KING'S BUSINESS SCHOOL

Data Analytics for Finance & Macro Research Centre

KING'S College LONDON

Deep Quantile Regression

I. Chronopoulos, A. Raftapostolos, G. Kapetanios

Working paper No. 2021/1 | April 2021 | ISSN 2516-5933

Deep Quantile Regression[*]

Ilias Chronopoulos
King's College London

Aristeidis Raftapostolos
University of Strathclyde

George Kapetanios
King's College London

April 7, 2021

Abstract

In this paper we propose a *deep quantile* estimator, using neural networks and their universal approximation property to examine a non-linear association between the conditional quantiles of a dependent variable and predictors. The proposed methodology is versatile and allows both the use of different penalty functions, as well as high dimensional covariates. We present a Monte Carlo exercise where we examine the finite sample properties of the proposed estimator and show that our approach delivers good finite sample performance. We use the *deep quantile* estimator to forecast Value-at-Risk and find significant gains over linear quantile regression alternatives, supported by various testing schemes. The paper also contributes to the interpretability of neural networks output by making comparisons between the commonly used SHAP values and an alternative method based on partial derivatives.

Keywords: Quantile regression, machine learning, neural networks, value-at-risk, forecasting.

JEL Classification: C45, C58, G17.

[*]Address correspondence to Aristeidis Raftapostolos, 199 Cathedral St, Glasgow G4 0QU, United Kingdom. Email: aristeidis.raftapostolos@strath.ac.uk.
This work was supported by the Economic and Social Research Council (ESRC) [ESRC grant $ES/P000703/1$ to Chronopoulos and ESRC grant $ES/R500938/1$ to Raftapostolos].

1 Introduction

Since the seminal work of Koenker and Bassett Jr (1978) and Koenker and Hallock (2001), quantile regression has grown in popularity and has found applications in several disciplines both in academia and industry, see e.g. Chernozhukov and Umantsev (2001), Adams, Adrian, Boyarchenko, and Giannone (2021) and Koenker, Chernozhukov, He, and Peng (2017). They generalise ordinary sample quantiles to the regression setting, that give more extensive information on the conditional distribution of a dependent variable, given the covariates, relative to the classical regression setting; i.e. estimation of the conditional mean. This extension can be of great importance under extreme events, where the conditional distribution of variables such as asset returns tends to exhibit skewness, or under the presence of outliers and/or asymmetries, see e.g. Baur and Schulze (2005).

A prevalent assumption made so far in the literature, is the linear association of the conditional quantile of the target variable and predictors. This is predominately an assumption that allows for streamlined computation and theoretical inference, but is clearly restrictive. Recent advances in machine learning (ML) literature show how modelling frameworks such as neural networks can be used to estimate general, non-linear and potentially highly complicated associations.

Specifically, a large number of studies have shown that feed-forward neural networks can approximate arbitrarily well any continuous function of several real variables, see e.g. Hornik (1991), Hornik, Stinchcombe, White, et al. (1989), Galant and White (1992) and Park and Sandberg (1991). Recent work by Liang and Srikant (2016), Mhaskar, Liao, and Poggio (2017) and Wang et al. (2018), extends this theory and present multi-layer perceptrons architectures that can approximate any function of interest to any desired degree of accuracy, provided sufficiently many hidden units and layers are available.

There is considerable empirical work identifying non-linearities and asymmetries in financial variables, see e.g. Gu, Kelly, and Xiu (2020b), Gu, Kelly, and Xiu (2020a), He and Krishnamurthy (2013) and Pohl, Schmedders, and Wilms (2018), where they illustrate that ML offers richer functional form specifications that can capture non-linearities between potential dependent and independent variables. Some examples include Gu, Kelly, and Xiu (2020b) in which, they evaluate the forecast accuracy of machine learning methods in measuring equity risk premia, and find that neural networks give substantial forecasting gains in asset pricing compared to linear models, and Bucci (2020) where a recurrent neural network, that approximates quite well the realised volatility and outperforms other classic non-linear estimators, is proposed. In a similar fashion, Smalter Hall and Cook (2017) use several neural network architectures to predict unemployment in the US and find that neural networks outperform the benchmark forecast at short horizons. In addition, Gu, Kelly, and Xiu (2020a) propose the use of a conditional Autoencoder[1], and illustrate its superior

[1]Autoencoders are an artificial neural network that can be used as a dimensionality reduction technique.

performance relative to linear unsupervised learning methods.

In this paper, we contribute to the expanding literature on the use of ML in finance and propose a novel *deep quantile* estimator that has the potential to capture non-linear associations between asset returns and predictors. ML based estimators for quantiles have been proposed in other fields, see e.g. Meinshausen (2006), where quantile random forecast are introduced, and Zhang, Quan, and Srinivasan (2018) that propose a quantile neural network estimator.

We first explore the small sample properties of the proposed estimator via Monte Carlo experiments, which show that the estimator delivers good finite sample performance. Then we examine the performance of the proposed estimator, in the context of one of the most widely examined problems in finance: that of measuring the risk of a portfolio adequately, via Value-at-Risk (*VaR*) modelling. *VaR* is a popular model that was first introduced in the late 80s and since then, has become a standard toolkit to measure market risk. It measures how much value a portfolio can lose within a given time period with some small probability, τ. *VaR* and quantiles are related in the following manner, let r be the return of a portfolio, then, τ^{th} *VaR* is equivalent of computing the negative value of the τ^{th} quantile of a portfolio return, $-q_{\tau}(r)$.

In this paper, we argue that the linear relationship between *VaR* and predictors can be restrictive and propose a quantile neural network estimator that allows a non-linear association between covariates and the *VaR*. This method appears particularly suitable for developing sound predictions for the daily stock return losses in the US over a sample from 1985 up to August 2020, the importance of which has been brought to the forefront by the recent COVID-19 pandemic. We are not the first to use ML methods for *VaR* forecasting, see e.g. Du, Wang, and Xu (2019), where they propose a Recurrent neural network, as a novel forecasting methodology for the *VaR* model and exhibit an improved forecast performance relative to traditional methods. To the best of our knowledge though, there has been no application that uses a neural network quantile estimator in finance for forecasting *VaR*.

Our empirical analysis shows that the proposed *deep quantile* estimator outperforms linear quantile regression forecasts, that assume a linear dependence between risk - return variables. We assess the forecasting accuracy between models based on two statistical criteria. The former deploys Diebold and Mariano (1995) test with Harvey, Leybourne, and Newbold (1997) correction models, while the latter presents the Giacomini and White (2006) test, both suggest that our neural network forecast are significant. Next, we use the linear quantile method as benchmark to assess whether our proposed estimator has forecast gains or not. This measure illustrate gains up to 98% relative to the linear one. Further, we deploy the quantile score test that favours our neural network estimator.

We further want to examine whether our proposed estimator nests forecasts produced from the linear models. We follow Cenesizoglu and Timmermann (2007) and adopt formal encompassing tests, that have been proposed by Giacomini and Komunjer (2005).

Overall, we find that forecasts from the *deep quantile* estimator encompass more times than the ones from the linear model. There are some cases where the test is inconclusive, and a forecast combination from both models would provide a better result, which is in line with the result of Bates and Granger (1969).

While ML methods show a great capacity at both approximating highly complicated non-linear functions and forecasting, they are under some criticism, as they lack of interpretability and are considered a "black box"; in the sense that they do not offer simple summaries of relationships in the data. Recently though, there has been a number of studies that try to make ML output interpretable, see e.g. Athey and Imbens (2017), Wager and Athey (2018), Belloni, Chernozhukov, and Hansen (2014), Joseph (2019). In this paper we also try to understand in a semi-structural fashion, which variables impact the forecasting performance of the *deep quantile* estimator more. To this end, we first use Shapley Additive Explanation Values (SHAP) as proposed by Lundberg and Lee (2017) and further developed in Joseph (2019), that have started to become a standard tool for interpretability in ML methods. Further we use partial derivatives, as a means of investigating the marginal contribution/influence of each variable to the output. We compare the partial derivatives and SHAP values over time, and our results can be summarised as follows. First, partial derivatives overall are more stable than SHAP values, and are able to produce interpretable results, at a fraction of the computation time of SHAP. Second, the partial derivatives of the *deep quantile* estimator fluctuate around the estimate of the conditional linear quantile and i) exhibit time variation and ii) can capture stressful events in the US economy for instance the COVID-19 pandemic and the 2008 financial crisis.

The remainder of the paper is organised as follows. Section 1 introduces the *deep quantile* estimator. Section 2 contains the Monte Carlo exercise. Section 3 presents our empirical application. Section 4 presents the semi-structural analysis. Conclusions are set out in Section 5.

2 Theory

In this section we start by summarising the underlying theory of a quantile regression as outlined by Koenker and Bassett Jr (1978) and Koenker (2005) and argue that the linear relationship of the conditional quantile of a dependent variable given the covariates, can be restrictive. We illustrate how some fundamental results on the universal approximation property of neural networks can be used to approximate a non-linear relationship instead, and propose a *deep quantile* estimator. We conclude with a discussion on how different penalisation schemes can be used and further how hyper-parameters can be selected via cross validation (CV).

2.1 Linear Quantile Regression

The standard goal in econometric analysis is to infer a relationship between a dependent variable and one or more covariates. Let $\{y_t, \boldsymbol{x}_t\}_{t=1}^T$ be a random sample from the following linear regression model

$$y_t = \boldsymbol{x}_t' \boldsymbol{\beta} + u_t, \tag{1}$$

where y_t is the dependent variable at time t, $\boldsymbol{\beta} = (\beta_1, \dots, \beta_N)'$ is a vector of unobserved slope parameters, $\boldsymbol{x}_t = (x_{t1}, \dots, x_{tN})'$ is a vector of known covariates, and u_t is the random error of the regression that satisfies $\mathbb{E}(u_t|\boldsymbol{x}_t) = 0$. Standard regression analysis tries to come up with an estimate of the conditional mean of y_t given $\boldsymbol{x}_t$, that minimises the expected squared error loss:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{T} \sum_{t=1}^{T} \left(y_t - \boldsymbol{x}_t' \boldsymbol{\beta}\right)^2. \tag{2}$$

This can be restrictive though, when i) non linearities and outliers exist and ii) since it provides just an aspect of the conditional distribution of y_t, given $\boldsymbol{x}_t$ by construction. These potential limitations led to the development of quantile regression. In their seminal work, Koenker and Bassett Jr (1978) generalise ordinary sample quantiles to the regression setting, that give more complete information on the conditional distribution of y_t given $\boldsymbol{x}_t$, for which we now provide a succinct description.

The quantile regression model can be defined as

$$Q_y(\tau|\boldsymbol{x}_t) = \boldsymbol{x}_t' \boldsymbol{\beta}(\tau), \quad \tau \in (0,1), \tag{3}$$

such that y_t satisfies the quantile constraint $Pr[y_t \leq \boldsymbol{x}_t' \boldsymbol{\beta}(\tau)|\boldsymbol{x}_t] = \tau$, where $\boldsymbol{\beta}(\tau)$ are regression coefficients that depend on τ. Quantile regression tries to come up with an estimate for the τ^{th} conditional quantile, $\hat{Q}_y(\tau, \boldsymbol{x}_t) \equiv \hat{\boldsymbol{\beta}}(\tau)$, by minimizing the following function

$$\hat{\boldsymbol{\beta}}(\tau) = \arg\min_{\boldsymbol{\beta}} \frac{1}{T} \sum_{t=1}^{T} \rho_\tau \left(y_t - \boldsymbol{x}_t' \boldsymbol{\beta}(\tau)\right), \tag{4}$$

where $\rho_\tau(\cdot)$ is the tick loss function defined as

$$\rho_\tau(u_t|\tau) = \begin{cases} \tau u_t(\tau), & \text{if } u_t(\tau) \geq 0 \\ (1-\tau) u_t(\tau), & \text{if } u_t(\tau) < 0 \end{cases}$$

and $u_t(\tau) = y_t - \boldsymbol{x}_t' \boldsymbol{\beta}(\tau)$. The quantile estimator in eq. (4), provides i) much richer information on the whole conditional distribution of y_t as function of the $\boldsymbol{x}_t$, and ii) more robust estimates under the presence of outliers and non linearities, when compared to the ordinary least squares estimator.

Notice that the linear association assumption, $Q_y(\tau|\boldsymbol{x}_t) = \boldsymbol{x}_t' \boldsymbol{\beta}(\tau)$, can be generally restrictive and based on simplifying assumptions. Instead, we consider the case of the following non-linear association,

$$Q_y(\tau|\boldsymbol{x}_t) = h_\tau(\boldsymbol{x}_t)$$

where $h_\tau(\cdot)$ is some unknown, (potentially highly) non-linear function. In this paper we propose an estimation strategy to approximate $h_\tau(\boldsymbol{x}_t)$ with neural networks using their universal function approximation ability. Specifically, we assume that there exists a neural network with a function $G_\tau(\boldsymbol{x}_t, \boldsymbol{w})$, to be defined below, that can approximate $h_\tau(\boldsymbol{x}_t)$ well. Before we illustrate how our methodology is implemented, we provide a discussion on how neural networks can approximate $h_\tau(\boldsymbol{x}_t)$.

