NumPy* (`https://numpy.org`) and no other high level libraries like *Tensorflow/Keras* (`https://tensorflow.org`) or *pyTorch* (`https://pytorch.org`) have been used. Link to the GitHub repository:

lucasburger.github.io/pyRC

A.4.1 Python's Scikit-Optimize

This section of the appendix is for some more light and details on how the python package was employed for the optimization of hyperparameters. In addition, we refer to the online presentation of the package under `https://scikit-optimize.github.io/stable/`. The following is mainly stated from this website: The package provides different optimizers which include the following models to approximate the expensive to evaluate function:

- **dummy_minimize**:
  This is basic grid search
- **gbrt_minimize**:
  Gradient Boosted Regression Trees
- **gp_minimize**:
  Gaussian Process
- **forest_minimize**:
  Tree Based regression

One can set multiple parameters in order to run the optimization. Required arguments are the function that one want to minimize and the dimensions and ranges to use. There are many other parameters to direct the optimization onto different paths, but all of them do have default values and don't necessarily need any attention. Visualization of the results is very easy as well and includes, among others, the diagnostics presented in figure A.7. For this example, the spectral radius and the leak of an ESN applied on the Mackey-Glass time series have been optimized. Convergence has been pretty fast, because it only took aroung 40 function evalutations.

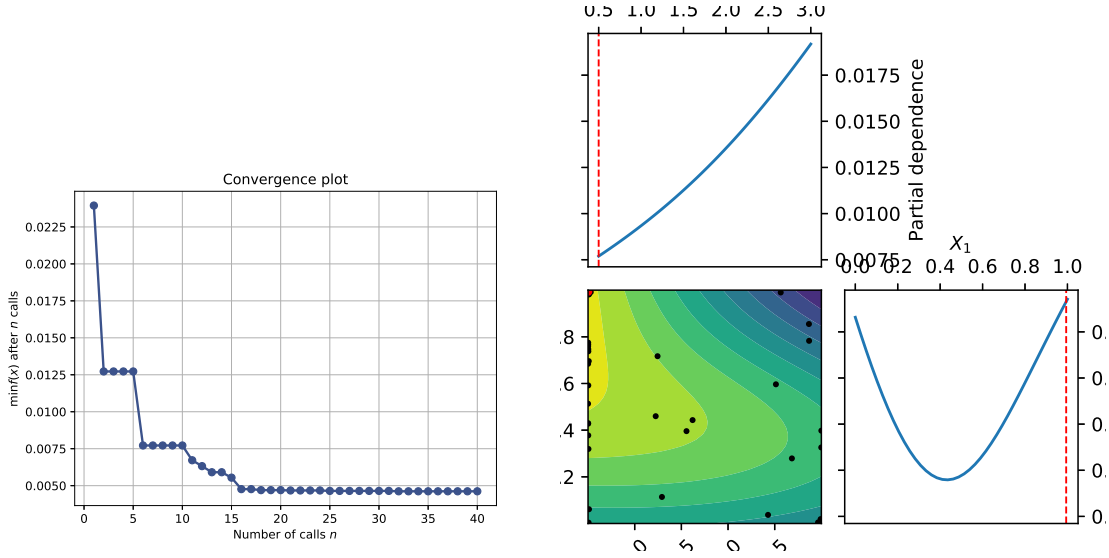

Figure A.7: On the left, we have the convergence plot of the function evaluations. On the right, we can see the partial dependence of two variables in rlation to each other. Blue dots represent functionevaluations and the red dot (barely recognizable in the top left corner) is the final value pair.

A.5 Mackey Glass Time Series and Network Dynamics

The Mackey-Glass differential equation (Glass and Mackey, 2010)

$$x'(t) = b\frac{x(t-\tau)}{1+x(t-\tau)^n} - cx(t) \quad , \quad t \in \mathbb{R} \tag{A.3}$$

for $b, c, \tau, n \in \mathbb{R}$ can be adjusted to generate complex dynamics.

Figure A.8 presents the Mackey-Glass time series for different values of the time-delay value τ.