2.2 Neural Networks

In this paper, we limit our attention to *feed-forward* neural networks, to approximate $h_\tau(\boldsymbol{x}_t)$. This architecture consists of an input layer of covariates, the hidden layer(s) where non-linear transformations of the covariates occur, and the output layer that gives the final prediction. Each hidden layer has several interconnected neurons relating it to both the previous and next ones. Specifically, information flows from one layer to the other, via neurons only in one direction, and the connections correspond to weights. Optimising a loss function *w.r.t* these weights makes neural networks capable of learning and inference.

Throughout our exposition, L denotes the total number of hidden layers, a measure for the depth of a neural network, and $J^{(l)}$ denotes the total number of neurons at layer l, a measure for its width. We start by presenting a general definition of a deep (multi-layer) feed-forward neural network. We assume as above, $\mathbf{x}_t = (x_{t1}, \ldots, x_{tN})'$ be the input vector where $\mathbf{x}_t \in \mathbb{R}^N$. $\sigma_l(\cdot)$, $l = 0, \ldots, L$ denotes the activation function used at l^{th} layer, that is applied element-wise and induces non-linearity. We assume that $\sigma_l(\cdot)$, $\forall l = 0, \ldots, L$ are non-constant, bounded, and continuous functions. Denote by $g^{(l)}$ the output of the l^{th} layer which is a vector of functions of length equal to the number of $J^{(l)}$ neurons in that layer, such that $g^{(0)} = \boldsymbol{x}_t$. The overall structure of the network is equal to:

$$G = g^{(L)}(g^{(L-1)}(\ldots(g^{(1)}(\cdot))))$$

where

$$g^{(l)}(\boldsymbol{x}_t) = \sigma_l\left(\boldsymbol{W}^{(l-1)} g^{(l-1)} + \mathbf{b}^l\right), \qquad \forall\, l : 1 \leq l \leq L,$$

$\boldsymbol{W}^{(l)}$ is a $J^{(l)} \times J^{(l-1)}$ matrix of weight parameters, $\boldsymbol{b}^{(l)}$ is a $J^{(l)} \times 1$ vector of so-called bias parameters giving a $J^{(l)}(1 + J^{(l-1)})$ total number of parameters in each hidden layer l, $J^{(0)} = N$ and $J^{(L)} = 1$. For the activation function $\sigma_l(\cdot)$, we use the Rectified Linear Unit (ReLU), $\sigma_l(\cdot)$=max($\cdot$, 0), for $l = 1, \ldots, L-1$ and a linear one for $l = L$.

According to various universal approximation theorems (see, e.g.,theoretical results in Hornik (1991), Hornik, Stinchcombe, White, et al. (1989), Galant and White (1992), Kapetanios and Blake (2010), Liang and Srikant (2016), Mhaskar, Liao, and Poggio (2017), Wang et al. (2018) $G(\cdot)$ can approximate arbitrarily well $h_\tau(\cdot)$, such that, for any $\epsilon > 0$,

$$\sup_x |G(x) - h_\tau(x)| < \epsilon.$$

In this sense, the above (ϵ)-approximation can be seen as a sieve type non-parametric estimation bound, where ϵ can become arbitrarily small by increasing the complexity of $G(\boldsymbol{\cdot})$.

The increase in complexity can occur, either by letting $L \to \infty$, which stands for *deep learning*, or by letting $J^{(l)} \to \infty$. While asymptotically, both ways deliver the same results (see e.g. Farrell, Liang, and Misra (2021) and references therein), the approximation error has been shown to decline exponentially with L, but only polynomially with $J^{(l)}$, providing some evidence for the prevalent use of deep learning. As an illustration, in Figure 1 we depict a *feed-forward* neural network with two inputs, two hidden layers, a total of five neurons and one output layer.

Figure 1 about here

2.3 Non-linear Quantile Regression

We assume that the conditional quantile follows a non-linear relationship $Q_y(\tau|\boldsymbol{x}_t) = h_\tau(\boldsymbol{x}_t)$ and there exists a function $G_\tau(\cdot)$, that can (ϵ)-approximate $h_\tau(\cdot)$ in sup norm. Using this assumption, we define the conditional quantile function as

$$Q_y(\tau|\boldsymbol{x}_t) = G_\tau(\boldsymbol{x}_t, \boldsymbol{w}),$$

where $G_\tau(\boldsymbol{x}_t, \boldsymbol{w})$ is the unknown non-linear function that can approximate $h_\tau(\boldsymbol{x}_t)$. We obtain the deep neural network conditional quantile estimate from the solution of the following minimization problem:

$$Q_y(\tau|\boldsymbol{x}_t) = \arg\min_{\boldsymbol{w}} \frac{1}{T} \sum_{t=1}^{T} \rho_\tau \left(y_t - G_\tau(\boldsymbol{x}_t, \boldsymbol{w}) \right), \tag{5}$$

where $\boldsymbol{w} = (\text{vec}(\boldsymbol{W}^{(0)})', \ldots, \text{vec}(\boldsymbol{W}^{(L)})', \boldsymbol{b}^{(1)'}, \ldots, \boldsymbol{b}^{(L)'})'$ contains all model parameters, and G_τ denotes the overall non-linear mapping.

2.4 Regularised Non-Linear Quantile Regression

Neural networks have a great capacity to extract non-linear features from the data, but this comes at a cost, since they are prone to over-fitting. This can lead to a severe drop in their forecasting performance, especially in small samples. There is a variety of commonly used techniques in ML, see e.g. Gu, Kelly, and Xiu (2020a) for a good summary, that can be used to ease this impact, originally coming from the high-dimensional statistical literature.

2.4.1 Regularisation

A common solution to this caveat is regularisation, where a penalty term is imposed on the weights of the neural network and is appended in the loss function. Regularisation, generally improves the out-of-sample performance of the network by decreasing the in-sample noise from over-parametrisation, utilising the bias-variance trade-off. Further, another benefit of regularization is that it provides computational gains in the optimization algorithm. The penalised loss function, for a given quantile τ, can be written as:

$$L(G_\tau(\boldsymbol{x}_t, \boldsymbol{w}), y_t) = \frac{1}{T}\sum_{t=1}^{T} \rho_\tau(y_t,\ \hat{G}_\tau(\boldsymbol{x}_t, \boldsymbol{w})) + \phi(\boldsymbol{w}), \tag{6}$$

where the penalty term is

$$\phi(\boldsymbol{w}) = \begin{cases} \lambda\,||\boldsymbol{w}||_1, & \text{Lasso} \\ \lambda\,||\boldsymbol{w}||_2^2, & \text{Ridge} \\ \lambda(1-\alpha)||\boldsymbol{w}||_1 + \lambda\alpha||\boldsymbol{w}||_2^2, & \text{Elastic Net} \\ 0, & \text{otherwise} \end{cases},$$

and λ and α are tuning parameters, for which we discuss their estimation below. Generally, there is a plethora of loss functions, and the choice among them, depends mainly on the task at hand. In this paper we use the tick loss function. The different penalisation schemes on $\phi(\boldsymbol{w})$ work as follows: Lasso or l_1-norm penalisation, is a regularisation method that shrinks uniformly all the weights to zero, and some at exactly zero. The latter is referred to as the variable selection property of the Lasso. Ridge works in a similar manner to the Lasso, by shrinking the weights, uniformly to zero, but not at exactly zero. Finally, the Elastic Net is a combination of Lasso and Ridge, that has been shown to retain good features from both methods, see e.g. Zou and Hastie (2005). In our appendix we provide details about the algorithm used to train our neural network models.

2.4.2 Cross Validation

We use cross validation to calibrate all the different (hyper)-parameters outlined above, and aim to maximise the out-of-sample (forecasting) performance of the network. The CV exercise boils down to a choice on, i) the total number of layers and neurons, ii) the learning rate for the Stochastic Gradient Decent (SGD), iii) the batch size, batch normalization, the level of regularization, and finally, a choice on the activation function. To that end, our aim is to build a neural network that has the best pseudo-out-of-sample (POOS) performance. To achieve this, we simply need to evaluate the model, select the optimal parameter and hyper-parameters and test its POOS behaviour. It is clear that tuning all these different architectures, parameters and hyper-parameters increases the computational cost a lot.

For this reason we tune the learning rate, γ, for the optimiser from a grid containing $[0.01, 0.001, 0.0001]$. We further fix the hyper-parameters of hidden layers as well as the batch size and activation functions, throughout our evaluation experiments. As we include a limited number of predictors we do not use batch normalization. Furthermore, we tune the regularisation parameter, λ, from a grid in $[0, 0.01, 0.001, 0.0001]$, both for Ridge and Lasso, and for the case of the elastic net we choose α from a grid in $[0, 0.1, 0.5, 0.9]$.

We split the whole sample into two subsamples, namely the training and test sample. Further, we hold out the last 10% of the training sample as a validation sample. First, we use the training sample to estimate (i.e. train) the network parameters, given a specific sequence of hyper-parameters. Then, the second subsample or validation is used to tune hyper-parameters by constructing the fitted/forecasted values given the parameters from train sample. We proceed with the calculation of the tick loss function as in eq. (6) and evaluate model's in-sample out-of-sample i.e. POOS performance on this subset. We repeat the same process a k number of times[2], we store the tick loss values on the validation subset for all k-times and we select the parameters and hyper-parameters that minimise the loss. Throughout the validation step we wish to find the optimal parameter and hyper-parameters that capture complex non-linear relations and produce reliable POOS forecasts.

Finally, in the test subsample we use the optimal parameters and hyper-parameters from validation step and evaluate the out-of-sample performance/forecasting ability of the network.

2.4.3 Optimisation

The estimation of neural networks is generally a computational cumbersome optimization problem due to non-linearities and non-convexities. The most commonly used solution utilises stochastic gradient descent (SGD) to train a neural network. SGD uses a *batch* of a specific size, that is, a small subset of the data at each iteration of optimization to evaluate the gradient, to alleviate the computation hurdle. The step of the derivative at each epoch is controlled by the learning rate. To avoid any noise in the evaluation of the SGD we adopt the adaptive moment estimation algorithm (ADAM) proposed by Kingma and Ba (2014)[3]. In the appendix we offer details on the workings of the optimisation algorithm used.

[2]$k = 48$ is the number of all possible grid search combinations per quantile.
[3]ADAM is using estimates for the first and second moments of the gradient to calculate the learning rate.

3 Monte Carlo

3.1 Setup

In this section we present Monte Carlo (MC) experiments, in order to study the finite sample performance of the *deep quantile* estimator, as proposed in Section 2, for the different penalisation schemes. We generate artificial data $\{y_t\}$ using a single predictor $\{x_t\}$, according to the following model

$$y_t = h_\tau(x_t) + u_t, \tag{7}$$

where u_t is the realisation of a random variable u distributed as, $u_t \sim iidN(-\sigma\Phi^{-1}(\tau), \sigma^2)$, $\sigma = 0.1$ and Φ^{-1} is the quantile function of the standard normal distribution. $h_\tau(\cdot)$ is the general non-linear function that we wish to approximate via the *deep quantile* estimator.

All the experiments are based on the following values: $\tau \in (0.01, 0.025, 0.05, 0.1, 0.2)$, $T \in (100, 300, 500, 1000, 2000, 5000)$ and the number of MC replications is 100. We consider the following four *data generating mechanisms* (DGM) to assess the finite sample properties of the *deep quantile* estimator:

Case I: We consider the case of a $N(0, 1)$ simulated single predictor that is generated as follows

$$\begin{aligned} y_t &= h_\tau(x_t) + u_t, \\ h_\tau(x_t) &= \sin(2\pi x_t), \\ x_t &\sim N(0, 1). \end{aligned}$$

This is the simplest design in our Monte Carlo experiments. The reason we use this simple case, is to showcase that linear methods, as expected, cannot produce reasonable inference under a sigmoid type of a non-linear function $h_\tau(\cdot)$.

Case II: We consider an AR(1) simulated single predictor as follows

$$\begin{aligned} y_t &= h_\tau(x_t) + u_t, \\ h_\tau(x_t) &= \sin(2\pi x_t), \end{aligned}$$

where x_t is simulated as

$$x_t = 0.8x_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, 1)$$

In this design we increase the complexity by introducing a correlated predictor, which is a more realistic design for asset returns.

Case III: We consider the case of a single predictor generated via a GARCH(1,1) model

$$\begin{aligned} y_t =& h_\tau(x_t) + u_t, \\ h_\tau(x_t) =& \sin(2\pi x_t), \end{aligned}$$

where x_t is simulated as:

$$\begin{aligned} x_t =& \sigma_t \varepsilon_t \\ \sigma_t^2 =& 1 + 0.7x_{t-1}^2 + 0.2\sigma_{t-1}^2. \end{aligned}$$

In this design, we wish to examine, how the proposed estimator fares, when the regressor is conditionally heteroskedastic, following a $GARCH(1,1)$ model. Similar to Case II, a $GARCH$ type of assumption on the distribution of asset returns is one commonly used in the literature.

Case IV: We consider the case of a single predictor that is generated as follows:

$$\begin{aligned} y_t =& h_\tau(x_t) + u_t, \\ h_\tau(x_t) =& G(x_t, \boldsymbol{w}) \\ x_t \sim& N(0,1). \end{aligned}$$

In this case we simulate $h_\tau(x_t)$ to reflect a function composition used in neural networks. We simulate it with 3 hidden layers and specific number of neurons, such as

$$G(x_t, \boldsymbol{w}) = \left(\boldsymbol{W}^{(3)}\left(\sin\left(\boldsymbol{W}^{(2)}\left(\sin\left(\boldsymbol{W}^{(1)}\left(\sin\left(\boldsymbol{W}^{(0)}x_t' + \boldsymbol{b}^{(1)}\right)\right) + \boldsymbol{b}^{(2)}\right)\right) + \boldsymbol{b}^{(3)}\right)\right)\right)',$$

where $\boldsymbol{w} = (\text{vec}(\boldsymbol{W}^{(0)})', \ldots, \text{vec}(\boldsymbol{W}^{(3)})^{(L)})', \boldsymbol{b}^{(1)'}, \ldots, \boldsymbol{b}^{(3)'})'$, $\boldsymbol{W}^{(0)}$ is 50×1, $\boldsymbol{W}^{(1)}$ is 10×50, $\boldsymbol{W}^{(2)}$ is 8×10 and $\boldsymbol{W}^{(3)}$ is 1×8. Further, for the weights, $\boldsymbol{w}$, we assume that they are sparse, so that, every entry $w_{i,j}$ is simulated as, $w_{i,j} = \delta_{i,j} 1(\delta_{i,j} > 0.5)$, where $\delta_{i,j} \sim U(0,1)$.

Across all cases, we estimate $h_\tau(x_t)$ using our proposed estimator with different penalisation schemes. Let $\hat{h}_{\tau,pen} = \hat{G}_{\tau,pen}(x_t, \boldsymbol{w})$ denote the estimate, where pen $= [0, \text{l1}, \text{l2}, \text{ElNet}]$[4] illustrates which penalty has been used.

We report the following metrics to evaluate the small sample properties, of our *deep quantile* estimator across R MC replications: i) the average mean squared error of the true residuals, $AMSE_{u_t} = \frac{1}{R}\frac{1}{T}\sum_{i=1}^{R}\left(\sum_{t=1}^{T} u_t^2\right)_i$, ii) the average mean squared error of the estimated residuals, $AMSE_{\hat{u}_t,pen} = \frac{1}{R}\frac{1}{T}\sum_{i=1}^{R}\left(\sum_{t=1}^{T}(y_t - \hat{y}_{t\,pen})^2\right)_i$ and finally, iii) the average absolute bias $ABIAS_{\hat{h}_{\tau},pen} = \frac{1}{R}\frac{1}{T}\sum_{i=1}^{R}\left(\sum_{t=1}^{T}\left|(h_\tau(x_t) - \hat{G}_{\tau,pen}(x_t, \boldsymbol{w}))\right|\right)_i$.

Table 1 about here

Table 2 about here

[4]They correspond to no penalisation, Lasso, Ridge and Elastic net, respectively.

Table 3 about here

Table 4 about here

3.2 Results

We present our Monte Carlo results in Tables 1 - 4. Across cases, our Monte Carlo results suggest that the *deep quantile* estimator has good finite sample properties, and can approximate non-linear functions. We further find, as expected, that the linear quantile regression estimator, does not work under non-linearity. Finally, we find evidence in favour of the penalisation schemes proposed in Section 2. Specifically, the penalised *deep quantile* estimator also has good finite sample properties, and in some cases, performs better that the non-regularised one; a finding in favour of weight regularisation.

4 Empirical Setup

In this section we outline our empirical application setup, where we use the proposed *deep quantile* estimator to forecast Value at Risk (*VaR*). We examine the predictive ability of the proposed estimator, relative to the linear one, using the quantile encompassing test of Giacomini and Komunjer (2005). We further examine the predictive performance of our models by testing their forecasting accuracy, using the Diebold and Mariano (1995), Giacomini and White (2006) and quantile score tests.

4.1 Deep Quantile *VaR* forecasting

The data used in our empirical application consist of around 36 years of daily returns on the S&P500 index (source: Bloomberg), from September 1985 to August 2020 (T = 9,053 observations). We use four different classes of *VaR* models and produce forecasts for the $10\%, 5\%$ and 1% empirical conditional quantiles, using the *deep quantile* estimator.

The first *VaR* specification we consider is the GARCH(1,1) that has been proposed by Bollerslev (1986), in which $\sigma^2_{1,t} = \omega_0 + \omega_1\sigma^2_{1,t-1} + \omega_2 r^2_{t-1}$, see eq. (8). The second *VaR* specification we examine, is RiskMetrics, proposed by J.P. Morgan (1996), which assumes $\sigma^2_{2,t} = \lambda\sigma^2_{2,t-1} + (1-\lambda)r^2_{t-1}$, where for daily returns, $\lambda = 0.94$, eq. (9).

The last two specifications we consider follow the *Conditional Autoregressive Value-at-Risk* model (CAViaR), proposed by Engle and Manganelli (2004), where a specific quantile is analysed, rather than the whole distribution. Specifically, the CAViaR model corrects the past $VaR_{j,t-1}$ estimates in the following way: it increases $VaR_{j,t}$ when $VaR_{j,t-1}$ is above the τ^{th} quantile, while, when the $VaR_{j,t-1}$ is less than the τ^{th} quantile, it reduces $VaR_{j,t}$. Thus, the third *VaR* we examine is the Symmetric absolute value (SV) that responds symmetrically to past returns, see eq. (10) and lastly, we consider the Asymmetric slope value (ASV) as it offers a different response to positive and negative returns, see eq. (11). For ease of exposition, we refer to $VaR_{1,t}$, $VaR_{2,t}$, $VaR_{3,t}$ and $VaR_{4,t}$ as GARCH, RM (RiskMetrics), SV and ASV, respectively. Below we summarise their specifications:

$$VaR_{1,t} = \beta_0 + \beta_1 \sigma_{1,t} \tag{8}$$
$$VaR_{2,t} = \beta_0 + \beta_1 \sigma_{2,t} \tag{9}$$
$$VaR_{3,t} = \beta_0 + \beta_1 VaR_{3,t-1} + \beta_2 |r_{t-1}| \tag{10}$$
$$VaR_{4,t} = \beta_0 + \beta_1 VaR_{4,t-1} + \beta_2 r^{+}_{t-1} - \beta_3 r^{-}_{t-1}, \tag{11}$$

where β_i, $i = 0, \dots, 2$ are parameters to be estimated. We use these specifications following Giacomini and Komunjer (2005).

As discussed in Section 2, the linear association between *VaR* and the covariates can be restrictive. Instead we assume that the relationship between the response variable, *VaR*, and the covariates has an unknown non-linear form for a given τ, that we wish to approximate with our proposed *deep quantile* estimator as

$$VaR_{1,t} = G_\tau(\sigma_{1,t}, \boldsymbol{w}) \tag{12}$$
$$VaR_{2,t} = G_\tau(\sigma_{2,t}, \boldsymbol{w}) \tag{13}$$
$$VaR_{3,t} = G_\tau(VaR_{3,t-1}, |r_{t-1}|, \boldsymbol{w}) \tag{14}$$
$$VaR_{4,t} = G_\tau(VaR_{4,t-1}, r^{+}_{t-1}, r^{-}_{t-1}, \boldsymbol{w}), \tag{15}$$

where $VaR_{j,t}$, $j = 1, \dots, 4$ is indexed at (day) $t = 1, \dots, T$. We include more than one covariates, depending on model j.

4.2 Forecasting Exercise Design

This section presents our forecasting exercise design. We use the first $6,348$ observations as a train sample and the remaining 705 observations to tune our hyper-parameters. We fix the number of layers, nodes, the batch size and activation functions and tune the learning rate and the regularization hyper-parameters. We use the validation sample to tune hyper-parameters based on the minimisation of the tick loss function, and finally we evaluate the forecasting performance of our models on the test sample. Generally, a forecasting exercise is performed either via a recursive or rolling window, yet in either setting to produce all one step ahead forecasts for the last 2,000 observations and to tune the hyper-parameters can be computationally challenging. Instead, we follow Giacomini and Komunjer (2005) and perform a fixed forecast window exercise, in which we estimate our models once. Specifically, we use data up to time n (denotes the end of the in-sample size) to estimate the unknown parameter vector β_i, $i = 0, \dots, 2$, then we use the estimates $\hat{\beta}_i$, $i = 0, \dots, 2$

throughout the out-of-sample exercise. We evaluate the forecasting performance of *VaR* models with the proposed *deep quantile* estimator as in Section 2.

4.3 Forecast Evaluation

In this section we discuss the various tests we have considered, in order to evaluate the predictive ability of the *deep quantile* estimator and present testing results.

4.3.1 Conditional Quantile Forecast Encompassing (CQFE)

We present the implementation of the CQFE test as proposed by Giacomini and Komunjer (2005) and the Generalized Method of Moments (GMM) estimation as proposed by Hansen (1982). Let $\hat{q}_{1,t}$ be a vector of τ-quantile forecasts produced from model 1 and $\hat{q}_{2,t}$ be the competing forecasts produced from model 2. The basic principle of CQFE is to test whether $\hat{q}_{1,t}$ conditionally encompasses $\hat{q}_{2,t}$. Encompassing occurs when the second set of forecasts fail to add new information to the first set of quantile forecasts (or vice versa) in which case the first (second) quantile forecast is said to encompass the second (first).

The aim of the CQFE test is to test the null hypothesis, that $\hat{q}_{1,mt}$ performs better that any linear combination of $\hat{q}_{1,mt}$ and $\hat{q}_{2,mt}$. Under the null hypothesis, it holds

$$E_t\left(\rho_\tau(y_{t+1} - \hat{q}_{1,mt})\right) \leq E_t\left(\rho_\tau(y_{t+1} - \theta_0 - \theta_1\hat{q}_{1,mt} - \theta_2\hat{q}_{2,mt})\right), \tag{16}$$

that is satisfied if and only if the weights (θ_1, θ_2) are equal to $(1, 0)$. The objective function of the GMM is:

$$J_T = g_T(\boldsymbol{\theta})'\boldsymbol{W}_T g_T(\boldsymbol{\theta}).$$

The optimal weights are computed:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} g_T(\boldsymbol{\theta})'\boldsymbol{W}_T g_T(\boldsymbol{\theta}),$$

where

$$g_T(\boldsymbol{\theta}) = \frac{\sum_{t=1}^{T}(\tau - \mathbb{1}_\tau\{y_{t+1} - \boldsymbol{\theta}'\boldsymbol{q}_{mt} < 0\})\,\boldsymbol{z}_T}{T},$$

where $\boldsymbol{W}_T$ is a positive definite matrix, $g_T(\boldsymbol{\theta})$ is the sample moment condition, $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2)'$ is a set of weights, $\boldsymbol{\theta}^\star = (\theta_0^\star, \theta_1^\star, \theta_2^\star)'$ denotes the optimal weights, $\hat{\boldsymbol{q}}_{mt} = (1, \hat{q}_{1,mt}, \hat{q}_{2,mt})'$ a vector with the forecasted values based on the pairwise models 1, and 2 in the CQFE test, m denotes the out-of-sample size and $\boldsymbol{z}_T$ is a vector of instruments. Hansen (1982) showed that by setting $\boldsymbol{W}_T = \boldsymbol{S}_T^{-1}$ i.e the inverse of an asymptotic covariance matrix, is optimal as it estimates $\boldsymbol{\theta}^\star$ with as small as possible asymptotic variance. $\boldsymbol{S}$ is also know as the spectral density matrix of $\boldsymbol{g}_T$. We follow Newey and West (1987) and use a heteroskedasticity robust estimate $\hat{\boldsymbol{S}}_T$, of $\boldsymbol{S}$ defined as:

$$\hat{\boldsymbol{S}}_T = \hat{\boldsymbol{S}}_0 + \sum_{j=1}^{m} \left(1 - \frac{j}{m+1}\right) \left(\hat{\boldsymbol{S}}_j + \hat{\boldsymbol{S}}_j'\right),$$

where

$$\hat{\boldsymbol{S}}_j = \frac{1}{T} \sum_{t=j+1}^{T} g_t(\hat{\boldsymbol{\theta}}) g_{t-j}(\hat{\boldsymbol{\theta}}).$$

$\hat{\boldsymbol{S}}_0$ is the estimated spectral density matrix evaluated at frequency zero. The GMM estimation is performed recursively, i.e. i) minimize J_T using an identity weighting matrix to get $\boldsymbol{\theta}^\star$, which give $\boldsymbol{W}_T$ via $\hat{\boldsymbol{S}}_T$ and ii) minimize J_T using $\boldsymbol{W}_T = \hat{\boldsymbol{S}}_T^{-1}$ from step i).