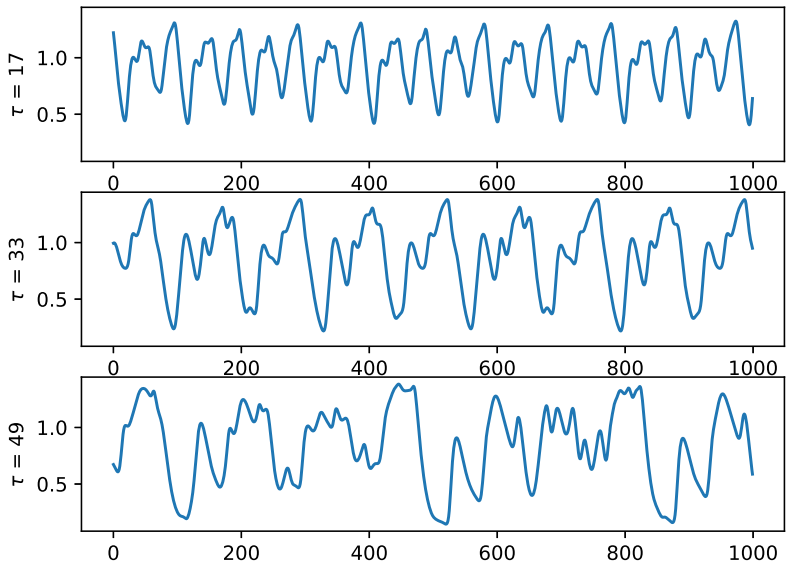

Figure A.8: Examples of the Mackey-Glass time series. Depending on the time-delay value of τ. Examples have been initialized with random values drawn from a uniform $\mathcal{U}(0.5, 1)$ that have been washed out over the simulation of 12000 points. Plots show the last 1000 of the simulations. Parameters are $b = 0.2$, $c = 0.1$, $n = 10$ and $\tau \in \{17, 33, 49\}$ as indicated in the plot. One can clearly see the effect on irregularity when increasing τ.

The neuron activations in figure A.9 indeed show certain similarities to the original time series which underlines the fact that the reservoir is able to reproduce different frequencies through its random, but well tuned weights. Depending on the input signal, the maginitude of neuron activations would be larger as well (there were also neurons with more variablity but they have been excluded from the plot).

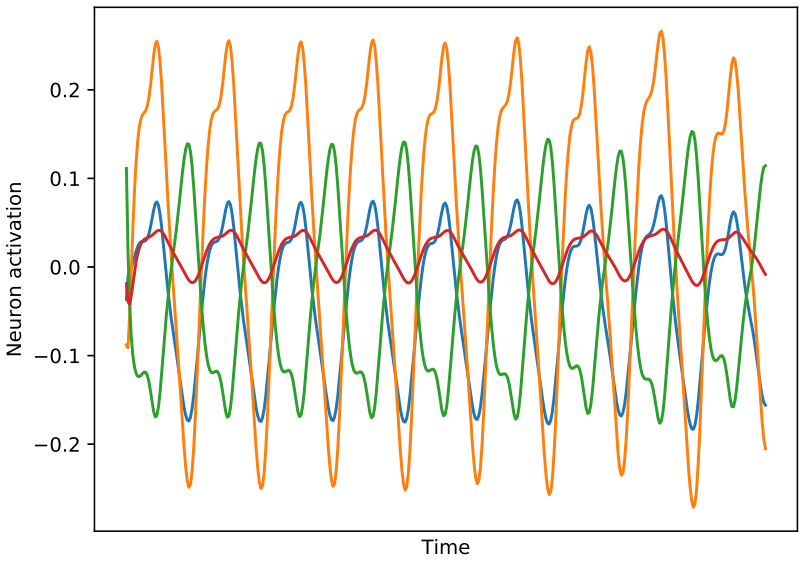

Figure A.9: Exemplary neuron activation when feeding the Mackey-Glass time series with $\tau = 17$. This shows that the neurons are able to present different representations of the input signal. Neurons have been selected based on their range to be able to nicely present them. Some neurons with much larger variations did exist as well.

References

Andersen, T., T. Bollerslev, and N. Meddahi (2011): "Realized volatility forecasting and market microstructure noise," *Journal of Econometrics*, 160, 220–234.

Andersen, T. G., T. Bollerslev, F. X. Diebold, and P. Labys (2001): "The Distribution of Realized Exchange Rate Volatility," *Journal of the American Statistical Association*, 96, 42–55.

——— (2003): "Modeling and Forecasting Realized Volatility," *Econometrica*, 71, 579–625.

Barndorff-Nielsen, O. E. and N. Shephard (2004): "Econometric Analysis of Realized Covariation: High Frequency Based Covariance, Regression, and Correlation in Financial Economics," *Econometrica*, 72, 885–925.

Basterrech, S., E. Alba, and V. Snášel (2015): "An Experimental Analysis of the Echo State Network Initialization Using the Particle Swarm Optimization," .

Bengio, Y., P. Simard, and P. Frasconi (1994): "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, 5, 157–166.