Consequently, we consider two separate test $H_{10} : (\theta_1^\star, \theta_2^\star) = (1,0)$ versus $H_{1a} : (\theta_1^\star, \theta_2^\star) \neq (1,0)$ and $H_{20} : (\theta_1^\star, \theta_2^\star) = (0,1)$ versus $H_{2a} : (\theta_1^\star, \theta_2^\star) \neq (0,1)$, which correspond to testing whether forecast $\hat{q}_{1,mt}$ encompasses $\hat{q}_{2,mt}$ or $\hat{q}_{2,mt}$ encompasses $\hat{q}_{1,mt}$. Then the CQFE statistics are defined as:

$$ENC1 = T\left((\theta_1^\star, \theta_2^\star) - (1,0)\right) \hat{\boldsymbol{\Omega}} \left((\theta_1^\star, \theta_2^\star) - (1,0)\right)' \tag{17}$$

$$ENC2 = T\left((\theta_1^\star, \theta_2^\star) - (0,1)\right) \hat{\boldsymbol{\Omega}} \left((\theta_1^\star, \theta_2^\star) - (0,1)\right)', \tag{18}$$

where

$$\hat{\boldsymbol{\Omega}} = g_T(\boldsymbol{\theta})' \boldsymbol{S}^{-1} g_T(\boldsymbol{\theta}).$$

The asymptotic distribution of the GMM estimates of $\boldsymbol{\theta}$ require the moment conditions to be once differentiable. To satisfy this requirement, we follow Giacomini and Komunjer (2005) and replace the moment condition with the following smooth approximation:

$$g_\tau(\boldsymbol{\theta}) = \frac{\sum_{t=1}^{T} \left[\tau - (1 - exp((y_{t+1} - \boldsymbol{\theta}'\hat{\boldsymbol{q}}_{mt})/\eta))\right] \mathbb{1}\{y_{t+1} - \boldsymbol{\theta}'\hat{\boldsymbol{q}}_{mt} < 0\}) \boldsymbol{z}_T}{T},$$

where η is the smoothing parameter. We choose the critical values, c_{crit} of the test from a χ^2 distribution, in which $\hat{q}_{i,mt}$ encompasses $\hat{q}_{j,mt}$, if $ENC_i \leq c_{crit}$ $\forall i \neq j = 1,2$. In the empirical application, the vector of instruments, $\boldsymbol{z}_T$ is $(1,\, r_t,\, VaR_{i,t},\, VaR_{j,t}) \forall\, i \neq j = 1,2$.

We select η to be 0.005, following the CQFE rejection probabilities in Giacomini and Komunjer (2005), since our POOS size is 2,000 observations. Table 5 is split into two blocks, the former checks how many times the linear quantile estimator wins or loses in comparison to *deep quantile* methods, while the latter depicts the comparison between *deep quantile* methods and linear quantile regression. Under this setting a *win* denotes that the prevailing model encompasses the competing benchmark model, while a *loss* means that the benchmark encompasses the prevailing model. Precisely, we consider a *win* when the computed p-value of CQFE test fails to reject the null hypothesis, i.e. H_{10} and H_{20}. On the contrary, in case that the CQFE test suggests that there is no encompassing between the forecasts, we consider this as a *loss*, i.e. the null hypothesis is rejected. Furthermore, the CQFE test has a grey zone in which the test can fail to reject both null hypotheses (H_{10} and H_{20}), hence the test is inconclusive.

The following results are presented in Table 5. When the smoothing parameter, η, is 0.005 and we examine the 10th quantile, the *deep quantile* estimator *wins* 23 times versus 13 times for the linear models. The linear models *loses* 51 times, in comparison to the *deep quantile* estimator, which loses 41 times. Additionally for the *deep quantile* estimator the test is inconclusive 116 times in comparison to the 11 times for the linear models. For the 5th quantile, the *deep quantile* estimator *wins* 25 times versus 12 times for the linear models. Further, the linear model *loses* 52 times in comparison to 39 times for the *deep quantile* estimator. Finally, for the *deep quantile* estimator, the CQFE test is inconclusive 103 times in comparison to the linear models, where the test is inconclusive 5 times.

In addition, we examine the 1st quantile with the aforementioned smoothing parameter. The *deep quantile* estimator *wins* 37 times versus 27 times for the linear models. Furthermore, the linear model *loses* 37 times in comparison to 27 times for the *deep quantile* estimator. Finally, for the *deep quantile* estimator the test is inconclusive 147 times in comparison to its linear counterpart, which is 23 times. Results for different smoothing parameters η suggest similar patterns and are available upon request.

Table 5 about here

4.3.2 Diebold Mariano Test

We perform a quantitative forecast comparison and test their statistical significance. To do so, we calculate the *Root Mean Squared Forecast error* (RMSFE) metric and perform the Diebold and Mariano (1995) (DM) test, with the Harvey, Leybourne, and Newbold (1997) adjustment to gauge the statistical significance of the models forecasting ability. With the DM test, we assess the forecast accuracy of the *deep quantile* estimator relative to the benchmark linear quantile regression model. In this exercise we set τ equal to 10%, 5% and 1%.

In general, *Root Mean Squared Forecast error* (RMSFE) is used to measure the accuracy of point estimates. RMSFE is defined as

$$RMSFE = \sqrt{\frac{\sum_{t=1}^{T}(y_{t+h} - \hat{G}_\tau\left(\boldsymbol{x}_{t+h}, \boldsymbol{w}\right))^2}{T}},$$

where h denotes the forecast horizon and $\hat{G}_\tau\left(\boldsymbol{x}_{t+h}, \boldsymbol{w}\right)$ is the solution to the eq. (5) after selecting the optimal $\boldsymbol{w}$ via CV at the τ^{th} quantile. Throughout our application $h = 1$. Results from the DM test are reported in Table 6, where asterisks denote the statistical significance of rejecting the null hypothesis of the test at 10%, 5% and 1% level of significance, for all quantiles and models we consider. These results suggest that forecasts produced using the non-linear estimator outperform forecasts obtained from the linear quantile regression estimator we considered.

4.3.3 Giacomini White Test

In a similar manner and to complement the DM test, we follow Carriero, Kapetanios, and Marcellino (2009) and further calculate the Giacomini and White (2006) test of equal forecasting accuracy, that can handle forecasts based on both nested and non-nested models, regardless of the estimation procedures used for the derivation of the forecasts, including our proposed *deep quantile* estimator. Table 7 illustrates the results for Giacomini and White (2006) test, where asterisks denote the statistical significance of rejecting the null hypothesis of the test at 10%, 5% and 1% level of significance, for all quantiles and different models we consider. Similarly to the DM forecasting accuracy test, the Giacomini and White (2006) test is again significant at 1% in most cases, with the following exceptions. Deep ridge in ASV model and $\tau = 0.1$ is significant at 5%. In the SV model for $\tau = 0.05$, only the deep ridge predictions are significant at 1%, while the rest are insignificant. For $\tau = 0.01$ the *deep quantile* is significant for SV and ASV model at 5%. Further, deep ridge for SV model is significant at 10% and finally, the deep Lasso and elastic net forecasts are not significant.

Table 6 about here

Table 7 about here

4.3.4 Forecast Gains

We proceed with the evaluation of the forecast gains/losses between deep quantile methods against the benchmark linear ones. We follow Carriero, Clark, and Marcellino (2015) and report the percentage gains metric defined as:

$$100\left(1 - \frac{RMSFE^M}{RMSFE^B}\right),$$

where $RMSFE^M$ comes from the competing model and $RMSFE^B$ from the linear QR model. Figure 2 shows the forecast gains. Positive values show evidence in favour of our *deep quantile* estimators, while negative ones favour linear quantile regression. Our *deep quantile* methods dominate the linear quantile estimators, with considerable forecasting gains that fluctuate between 60% and 98%. In many of the cases, the use of regularization in the *deep quantile* estimator improves the forecasting performance. Finally, the forecast gains of the ASV model are the highest relative to the rest of the models.

Figure 2 about here

4.3.5 Quantile Score

Another measure to assess the forecasting performance of the *deep quantile* estimator is the average Quantile Score (QS). The quantile score takes only two possible values and examines whether or not the predicted values exceed the quantile τ. QS is defined as follows

$$QS = \frac{\sum_{t=1}^{T} (\rho_\tau \left(y_{t+h} - \hat{G}_\tau \left(\boldsymbol{x}_{t+h}, \boldsymbol{w}\right)\right)}{T}.$$

The quantile score of the forecasts from the *deep quantile* estimator is compared with the score of the linear model. Positive values show evidence in favour of the *deep quantile* estimator, while negative values support the linear estimator. Table 8 presents the quantile score results. We observe that in all cases the *deep quantile* estimator prevails to the competing linear estimator except for RM model when τ is 0.05 and 0.01 for *deep quantile*, deep ridge and deep elastic net. Finally, the linear model prevails when $\tau = 0.01$ and GARCH *VaR* specification is considered in *deep quantile*, deep ridge and deep elastic net models.

Table 8 about here

5 Semi-Structural analysis

A general issue in ML is the trade-off between accuracy and interpretability; where the output of a highly complicated model, e.g. a deep neural network, can have great accuracy or forecasting performance, but cannot be easily interpreted. In this section we first discuss the details of two methods that can be used to make ML methods interpretable. The first one is the Shapley Additive Explanation Values (SHAP), that has received a lot of attention recently, and the second is partial derivatives. Further we make a formal comparison on the output of both methods, based on the output of the *deep quantile* estimator that illustrates, i) that both methods can be used to make the impact of each covariate in neural networks interpretable and ii) perhaps surprisingly that the use of partial derivatives, offers more stable results at a fraction of the computational cost.

5.1 SHAP values

SHAP values are a general class of additive attribution methods, based on Shapley values; initially developed by Shapley (1953) to determine how to fairly split a pay-off among players in a cooperative game. In the context of ML, the goal of SHAP values is to explain the prediction of the dependent variable by estimating the contribution of each covariate to the prediction. SHAP values, following the exposition in Lundberg and Lee (2017) and Lundberg, Erion, and Lee (2018) can be constructed as follows.

Let $f(\boldsymbol{x}_t) = \hat{G}(\boldsymbol{x}_t, \boldsymbol{w})$ be the output of the estimated model we wish to interpret, given a $N \times 1$ vector of covariates $\boldsymbol{x}_t$, and $\hat{f}$ the explanation model, to be defined below. Further, let $\boldsymbol{x}_t^\dagger$ be the $M \times 1$ subset (vector) of $\boldsymbol{x}_t$ that contains simplified covariates. These simplified covariates, can be mapped to the original through a mapping function $h_{\boldsymbol{x}_t}(\cdot)$, such that $\boldsymbol{x}_t = h_{\boldsymbol{x}_t}(\boldsymbol{x}_t^\dagger)$. Then under the local accuracy property of Lundberg and Lee (2017), if there exists a $\boldsymbol{z}_t^\dagger$ with binary inputs, such that $\boldsymbol{z}_t^\dagger \approx \boldsymbol{x}_t^\dagger$, then $\hat{f}(\boldsymbol{z}_t^\dagger) \approx f(h_{\boldsymbol{x}_t}(\boldsymbol{z}_t^\dagger))$, where the explanation model (i.e. the additive attribution function) is

$$\hat{f}(\boldsymbol{z}_t^\dagger) = \phi_0 + \sum_{i=1}^{M} \phi_i z_{t,i}^\dagger, \tag{19}$$

and $\hat{f}(\boldsymbol{z}_t^\dagger)$ represents the linear decomposition of the original ML model, where ϕ_0 is the intercept, $\phi_i \in \mathbb{R}$ is the effect to each dependent variable $\boldsymbol{z}_t^\dagger \in (0,1)$, that provides local and global inference at the same time. If $z_{t,i} = 1$ then the covariate is observed, on the contrary, if $z_{t,i} = 0$ then the covariate is unknown. Under the following three properties: i) local accuracy i.e. the explanation function should match the original model, ii) missingness, ensures that input variable have no attributed effect and iii) consistency, if an input variables is important, then the effect to each dependent variable should not decline, the SHAP value is

$$\phi_i = \sum_{M \subseteq N \setminus \{i\}} \frac{|M|!\,(N - |M| - 1)!}{N!} \left[f_{M\cup\{i\}}\left(x_{M\cup\{i\}}\right) - f_M(x_M) \right], \tag{20}$$

where N is the set of all predictors, $|M|$ is the number of non-zero elements in $\boldsymbol{x}_t^\dagger$, $f_M(x_M)$ is the model's output using except from the i^{th} covariate, and $f_{M\cup\{i\}}\left(x_{M\cup\{i\}}\right)$ is the output of the model, when $\{i\}$ is included in the covariate set .