Billingsley, P. (2008): *Probabilty and measure*, USA: John Wiley Sons.

Boedecker, J., O. Obst, N. M. Mayer, and M. Asada (2009): "Initialization and selforganized optimization of recurrent neural network connectivity," *HFSP Journal*, 3, 340–349, pMID: 20357891.

Buonomano, D. V. and M. Merzenich (1995): "Temporal information transformed into a spatial code by a neural network with realistic properties," *Science*, 267 5200, 1028–1030.

Cesa-Bianchi, N. and G. Lugosi (2006): *Prediction, Learning, and Games*, USA: Cambridge University Press.

Corsi, F. (2009): "A Simple Approximate Long-Memory Model of Realized Volatility," *Journal of Financial Econometrics*, 7, 174–196.

Crook, N. (2007): "Nonlinear transient computation," *Neurocomputing*, 70, 1167 – 1176, advances in Computational Intelligence and Learning.

Engel, P. M. and M. R. Heinen (2010): "Incremental Learning of Multivariate Gaussian Mixture Models," in *Advances in Artificial Intelligence – SBIA 2010*,

ed. by A. C. da Rocha Costa, R. M. Vicari, and F. Tonidandel, Berlin, Heidelberg: Springer Berlin Heidelberg, 82–91.

Fama, E. F. (1970): "Efficient Capital Markets: A Review of Theory and Empirical Work," *The Journal of Finance*, 25, 383–417.

Gallicchio, C. and A. Micheli (2017): "Deep Echo State Network (DeepESN): A Brief Survey," .

Gallicchio, C., A. Micheli, and L. Pedrelli (2017): "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, 268, 87 – 99, advances in artificial neural networks, machine learning and computational intelligence.

Glass, L. and M. Mackey (2010): "Mackey-Glass equation," *Scholarpedia*, 5, 6908, revision #186443.

Gonon, L. and J.-P. Ortega (2018): "Reservoir Computing Universality With Stochastic Inputs," Preprint.

Gormley, I. C. and S. Frühwirth-Schnatter (2018): "Mixtures of Experts Models," .

Grigoryeva, L., J. Henriques, L. Larger, and J.-P. Ortega (2016): "Nonlinear Memory Capacity of Parallel Time-Delay Reservoir Computers in the Processing of Multidimensional Signals," *Neural Computation*, 28, 1411–1451, pMID: 27172266.

Grigoryeva, L. and J.-P. Ortega (2018a): "Echo state networks are universal," *Neural Networks*, 108, 495 – 508.

——— (2018b): "Universal discrete-time reservoir computers with stochastic inputs and linear readouts using non-homogeneous state-affine systems," *Journal of Machine Learning Research*, 19, 1–40.

Hastie, T., R. Tibshirani, and J. Friedman (2001): *The Elements of Statistical Learning*, Springer Series in Statistics, New York, NY, USA: Springer New York Inc.

Heinen, M., P. Engel, and R. Pinto (2011): "IGMN: An Incremental Gaussian Mixture Network that Learns Instantaneously from Data Flows," .

Heinen, M. R. (2011): "A connectionist approach for incremental function approximation and on-line tasks," .

Huang, G.-B., Q.-Y. Zhu, and C.-K. Siew (2006): "Extreme learning machine: Theory and applications," *Neurocomputing*, 70, 489 – 501, neural Networks.

Jaeger, H. (2001): "The echo state approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148.

——— (2002): "Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach," *GMD-Forschungszentrum Informationstechnik, 2002.*, 5.

——— (2003): "Adaptive Nonlinear System Identification with Echo State Networks," *NIPS*.

Jaeger, H. and H. Haas (2004): "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, 304, 78–80.

Jaeger, H., M. Lukoševičius, D. Popovici, and U. Siewert (2007): "Optimization and applications of echo state networks with leaky- integrator neurons," *Neural Networks*, 20, 335 – 352, echo State Networks and Liquid State Machines.

Jarvis, S., S. Rotter, and U. Egert (2010): "Extending Stability Through Hierarchical Clusters in Echo State Networks," *Frontiers in Neuroinformatics*, 4, 11.

Kalliovirta, L., M. Meitz, and P. Saikkonen (2015): "A Gaussian Mixture Autoregressive Model for Univariate Time Series," *Journal of Time Series Analysis*, 36, 247–266.

Kerridge, D. (1967): "Errors of Prediction in Multiple Regression with Stochastic Regressor Variables," *Technometrics*, 9, 309–311.

Kim, T. and T. Adali (2001): "Complex backpropagation neural network using elementary transcendental activation functions," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 2, 1281–1284 vol.2.