The calculation of SHAP values can be computationally expensive, as it requires 2^N possible permutations of the predictors. For the case of deep neural networks Lundberg and Lee (2017), and Shrikumar, Greenside, and Kundaje (2017), have shown that *DeepLIFT* can be used as an approximation of the deep SHAP that is computationally feasible[5], preserving the three properties above. *DeepLIFT* is a recursive prediction explanation method for deep learning. The Additive feature attribution methods analogy of *DeepLIFT* is called the summation-to-delta property is

$$\sum_{i=1}^{N} C_{\Delta x_{t,i} \Delta o} = \Delta o. \tag{21}$$

Then the SHAP values can be obtained as

$$\phi_i = C_{\Delta x_{t,i} \Delta o},$$

where $C_{\Delta x_{t,i} \Delta o}$, represents the impact of a covariate to a reference value relative to the initial value, is assigned to each $x_{t,i}$ covariate, $o = f(\cdot)$ is the output of the model, $\Delta o = f(x) - f(r)$, $\Delta x_{t,i} = f(x_{t,i}) - r_{t,i}$ and r the reference value. Eq. (21) matches eq. (19), if in Δo we set $\phi_0 = f(r_{t,i})$ and $\phi_i = C_{\Delta x_{t,i} \Delta o}$.

[5]There are other methods to achieve this, such as Tree Explainer, Kernel Explainer, Linear Explainer, Gradient Explainer.

5.2 Partial Derivatives

The use of partial derivatives on the interpretation of a model is straight forward in econometrics, with various uses, ranging from the simple linear regression model to impulse response analysis. In this section we show how partial derivatives can be used even in highly non-linear deep neural networks. Before we start the analysis, note that while the deep neural networks are highly non linear, their solution/output via SGD optimization methods is always differentiable.

For a general $\boldsymbol{x}_t \in \mathbb{R}^N$, let

$$d_{j,i,t} = \frac{\partial \hat{G}_{j,\tau}(\boldsymbol{x}_t, \boldsymbol{w})}{\partial x_{j,i,t-1}}, \tag{22}$$

denote the partial derivative of covariate $x_i = x_{it}$, for $i = 1, \ldots, N$ at time $t = 1, \ldots, T$, $\hat{G}_{j,\tau}(\boldsymbol{x}_t, \boldsymbol{w})$ is the forecasted $VaR_{j,t}$, across the j different VaR specifications we consider. We assess the partial derivative in time, since, following Kapetanios (2007), we expect it to vary in time, due to the inherent non-linearity of the neural network. Our covariate(s) $\boldsymbol{x}_t$ are the conditional volatility for GARCH and RM, VaR lagged values, the absolute $S\&P500$ daily return and the positive and negative S&P500 daily returns for SV and ASV, respectively. It is evident that under the classic linear regression problem, or linear quantile regression model, the effect of the covariates $\boldsymbol{x}_t$ to the dependent variable y_t is constant, time invariant, and corresponds to $\hat{\boldsymbol{\beta}}(\tau)$.

5.3 Results

In this application we use the whole sample size i.e. around 36 years of daily returns on the $S\&P500$ index to provide an accurate interpretation of our *deep quantile* estimator. Figures 3 - 6 illustrate the partial derivatives and SHAP values evaluated in time on the output of our *deep quantile*[6] estimator, for a specific quantile τ. Further, we compare the partial derivatives of the *deep quantile* estimator relative to the linear quantile regression partial derivative, i.e. the $\boldsymbol{\beta}(\tau)$ coefficient. Both partial derivatives and SHAP values seem to identify interesting patterns that can be linked to some well known events. Below we discuss our results for all models we have considered in our empirical application.

The results for the first two models, i.e. *GARCH* and *RM* can be summarised together, since in both models there is only one covariate, that is the conditional volatility, but with a different specification. The results for this model are illustrated in Figures 3. We find that the partial derivative appears to be more stable over time, fluctuating around the constant partial derivative, $\boldsymbol{\beta}(\tau)$, of the linear quantile estimator. When there is a crisis or a stressful event in the financial markets, they increase. As an example, we see significant spikes in

[6]In this section we limit our attention in the output of the best performing model, in terms of its forecasting capacity, as reflected by the forecast gains measure in Section 4, for each model, based on the different penalisation schemes. Results from all the different penalisation schemes suggest similar patterns to the ones discussed above and are available upon request.

the partial derivatives, both in March 2020 as well as in 2008, which stand for the onset of the CoVid 19 pandemic and the *Great Recession* respectively. We also find that the biggest increase occurs in 1987, the year when *Black Monday* happened, and also significant variation during the U.S government shutdown in 2019. The values for the partial derivative generally increase, as τ decreases. SHAP values have a similar behaviour with the partial derivatives, but are more volatile across time. For the first two models, there are some events, e.g. 1991, where the values both SHAP and partial derivatives do not increase a lot. We view this finding as an inability of these two models, to properly account for this crisis.

In the last two models, the merit of SHAP values and partial derivatives becomes clear, since in these models we have more than one covariates and both methods can provide an indication on the effect of each covariate on the final output. Overall, we find that increasing the number of covariates, allow the models to account for all crises within the sample. For the case of the SV model, we find that the important covariate is the lagged values of VaR, rather than the absolute values of $S\&P500$. Similar to the one covariate models, we find that the partial derivatives are more stable than SHAP values, fluctuating closely around $\boldsymbol{\beta}(\tau)$ and picking up when there are crisis or distress in the economy or financial markets. The SHAP values again appear to be more volatile with a wider range. Similar to the findings of the one covariate models, the higher the values for the partial derivative and SHAP, the lower the τ quantile.

For the case of the ASV model, which according to the results of Section 4 has the best forecasting performance, among all other models, we find that again the lagged values of VaR is the most significant covariate, the negative $S\&P500$ returns have some impact and the positive $S\&P500$ returns are almost insignificant. Similar to the cases above, we find that the partial derivative is more stable than SHAP values, fluctuating closely around $\boldsymbol{\beta}(\tau)$ and picking up when there is a crisis or distress in the economy or financial markets. The SHAP values again appear to be more volatile with a wider range. Again and same as before, lower quantiles have higher partial derivatives. The results for these two models are illustrated in Figures 4, 5, 6.

Different penalization schemes maintain the aforementioned results, with a lower magnitude. Overall, we observe that the linear quantile regression shows a fixed pattern across time and is evident that this model does not anticipate shocks in the economy. Further VaR measures how much a portfolio can lose within a given time period with some small probability, our analysis suggests that it is higher during stressful events. As Engle and Manganelli (2004) suggest, SV and ASV react more to negative shocks and in stressful events their spike is larger than the $GARCH$ and RM models. Finally, covariates with the minimum contribution on the forecasted values, such as the positive $S\&P500$ returns has almost zero impact on both SHAP and partial derivatives values.

Figure 3 about here

Figure 4 about here

Figure 5 about here

Figure 6 about here

6 Conclusion

In this paper we contribute to the expanding literature on the use of ML in finance and propose a *deep quantile* estimator that has the potential to capture the non-linear association between asset returns and predictors. In Section 2, we lay out the exact workings of our proposed estimator, and illustrate how it generalises linear quantile regression.

In the Monte Carlo exercise in Section 2, we study the finite sample properties of the *deep quantile* estimator, based on a number of data generating processes. We present extensive evidence that the estimator gives good small sample performance, that is a function of T, uniformly across different regularisation schemes.

We use the *deep quantile* estimator, with various penalization schemes, to forecast *VaR*. We find that our estimator gives considerable predictive gains, up to 98%, relative to the *VaR* forecasts produced by the linear quantile regression. This result is backed by the forecasting accuracy tests, i.e. the Diebold and Mariano (1995), the Giacomini and White (2006) and the quantile score tests. Further, we find, using the CQFE test of Giacomini and Komunjer (2005) that forecasts from the *deep quantile* estimator encompass the forecast from the linear model more times. These findings are in support of the non-linear association between the conditional quantile and covariates, hence suggesting a new avenue in forecasting in finance and in macroeconomics during extreme events.

In addition, we do a semi-structural analysis to examine the contribution of the predictors in *VaR* over time. We consider, following the ML literature, the SHAP values and further partial derivatives. Our findings suggests that our non-linear estimator reacts more in stressful events and exhibits a time-variation, while the linear quantile estimator presents a constant time invariant behaviour. We conclude that financial variables are characterised by non-linearities, that our proposed *deep quantile* estimator can approximate quite well. Finally, we make a formal comparison between SHAP and partial derivatives, and interestingly find that partial derivatives can be used to make ML methods interpretable, are less volatile, easier to interpret and can be computed at a fraction of time used in the calculation of SHAP values.

Appendix

Algorithm 1: MLP estimation

```
Result: $w_j$
load input, target variables and hyperparameters
initialize $w_j$, epoch=500 batch size;
for i in epoch do
    for j in batch size do
        forward propogation;
        compute loss function;
        backward propagation;
        update parameters;
    end
    calculate the output variable;
end
```

Further we apply another regularization scheme, the so-called early stopping. Early stopping stops our algorithm from a potential overfit.

Algorithm 2: Early Stopping

```
Result: $w_j$
initialize j, $\epsilon = \infty$, epoch=500 and patience =7;
for i in epoch do
    for j in batch size do
        forward propogation;
        compute loss function;
        backward propagation;
        update parameters;
    end
    calculate the forecast error;
    if forecast error $\leq \epsilon$ then
        j=0;
        forecast error = $\epsilon$;
        $w_j = w_j^{'}$;
    else
        j+=1;
    end
end
```

References

ADAMS, P. A., T. ADRIAN, N. BOYARCHENKO, AND D. GIANNONE (2021): "Forecasting macroeconomic risks," *International Journal of Forecasting*.

ATHEY, S., AND G. W. IMBENS (2017): "The state of applied econometrics: Causality and policy evaluation," *Journal of Economic Perspectives*, 31(2), 3–32.

BATES, J. M., AND C. W. GRANGER (1969): "The combination of forecasts," *Journal of the Operational Research Society*, 20(4), 451–468.

BAUR, D., AND N. SCHULZE (2005): "Coexceedances in financial markets—a quantile regression analysis of contagion," *Emerging Markets Review*, 6(1), 21–43.

BELLONI, A., V. CHERNOZHUKOV, AND C. HANSEN (2014): "Inference on treatment effects after selection among high-dimensional controls," *The Review of Economic Studies*, 81(2), 608–650.

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

BUCCI, A. (2020): "Realized Volatility Forecasting with Neural Networks," *Journal of Financial Econometrics*, 18(3), 502–531.

CARRIERO, A., T. E. CLARK, AND M. MARCELLINO (2015): "Bayesian VARs: specification choices and forecast accuracy," *Journal of Applied Econometrics*, 30(1), 46–73.

CARRIERO, A., G. KAPETANIOS, AND M. MARCELLINO (2009): "Forecasting exchange rates with a large Bayesian VAR," *International Journal of Forecasting*, 25(2), 400–417.

CENESIZOGLU, T., AND A. TIMMERMANN (2007): "Predictability of stock returns: a quantile regression approach," Discussion paper, Technical report, Technical Report, Cirano.

CHERNOZHUKOV, V., AND L. UMANTSEV (2001): "Conditional value-at-risk: Aspects of modeling and estimation," *Empirical Economics*, 26(1), 271–292.

DIEBOLD, F. X., AND R. S. MARIANO (1995): "Comparing predictive accuracy," *Journal of Business and Economic Statistics*, 13, 253–263, Reprinted in Mills, T. C. (ed.) (1999), *Economic Forecasting. The International Library of Critical Writings in Economics*. Cheltenham: Edward Elgar.

DU, Z., M. WANG, AND Z. XU (2019): "On Estimation of Value-at-Risk with Recurrent Neural Network," in *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, pp. 103–106. IEEE.

ENGLE, R. F., AND S. MANGANELLI (2004): "CAViaR: Conditional autoregressive value at risk by regression quantiles," *Journal of Business & Economic Statistics*, 22(4), 367–381.

FARRELL, M. H., T. LIANG, AND S. MISRA (2021): "Deep neural networks for estimation and inference," *Econometrica*, 89(1), 181–213.

GALANT, A., AND H. WHITE (1992): "On learning the derivatives of an unknown mapping with multilayer feed forward neural network," *Neural networks*, 5, 129–138.

GIACOMINI, R., AND I. KOMUNJER (2005): "Evaluation and combination of conditional quantile forecasts," *Journal of Business & Economic Statistics*, 23(4), 416–431.

GIACOMINI, R., AND H. WHITE (2006): "Tests of conditional predictive ability," *Econometrica*, 74(6), 1545–1578.

GU, S., B. KELLY, AND D. XIU (2020a): "Autoencoder asset pricing models," *Journal of Econometrics*.

——— (2020b): "Empirical asset pricing via machine learning," *The Review of Financial Studies*, 33(5), 2223–2273.

HANSEN, L. P. (1982): "Large sample properties of generalized method of moments estimators," *Econometrica: Journal of the Econometric Society*, pp. 1029–1054.

HARVEY, D., S. LEYBOURNE, AND P. NEWBOLD (1997): "Testing the equality of prediction mean squared errors," *International Journal of forecasting*, 13(2), 281–291.

HE, Z., AND A. KRISHNAMURTHY (2013): "Intermediary asset pricing," *American Economic Review*, 103(2), 732–70.

HORNIK, K. (1991): "Approximation capabilities of multilayer feedforward networks," *Neural networks*, 4(2), 251–257.

HORNIK, K., M. STINCHCOMBE, H. WHITE, ET AL. (1989): "Multilayer feedforward networks are universal approximators.," *Neural networks*, 2(5), 359–366.

JOSEPH, A. (2019): "Shapley regressions: a framework for statistical inference on machine learning models," Discussion paper, Bank of England.

J.P. MORGAN, M. (1996): "Reuters (1996) RiskMetrics-Technical Document," *JP Morgan*.

KAPETANIOS, G. (2007): "Measuring conditional persistence in nonlinear time series," *Oxford Bulletin of Economics and Statistics*, 69(3), 363–386.

KAPETANIOS, G., AND A. P. BLAKE (2010): "Tests of the martingale difference hypothesis using boosting and RBF neural network approximations," *Econometric Theory*, 26(5), 1363–1397.

KINGMA, D. P., AND J. BA (2014): "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*.

KOENKER, R. (2005): *Quantile Regression*, Econometric Society Monographs. Cambridge University Press.

KOENKER, R., AND G. BASSETT JR (1978): "Regression quantiles," *Econometrica: journal of the Econometric Society*, pp. 33–50.

KOENKER, R., V. CHERNOZHUKOV, X. HE, AND L. PENG (2017): *Handbook of quantile regression*. CRC press.

KOENKER, R., AND K. F. HALLOCK (2001): "Quantile regression," *Journal of economic perspectives*, 15(4), 143–156.

LIANG, S., AND R. SRIKANT (2016): "Why deep neural networks for function approximation?," *arXiv preprint arXiv:1610.04161*.

LUNDBERG, S. M., G. G. ERION, AND S.-I. LEE (2018): "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*.

LUNDBERG, S. M., AND S.-I. LEE (2017): "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, pp. 4765–4774.

MEINSHAUSEN, N. (2006): "Quantile regression forests," *Journal of Machine Learning Research*, 7(Jun), 983–999.

MHASKAR, H., Q. LIAO, AND T. POGGIO (2017): "When and why are deep networks better than shallow ones?," in *Thirty-First AAAI Conference on Artificial Intelligence*.

NEWEY, W. K., AND K. D. WEST (1987): "Hypothesis testing with efficient method of moments estimation," *International Economic Review*, pp. 777–787.

PARK, J., AND I. W. SANDBERG (1991): "Universal Approximation using Radial-Basis-Function Networks," *Neural Computation*, 3(4), 246–257.

POHL, W., K. SCHMEDDERS, AND O. WILMS (2018): "Higher order effects in asset pricing models with long-run risks," *The Journal of Finance*, 73(3), 1061–1111.

SHAPLEY, L. S. (1953): "A value for n-person games," *Contributions to the Theory of Games*, 2(28), 307–317.

SHRIKUMAR, A., P. GREENSIDE, AND A. KUNDAJE (2017): "Learning important features through propagating activation differences," *arXiv preprint arXiv:1704.02685*.

SMALTER HALL, A., AND T. R. COOK (2017): "Macroeconomic indicator forecasting with deep neural networks," *Federal Reserve Bank of Kansas City Working Paper*, (17-11).

WAGER, S., AND S. ATHEY (2018): "Estimation and inference of heterogeneous treatment effects using random forests," *Journal of the American Statistical Association*, 113(523), 1228–1242.

WANG, Q., ET AL. (2018): "Exponential convergence of the deep neural network approximation for analytic functions," *arXiv preprint arXiv:1807.00297*.

ZHANG, W., H. QUAN, AND D. SRINIVASAN (2018): "An improved quantile regression neural network for probabilistic load forecasting," *IEEE Transactions on Smart Grid*, 10(4), 4425–4434.

ZOU, H., AND T. HASTIE (2005): "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, 67(2), 301–320.

Table 1: Monte Carlo results for Case I. Model: $y_t = h_\tau(x_t) + u_t,\ h_\tau(x_t) = \sin(2\pi x_t),\ x_t \sim\ N(0,1),\ u_t \sim iidN(-\sigma\Phi^{-1}(\tau), \sigma^2),\ \sigma = 0.1$ and Φ^{-1} is the quantile function of the standard normal distribution.

T	Model	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$
			$\tau = 0.01$			$\tau = 0.025$			$\tau = 0.05$			$\tau = 0.1$			$\tau = 0.2$	
50	Deep QR	0.06	1.30	0.67	0.05	1.19	0.64	0.04	1.07	0.60	0.03	0.86	0.48	0.02	0.66	0.36
	Lasso QR	0.06	1.18	0.60	0.05	1.13	0.60	0.04	0.95	0.52	0.03	0.72	0.41	0.02	0.55	0.29
	Ridge QR	0.06	1.30	0.77	0.05	1.18	0.74	0.04	0.97	0.68	0.02	0.73	0.58	0.02	0.48	0.48
	Elastic Net QR	0.06	1.13	0.72	0.05	1.03	0.70	0.04	0.86	0.64	0.02	0.61	0.53	0.02	0.34	0.39
	Linear QR	0.06	1.63	0.83	0.05	1.65	0.86	0.04	1.53	0.85	0.03	1.39	0.80	0.02	1.15	0.69
100	Deep QR	0.07	1.44	0.75	0.05	1.26	0.70	0.04	1.02	0.58	0.03	0.64	0.40	0.02	0.43	0.23
	Lasso QR	0.07	1.30	0.66	0.05	1.19	0.64	0.04	0.99	0.57	0.03	0.62	0.39	0.02	0.34	0.19
	Ridge QR	0.07	1.48	0.83	0.05	1.23	0.75	0.04	0.97	0.66	0.03	0.68	0.54	0.02	0.36	0.36
	Elastic Net QR	0.07	1.32	0.79	0.05	1.12	0.73	0.04	0.85	0.62	0.03	0.55	0.45	0.02	0.28	0.30
	Linear QR	0.07	1.75	0.89	0.05	1.64	0.87	0.04	1.53	0.85	0.03	1.37	0.81	0.02	1.13	0.70
300	Deep QR	0.07	1.42	0.74	0.05	0.88	0.51	0.04	0.53	0.32	0.03	0.30	0.18	0.02	0.17	0.09
	Lasso QR	0.07	1.38	0.70	0.05	1.17	0.64	0.04	0.80	0.49	0.03	0.35	0.23	0.02	0.16	0.09
	Ridge QR	0.06	1.48	0.81	0.05	0.96	0.60	0.04	0.52	0.37	0.03	0.28	0.24	0.02	0.13	0.15
	Elastic Net QR	0.06	1.27	0.76	0.05	0.94	0.60	0.04	0.50	0.37	0.03	0.25	0.22	0.02	0.12	0.15
	Linear QR	0.07	1.82	0.91	0.05	1.68	0.89	0.04	1.58	0.87	0.03	1.43	0.83	0.02	1.17	0.73
500	Deep QR	0.06	0.98	0.53	0.05	0.57	0.32	0.04	0.40	0.23	0.03	0.25	0.15	0.02	0.15	0.09
	Lasso QR	0.06	1.30	0.65	0.05	1.10	0.61	0.04	0.54	0.33	0.03	0.30	0.19	0.02	0.16	0.09
	Ridge QR	0.06	1.43	0.80	0.05	0.63	0.41	0.04	0.37	0.26	0.03	0.23	0.20	0.02	0.11	0.14
	Elastic Net QR	0.06	1.26	0.76	0.05	0.67	0.44	0.04	0.34	0.25	0.03	0.22	0.19	0.02	0.10	0.14
	Linear QR	0.06	1.73	0.87	0.05	1.62	0.85	0.04	1.52	0.83	0.03	1.37	0.79	0.02	1.11	0.69
1000	Deep QR	0.07	0.58	0.31	0.05	0.42	0.23	0.04	0.29	0.17	0.03	0.18	0.11	0.02	0.11	0.07
	Lasso QR	0.07	1.35	0.68	0.05	0.63	0.37	0.04	0.35	0.20	0.03	0.21	0.14	0.02	0.11	0.07
	Ridge QR	0.06	1.10	0.62	0.05	0.46	0.27	0.04	0.35	0.24	0.03	0.20	0.17	0.02	0.10	0.11
	Elastic Net QR	0.06	1.21	0.72	0.05	0.43	0.28	0.04	0.29	0.21	0.03	0.18	0.16	0.02	0.09	0.12
	Linear QR	0.07	1.77	0.89	0.05	1.65	0.87	0.04	1.54	0.85	0.03	1.39	0.81	0.02	1.14	0.71
2000	Deep QR	0.06	0.48	0.25	0.05	0.36	0.18	0.04	0.30	0.16	0.03	0.15	0.08	0.02	0.09	0.05
	Lasso QR	0.06	1.10	0.58	0.05	0.43	0.24	0.04	0.28	0.17	0.03	0.16	0.10	0.02	0.10	0.05
	Ridge QR	0.06	0.58	0.33	0.05	0.32	0.20	0.04	0.26	0.18	0.03	0.15	0.13	0.02	0.08	0.09
	Elastic Net QR	0.06	0.71	0.42	0.05	0.33	0.22	0.04	0.23	0.17	0.03	0.13	0.12	0.02	0.07	0.09
	Linear QR	0.06	1.77	0.89	0.05	1.65	0.87	0.04	1.54	0.85	0.03	1.39	0.81	0.02	1.14	0.71
5000	Deep QR	0.07	0.43	0.22	0.05	0.26	0.13	0.04	0.17	0.09	0.03	0.12	0.07	0.02	0.06	0.04
	Lasso QR	0.07	0.60	0.31	0.05	0.32	0.17	0.04	0.18	0.11	0.03	0.11	0.06	0.02	0.07	0.03
	Ridge QR	0.06	0.39	0.22	0.05	0.22	0.14	0.04	0.14	0.11	0.03	0.09	0.08	0.02	0.06	0.07
	Elastic Net QR	0.06	0.45	0.26	0.05	0.25	0.16	0.04	0.13	0.10	0.03	0.08	0.08	0.02	0.05	0.06
	Linear QR	0.07	1.76	0.89	0.05	1.64	0.87	0.04	1.53	0.85	0.03	1.38	0.81	0.02	1.13	0.71

Notes: Table presents the average mean squared error of the true residuals, $AMSE_{u_t}$, the average mean squared error of the estimated residuals, $AMSE_{\hat{u}_t,pen}$ and the average absolute bias $ABIAS_{\hat{h}_\tau,pen}$, for the different penalization schemes (Model), $T = 100, 300, 500, 1000, 2000, 5000$ and different quantiles, $\tau = 0.01, 0.025, 0.05, 0.1, 0.2$.

Table 2: Monte Carlo results for Case II. Model: $y_t = h_\tau(x_t) + u_t,\ h_\tau(x_t) = \sin(2\pi x_t),\ x_t = 0.8x_{t-1} + \varepsilon_t,\ \varepsilon_t \sim N(0,1),\ u_t \sim iidN(-\sigma\Phi^{-1}(\tau), \sigma^2),\ \sigma = 0.1$ and Φ^{-1} is the quantile function of the standard normal distribution.