Koprinkova-Hristova, P. and G. Palm (2011): "ESN Intrinsic Plasticity versus Reservoir Stability," in *Artificial Neural Networks and Machine Learning – ICANN 2011*, ed. by T. Honkela, W. Duch, M. Girolami, and S. Kaski, Berlin, Heidelberg: Springer Berlin Heidelberg, 69–76.

Kushner, H. and G. Yin (2003): *Stochastic Approximation and Recursive Algorithms and Applications*, vol. 35, USA: Springer-Verlag New York.

L. Büsing, B. Schrauwen, R. L. (2010): "Connectivity, dynamics and memory in reservoir computing with Binary and Analog Neurons," *Neural Computation*, 5.

Lin, X., Z. Yang, and Y. Song (2011): "Intelligent stock trading system based on improved technical analysis and Echo State Network," *Expert Systems with Applications*, 38, 11347 – 11354.

Liu, Y., J. Zhang, C. Gao, J. Qu, and L. Ji (2019): "Natural-Logarithm-Rectified Activation Function in Convolutional Neural Networks," *ArXiv*, abs/1908.03682.

Lukoševičius, M. (2012): *A Practical Guide to Applying Echo State Networks*, Berlin, Heidelberg: Springer Berlin Heidelberg, 659–686.

Ma, Q., E. Chen, Z. Lin, J. Yan, Z. Yu, and W. W. Y. Ng (2019): "Convolutional Multitimescale Echo State Network," *IEEE Transactions on Cybernetics*, 1–13.

Maass, W., T. Natschläger, and H. Markram (2002): "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural computation*, 14, 2531–60.

Markowitz, H. (1952): "Portfolio Selection*," *The Journal of Finance*, 7, 77–91.

Ozturk, M. C., D. Xu, and J. C. Príncipe (2007): "Analysis and Design of Echo State Networks," *Neural Computation*, 19, 111–138.

Pinto, R., P. Engel, and M. Heinen (2011): "Echo State Incremental Gaussian Mixture Network for Spatio-Temporal Pattern Processing," .

Rachez, A. and M. Hagiwara (2014): "Language modeling using augmented echo state networks," *International Journal of Innovative Computing, Information and Control*, 10, 1969–1981.

Rigamonti, M., P. Baraldi, E. Zio, I. Roychoudhury, K. Goebel, and S. Poll (2018): "Ensemble of optimized echo state networks for remaining useful life prediction," *Neurocomputing*, 281, 121 – 138.

Robert Lengenstein, W. M. (2007): "Edge of chaos and prediction of computational performance for neural curcuit models," *Neural Networks*, 3.

Schrauwen, B., M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt (2008): "Improving reservoirs using intrinsic plasticity," *Neurocomputing*, 71, 1159 – 1171, progress in Modeling, Theory, and Application of Computational Intelligence.

Stanek, K. (2011): "Reservoir computing in financial forecasting using committee methods," *Master Thesis*.

Strauss, T., W. Wustlich, and R. Labahn (2012): "Design Strategies for Weight Matrices of Echo State Networks," *Neural computation*, 24.

Tanaka, G., T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose (2019): "Recent advances in physical reservoir computing: A review," *Neural Networks*, 115, 100 – 123.

Triesch, J. (2005): "A Gradient Rule for the Plasticity of a Neuron's Intrinsic Excitability," in *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, Berlin, Heidelberg: Springer Berlin Heidelberg, 65–70.

Verplancke, T., S. Looy, K. Steurbaut, D. Benoit, F. Turck, G. De Moor, and J. Decruyenaere (2010): "Novel time series analysis approach for prediction of dialysis in critically ill patients using echo-state networks," *BMC medical informatics and decision making*, 10, 4.

Verzelli, P., C. Alippi, and L. Livi (2019): "Echo State Networks with Self-Normalizing Activations on the Hyper-Sphere," *Scientific Reports*, 9.

Wu, Q., E. Fokoué, and D. Kudithipudi (2018): "On the Statistical Challenges of Echo State Networks and Some Potential Remedies," .

Xu, X., D. Niu, M. Fu, H. Xia, and H. Wu (2015): "A Multi Time Scale Wind Power Forecasting Model of a Chaotic Echo State Network Based on a Hybrid Algorithm of Particle Swarm Optimization and Tabu Search," *Energies*, 8, 1–21.

Yildiz, I. B., H. Jaeger, and S. J. Kiebel (2012): "Re-visiting the echo state property," *Neural Networks*, 35, 1 – 9.