T	Model	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$
			$\tau = 0.01$			$\tau = 0.025$			$\tau = 0.05$			$\tau = 0.1$			$\tau = 0.2$	
50	Deep QR	0.06	1.42	0.83	0.05	1.35	0.83	0.04	1.27	0.81	0.03	1.11	0.78	0.02	0.89	0.72
	Lasso QR	0.06	1.28	0.78	0.05	1.22	0.78	0.04	1.11	0.76	0.03	0.95	0.72	0.02	0.65	0.63
	Ridge QR	0.06	1.39	0.82	0.05	1.24	0.79	0.04	1.15	0.77	0.03	1.02	0.75	0.02	0.74	0.65
	Elastic Net QR	0.06	1.25	0.77	0.05	1.16	0.76	0.04	1.08	0.74	0.03	0.91	0.71	0.02	0.61	0.60
	Linear QR	0.06	1.73	0.87	0.05	1.71	0.89	0.04	1.62	0.87	0.03	1.43	0.83	0.02	1.18	0.70
100	Deep QR	0.06	1.51	0.84	0.05	1.45	0.84	0.04	1.30	0.80	0.03	1.09	0.74	0.02	0.78	0.64
	Lasso QR	0.06	1.34	0.80	0.05	1.29	0.80	0.04	1.23	0.79	0.03	1.00	0.72	0.02	0.68	0.63
	Ridge QR	0.06	1.47	0.83	0.05	1.37	0.82	0.04	1.23	0.78	0.03	1.02	0.72	0.02	0.73	0.63
	Elastic Net QR	0.06	1.25	0.78	0.05	1.23	0.78	0.04	1.11	0.75	0.03	0.91	0.68	0.02	0.63	0.58
	Linear QR	0.06	1.81	0.89	0.05	1.73	0.90	0.04	1.63	0.87	0.03	1.48	0.83	0.02	1.18	0.71
300	Deep QR	0.07	1.61	0.86	0.05	1.41	0.82	0.04	1.15	0.72	0.03	0.81	0.57	0.02	0.58	0.48
	Lasso QR	0.07	1.35	0.79	0.05	1.30	0.80	0.04	1.22	0.79	0.03	0.93	0.68	0.02	0.54	0.49
	Ridge QR	0.07	1.55	0.85	0.05	1.38	0.82	0.04	1.17	0.75	0.03	0.83	0.60	0.02	0.48	0.43
	Elastic Net QR	0.07	1.31	0.79	0.05	1.25	0.79	0.04	1.14	0.75	0.03	0.77	0.58	0.02	0.42	0.40
	Linear QR	0.07	1.76	0.89	0.05	1.64	0.87	0.04	1.54	0.85	0.03	1.39	0.82	0.02	1.14	0.71
500	Deep QR	0.07	1.61	0.86	0.05	1.31	0.75	0.04	1.04	0.65	0.03	0.68	0.48	0.02	0.45	0.37
	Lasso QR	0.07	1.38	0.80	0.05	1.32	0.80	0.04	1.24	0.79	0.03	0.85	0.61	0.02	0.44	0.40
	Ridge QR	0.07	1.60	0.86	0.05	1.38	0.80	0.04	1.12	0.70	0.03	0.71	0.51	0.02	0.41	0.36
	Elastic Net QR	0.07	1.35	0.79	0.05	1.29	0.79	0.04	1.05	0.69	0.03	0.66	0.49	0.02	0.35	0.33
	Linear QR	0.07	1.80	0.90	0.05	1.68	0.88	0.04	1.56	0.86	0.03	1.43	0.83	0.02	1.17	0.72
1000	Deep QR	0.06	1.41	0.76	0.05	1.07	0.63	0.04	0.75	0.47	0.03	0.55	0.38	0.02	0.38	0.31
	Lasso QR	0.06	1.36	0.80	0.05	1.32	0.80	0.04	1.21	0.77	0.03	0.68	0.47	0.02	0.40	0.34
	Ridge QR	0.06	1.55	0.85	0.05	1.19	0.70	0.04	0.85	0.54	0.03	0.54	0.39	0.02	0.37	0.31
	Elastic Net QR	0.06	1.36	0.80	0.05	1.17	0.72	0.04	0.86	0.56	0.03	0.52	0.38	0.02	0.32	0.29
	Linear QR	0.06	1.78	0.89	0.05	1.66	0.88	0.04	1.56	0.86	0.03	1.40	0.82	0.02	1.15	0.72
2000	Deep QR	0.06	1.16	0.63	0.05	0.75	0.45	0.04	0.62	0.39	0.03	0.45	0.31	0.02	0.30	0.25
	Lasso QR	0.06	1.35	0.79	0.05	1.25	0.76	0.04	0.82	0.52	0.03	0.51	0.36	0.02	0.33	0.29
	Ridge QR	0.06	1.45	0.80	0.05	0.95	0.56	0.04	0.61	0.39	0.03	0.45	0.32	0.02	0.29	0.25
	Elastic Net QR	0.06	1.33	0.78	0.05	0.96	0.59	0.04	0.61	0.40	0.03	0.42	0.31	0.02	0.25	0.24
	Linear QR	0.06	1.77	0.89	0.05	1.65	0.87	0.04	1.53	0.85	0.03	1.39	0.81	0.02	1.14	0.71
5000	Deep QR	0.06	0.88	0.48	0.05	0.59	0.35	0.04	0.47	0.30	0.03	0.34	0.25	0.02	0.24	0.21
	Lasso QR	0.06	1.32	0.78	0.05	0.90	0.52	0.04	0.66	0.41	0.03	0.43	0.30	0.02	0.27	0.24
	Ridge QR	0.06	1.22	0.68	0.05	0.62	0.37	0.04	0.48	0.31	0.03	0.32	0.23	0.02	0.22	0.20
	Elastic Net QR	0.06	1.14	0.67	0.05	0.72	0.43	0.04	0.49	0.32	0.03	0.30	0.23	0.02	0.20	0.19
	Linear QR	0.06	1.76	0.89	0.05	1.65	0.87	0.04	1.54	0.85	0.03	1.39	0.81	0.02	1.14	0.71

Notes: Table presents the average mean squared error of the true residuals, $AMSE_{u_t}$, the average mean squared error of the estimated residuals, $AMSE_{\hat{u}_t,pen}$ and the average absolute bias $ABIAS_{\hat{h}_\tau,pen}$, for the different penalization schemes (Model), $T = 100, 300, 500, 1000, 2000, 5000$ and different quantiles, $\tau = 0.01, 0.025, 0.05, 0.1, 0.2$.

Table 3: Monte Carlo results for Case III. Model: $y_t = h_\tau(x_t) + u_t$, $h_\tau(x_t) = \sin(2\pi x_t)$, $x_t = \sigma_t \varepsilon_t$, $\sigma_t^2 = 1 + 0.7x_{t-1}^2 + 0.2\sigma_{t-1}^2$, $\varepsilon_t \sim N(0,1)$, $u_t \sim iidN(-\sigma\Phi^{-1}(\tau), \sigma^2)$, $\sigma = 0.1$ and Φ^{-1} is the quantile function of the standard normal distribution.

T	Model	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$
			$\tau = 0.01$			$\tau = 0.025$			$\tau = 0.05$			$\tau = 0.1$			$\tau = 0.2$	
50	Deep QR	0.06	1.57	0.87	0.05	1.46	0.86	0.04	1.37	0.84	0.03	1.20	0.80	0.02	0.91	0.73
	Lasso QR	0.06	1.40	0.83	0.05	1.33	0.81	0.04	1.24	0.80	0.03	1.05	0.76	0.02	0.76	0.67
	Ridge QR	0.06	1.46	0.84	0.05	1.34	0.82	0.04	1.27	0.81	0.03	1.07	0.76	0.02	0.81	0.69
	Elastic Net QR	0.06	1.36	0.81	0.05	1.24	0.79	0.04	1.18	0.78	0.03	1.00	0.73	0.02	0.70	0.64
	Linear QR	0.06	1.75	0.87	0.05	1.74	0.90	0.04	1.61	0.88	0.03	1.46	0.84	0.02	1.20	0.72
100	Deep QR	0.06	1.48	0.83	0.05	1.43	0.83	0.04	1.33	0.82	0.03	1.17	0.79	0.02	0.88	0.71
	Lasso QR	0.06	1.38	0.81	0.05	1.32	0.80	0.04	1.19	0.78	0.03	1.05	0.75	0.02	0.76	0.68
	Ridge QR	0.06	1.42	0.81	0.05	1.35	0.81	0.04	1.26	0.80	0.03	1.09	0.77	0.02	0.80	0.68
	Elastic Net QR	0.06	1.31	0.78	0.05	1.23	0.78	0.04	1.14	0.77	0.03	0.99	0.74	0.02	0.69	0.64
	Linear QR	0.06	1.70	0.86	0.05	1.62	0.86	0.04	1.50	0.84	0.03	1.34	0.79	0.02	1.10	0.69
300	Deep QR	0.06	1.61	0.86	0.05	1.52	0.85	0.04	1.39	0.83	0.03	1.20	0.78	0.02	0.84	0.65
	Lasso QR	0.06	1.48	0.83	0.05	1.38	0.82	0.04	1.29	0.81	0.03	1.13	0.78	0.02	0.79	0.69
	Ridge QR	0.06	1.50	0.83	0.05	1.39	0.82	0.04	1.29	0.81	0.03	1.12	0.76	0.02	0.78	0.65
	Elastic Net QR	0.06	1.40	0.80	0.05	1.32	0.80	0.04	1.23	0.79	0.03	1.05	0.75	0.02	0.72	0.63
	Linear QR	0.06	1.73	0.87	0.05	1.61	0.85	0.04	1.51	0.84	0.03	1.36	0.79	0.02	1.11	0.69
500	Deep QR	0.06	1.70	0.89	0.05	1.58	0.87	0.04	1.36	0.81	0.03	1.16	0.75	0.02	0.83	0.63
	Lasso QR	0.06	1.60	0.87	0.05	1.49	0.85	0.04	1.35	0.83	0.03	1.17	0.79	0.02	0.79	0.66
	Ridge QR	0.06	1.57	0.86	0.05	1.49	0.85	0.04	1.35	0.82	0.03	1.16	0.77	0.02	0.78	0.63
	Elastic Net QR	0.06	1.47	0.83	0.05	1.36	0.82	0.04	1.26	0.80	0.03	1.07	0.76	0.02	0.71	0.60
	Linear QR	0.06	1.80	0.90	0.05	1.68	0.88	0.04	1.56	0.86	0.03	1.42	0.82	0.02	1.16	0.72
1000	Deep QR	0.07	1.69	0.89	0.05	1.49	0.84	0.04	1.27	0.76	0.03	0.97	0.64	0.02	0.70	0.53
	Lasso QR	0.07	1.58	0.86	0.05	1.45	0.84	0.04	1.35	0.83	0.03	1.14	0.78	0.02	0.73	0.62
	Ridge QR	0.07	1.58	0.86	0.05	1.46	0.84	0.04	1.31	0.80	0.03	1.03	0.69	0.02	0.65	0.52
	Elastic Net QR	0.07	1.48	0.83	0.05	1.38	0.82	0.04	1.23	0.79	0.03	0.98	0.68	0.02	0.60	0.51
	Linear QR	0.07	1.77	0.89	0.05	1.65	0.87	0.04	1.54	0.85	0.03	1.39	0.81	0.02	1.14	0.71
2000	Deep QR	0.06	1.63	0.86	0.05	1.37	0.77	0.04	1.07	0.64	0.03	0.81	0.53	0.02	0.56	0.43
	Lasso QR	0.06	1.56	0.85	0.05	1.44	0.84	0.04	1.31	0.81	0.03	1.05	0.70	0.02	0.65	0.52
	Ridge QR	0.06	1.55	0.85	0.05	1.43	0.83	0.04	1.25	0.76	0.03	0.86	0.58	0.02	0.54	0.43
	Elastic Net QR	0.06	1.45	0.82	0.05	1.35	0.81	0.04	1.20	0.76	0.03	0.81	0.57	0.02	0.52	0.42
	Linear QR	0.06	1.74	0.88	0.05	1.63	0.86	0.04	1.52	0.84	0.03	1.37	0.80	0.02	1.13	0.70
5000	Deep QR	0.06	1.41	0.75	0.05	1.13	0.64	0.04	0.84	0.51	0.03	0.62	0.42	0.02	0.42	0.34
	Lasso QR	0.06	1.50	0.84	0.05	1.42	0.82	0.04	1.27	0.77	0.03	0.87	0.58	0.02	0.52	0.40
	Ridge QR	0.06	1.53	0.84	0.05	1.36	0.79	0.04	1.07	0.65	0.03	0.65	0.44	0.02	0.44	0.35
	Elastic Net QR	0.06	1.40	0.81	0.05	1.29	0.77	0.04	0.99	0.62	0.03	0.64	0.44	0.02	0.41	0.34
	Linear QR	0.06	1.77	0.89	0.05	1.65	0.87	0.04	1.54	0.85	0.03	1.39	0.81	0.02	1.14	0.71

Notes: Table presents the average mean squared error of the true residuals, $AMSE_{u_t}$, the average mean squared error of the estimated residuals, $AMSE_{\hat{u}_t,pen}$ and the average absolute bias $ABIAS_{\hat{h}_\tau,pen}$, for the different penalization schemes (Model), $T = 100, 300, 500, 1000, 2000, 5000$ and different quantiles, $\tau = 0.01, 0.025, 0.05, 0.1, 0.2$.

Table 4: Monte Carlo results for Case IV. Model: $y_t = h_\tau(x_t) + u_t$, $h_\tau(x_t) = G(x_t, \boldsymbol{w})$, $x_t \sim N(0,1)$, $w_{i,j} = \delta_{i,j} 1(\delta_{i,j} > 0.5)$, $\delta_{i,j} \sim U(0,1)$, $u_t \sim iidN(-\sigma\Phi^{-1}(\tau), \sigma^2)$, $\sigma = 0.1$ and Φ^{-1} is the quantile function of the standard normal distribution.

T	Model	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$	$AMSE_{u_t}$	$AMSE_{\hat{u}_t,pen}$	$ABIAS_{\hat{h}_\tau,pen}$
			$\tau = 0.01$			$\tau = 0.025$			$\tau = 0.05$			$\tau = 0.1$			$\tau = 0.2$	
50	Deep QR	0.06	0.26	0.22	0.05	0.23	0.21	0.04	0.28	0.23	0.03	0.10	0.11	0.02	0.10	0.13
	Lasso QR	0.06	0.27	0.23	0.05	0.22	0.21	0.04	0.22	0.19	0.03	0.09	0.10	0.02	0.07	0.10
	Ridge QR	0.06	0.27	0.24	0.05	0.23	0.21	0.04	0.24	0.20	0.03	0.08	0.10	0.02	0.06	0.09
	Elastic Net QR	0.06	0.23	0.23	0.05	0.19	0.19	0.04	0.20	0.19	0.03	0.07	0.10	0.02	0.05	0.09
	Linear QR	0.06	0.49	0.22	0.05	0.66	0.33	0.04	0.76	0.40	0.03	0.40	0.23	0.02	0.40	0.24
100	Deep QR	0.06	0.36	0.24	0.05	0.30	0.21	0.04	0.20	0.16	0.03	0.10	0.10	0.02	0.09	0.10
	Lasso QR	0.06	0.34	0.23	0.05	0.29	0.21	0.04	0.23	0.18	0.03	0.11	0.10	0.02	0.06	0.08
	Ridge QR	0.06	0.37	0.25	0.05	0.28	0.20	0.04	0.19	0.16	0.03	0.10	0.10	0.02	0.06	0.08
	Elastic Net QR	0.06	0.30	0.23	0.05	0.25	0.19	0.04	0.17	0.15	0.03	0.08	0.09	0.02	0.04	0.06
	Linear QR	0.06	0.66	0.32	0.05	0.67	0.35	0.04	0.66	0.36	0.03	0.65	0.37	0.02	0.43	0.25
300	Deep QR	0.06	0.30	0.20	0.05	0.16	0.12	0.04	0.11	0.09	0.03	0.06	0.05	0.02	0.05	0.05
	Lasso QR	0.06	0.35	0.22	0.05	0.20	0.14	0.04	0.15	0.11	0.03	0.07	0.06	0.02	0.03	0.04
	Ridge QR	0.06	0.33	0.20	0.05	0.16	0.12	0.04	0.13	0.10	0.03	0.06	0.06	0.02	0.03	0.03
	Elastic Net QR	0.06	0.30	0.20	0.05	0.15	0.12	0.04	0.11	0.10	0.03	0.05	0.06	0.02	0.02	0.04
	Linear QR	0.06	0.81	0.40	0.05	0.81	0.42	0.04	0.73	0.40	0.03	0.56	0.33	0.02	0.33	0.20
500	Deep QR	0.06	0.24	0.14	0.05	0.11	0.08	0.04	0.08	0.06	0.03	0.07	0.06	0.02	0.03	0.04
	Lasso QR	0.06	0.32	0.19	0.05	0.15	0.10	0.04	0.10	0.07	0.03	0.08	0.07	0.02	0.04	0.05
	Ridge QR	0.06	0.27	0.16	0.05	0.10	0.07	0.04	0.08	0.07	0.03	0.07	0.06	0.02	0.03	0.04
	Elastic Net QR	0.06	0.28	0.17	0.05	0.09	0.07	0.04	0.08	0.07	0.03	0.06	0.06	0.02	0.03	0.04
	Linear QR	0.06	0.79	0.39	0.05	0.74	0.38	0.04	0.60	0.33	0.03	0.50	0.29	0.02	0.46	0.29
1000	Deep QR	0.06	0.15	0.08	0.05	0.10	0.06	0.04	0.07	0.05	0.03	0.04	0.04	0.02	0.04	0.05
	Lasso QR	0.06	0.25	0.15	0.05	0.14	0.09	0.04	0.07	0.05	0.03	0.05	0.05	0.02	0.04	0.04
	Ridge QR	0.06	0.20	0.11	0.05	0.09	0.06	0.04	0.07	0.05	0.03	0.04	0.04	0.02	0.04	0.04
	Elastic Net QR	0.06	0.20	0.12	0.05	0.10	0.07	0.04	0.06	0.05	0.03	0.04	0.05	0.02	0.03	0.04
	Linear QR	0.06	0.77	0.38	0.05	0.75	0.38	0.04	0.61	0.34	0.03	0.50	0.29	0.02	0.53	0.33
2000	Deep QR	0.06	0.12	0.05	0.05	0.10	0.06	0.04	0.06	0.04	0.03	0.04	0.03	0.02	0.03	0.03
	Lasso QR	0.06	0.16	0.08	0.05	0.12	0.07	0.04	0.07	0.05	0.03	0.05	0.05	0.02	0.03	0.04
	Ridge QR	0.06	0.12	0.06	0.05	0.10	0.06	0.04	0.06	0.04	0.03	0.04	0.04	0.02	0.03	0.03
	Elastic Net QR	0.06	0.13	0.07	0.05	0.09	0.06	0.04	0.05	0.05	0.03	0.03	0.04	0.02	0.02	0.04
	Linear QR	0.06	0.66	0.33	0.05	0.78	0.41	0.04	0.64	0.35	0.03	0.51	0.29	0.02	0.48	0.29
5000	Deep QR	0.06	0.12	0.06	0.05	0.08	0.04	0.04	0.06	0.04	0.03	0.04	0.03	0.02	0.03	0.03
	Lasso QR	0.06	0.14	0.07	0.05	0.10	0.05	0.04	0.07	0.05	0.03	0.04	0.04	0.02	0.03	0.03
	Ridge QR	0.06	0.10	0.05	0.05	0.08	0.04	0.04	0.06	0.04	0.03	0.03	0.03	0.02	0.02	0.03
	Elastic Net QR	0.06	0.12	0.07	0.05	0.08	0.05	0.04	0.05	0.04	0.03	0.03	0.04	0.02	0.02	0.03
	Linear QR	0.06	0.71	0.35	0.05	0.69	0.36	0.04	0.57	0.31	0.03	0.60	0.35	0.02	0.52	0.32

Notes: Table presents the average mean squared error of the true residuals, $AMSE_{u_t}$, the average mean squared error of the estimated residuals, $AMSE_{\hat{u}_t,pen}$ and the average absolute bias $ABIAS_{\hat{h}_\tau,pen}$, for the different penalization schemes (Model), $T = 100, 300, 500, 1000, 2000, 5000$ and different quantiles, $\tau = 0.01, 0.025, 0.05, 0.1, 0.2$.

Table 5: CQFE test results in compact form.

$\eta = 0.005$	Wins	Losses	Inconclusive	Wins	Losses	Inconclusive	Wins	Losses	Inconclusive
		$\tau = 0.1$			$\tau = 0.05$			$\tau = 0.01$	
Linear GARCH	6	10	6	3	13	3	3	13	2
Linear RM	5	11	3	6	10	2	6	10	3
Linear SV	1	15	1	0	16	0	9	7	9
Linear ASV	1	15	1	3	13	0	9	7	9
deep GARCH	4	0	11	0	4	9	2	2	8
deep Lasso GARCH	3	1	13	1	3	8	1	3	11
deep Ridge GARCH	4	0	12	2	2	2	3	1	14
deep ELNet GARCH	1	3	13	3	1	15	1	3	8
deep RM	1	3	8	0	4	5	2	2	7
deep Lasso RM	1	3	11	1	3	7	3	1	8
deep Ridge RM	1	3	10	2	2	8	2	2	9
deep ELNet RM	1	3	9	1	3	10	2	2	5
deep SV	1	3	8	0	4	5	2	2	7
deep Lasso SV	1	3	4	1	3	4	3	1	10
deep Ridge SV	1	3	4	1	3	7	3	1	10
deep ELNet SV	1	3	2	1	3	3	2	2	8
deep ASV	1	3	2	3	1	7	3	1	11
deep Lasso ASV	1	3	4	3	1	3	2	2	11
deep Ridge ASV	0	4	2	3	1	7	3	1	9
deep ELNet ASV	1	3	3	3	1	3	3	1	11

Notes: The first four rows present the comparison of linear model j with all non-linear, while the rest depict the comparison of the non-linear model j with all linear ones. In columns we report the number of times that a model encompasses the competing one as Wins, the number of times it does not as Losses, the number of times the test is Inconclusive. Results are reported for this smoothing parameter $\eta = 0.005$.

Table 6: Results of the Diebold and Mariano (1995) test. Relative RMSFE of the non-linear model to the linear model.

τ	Model	QR NN	QR NN Lasso	QR NN Ridge	QR Elastic Net
0.1	GARCH	0.04***	0.09***	0.08***	0.21***
	RM	0.08***	0.15***	0.03***	0.07***
	SV	0.10***	0.19***	0.07***	0.17***
	ASV	0.02***	0.08***	0.02***	0.02***
0.05	GARCH	0.12***	0.21***	0.12***	0.27***
	RM	0.06***	0.08***	0.02***	0.03***
	SV	0.05***	0.08***	0.39***	0.08***
	ASV	0.02***	0.08***	0.02***	0.01***
0.01	GARCH	0.22***	0.29***	0.14***	0.27***
	RM	0.04***	0.11***	0.06***	0.15***
	SV	0.04***	0.09***	0.05***	0.10***
	ASV	0.03***	0.22***	0.11***	0.02***

Notes: *, **, and *** denote results from Diebold and Mariano (1995) test with the Harvey, Leybourne, and Newbold (1997) adjustment for predictive accuracy, indicating rejection of the null hypothesis that the models have the same predictive accuracy at the 10%, 5%, and 1% levels of significance, respectively. Values less than 1 indicate that there are forecast gains associated with use of our *deep quantile* estimator.

Table 7: Results of the Giacomini and White (2006) test. Relative RMSFE of the non-linear model to the linear model.

τ	Model	QR NN	QR NN Lasso	QR NN Ridge	QR Elastic Net
0.1	GARCH	0.04***	0.09***	0.08***	0.21***
	RM	0.08***	0.15***	0.03***	0.07***
	SV	0.10***	0.19***	0.07***	0.17***
	ASV	0.02***	0.08***	0.02**	0.02***
0.05	GARCH	0.12***	0.21***	0.12***	0.27***
	RM	0.06***	0.08***	0.02***	0.03***
	SV	0.05	0.08	0.39***	0.08
	ASV	0.02***	0.08***	0.02***	0.01***
0.01	GARCH	0.22***	0.29***	0.14***	0.27***
	RM	0.04***	0.11***	0.06***	0.15***
	SV	0.04**	0.09	0.05*	0.10
	ASV	0.03**	0.22***	0.11***	0.02**

Notes: *, **, and *** denote results from the Giacomini and White (2006) test, indicating rejection of the null hypothesis of equal forecast accuracy of the models at the 10%, 5%, and 1% levels of significance, respectively. Values less than 1 indicate that there are forecast gains associated with use of our *deep quantile* estimator.

Table 8: Average Quantile Score of the non-linear model (in columns) relative to the linear model (in rows).

τ	Model	QR NN	QR NN Lasso	QR NN Ridge	QR Elastic Net
0.1	GARCH	0.0033	0.0048	0.0048	0.0057
	RM	0.0071	0.0077	0.0054	0.0069
	SV	0.0209	0.0219	0.0209	0.0217
	ASV	0.0295	0.0302	0.0295	0.0296
0.05	GARCH	0.0008	0.0019	0.0004	0.0023
	RM	-0.0004	0.0010	-0.0010	-0.0001
	SV	0.0360	0.0360	0.0380	0.0360
	ASV	0.0525	0.0531	0.0525	0.0524
0.01	GARCH	-0.0003	0.0000	-0.0006	-0.0001
	RM	-0.0022	0.0002	-0.0006	-0.0001
	SV	0.0796	0.0796	0.0796	0.0796
	ASV	0.0854	0.086	0.0857	0.0842

Notes: Positive values favour the non-linear model, while negative ones favour the linear quantile regression model.

Figure 1: Feed forward neural network

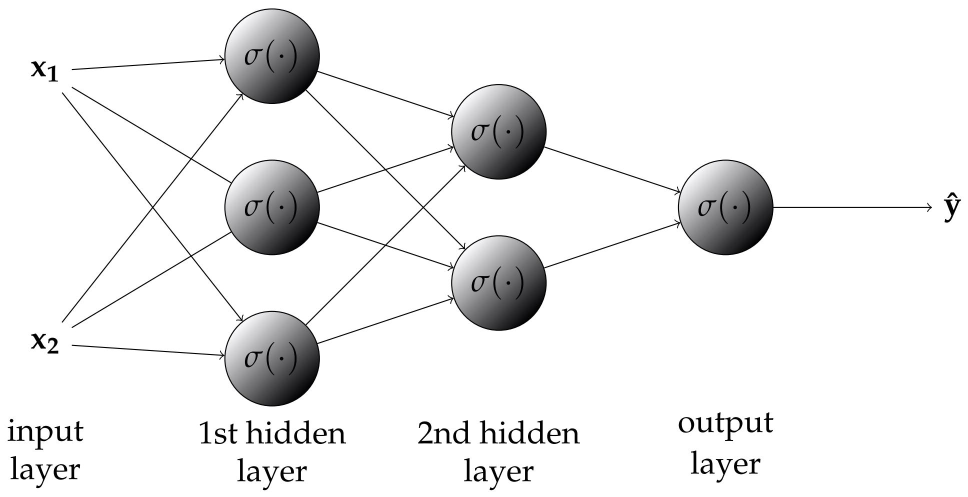


Figure 2: Forecast relative gains.

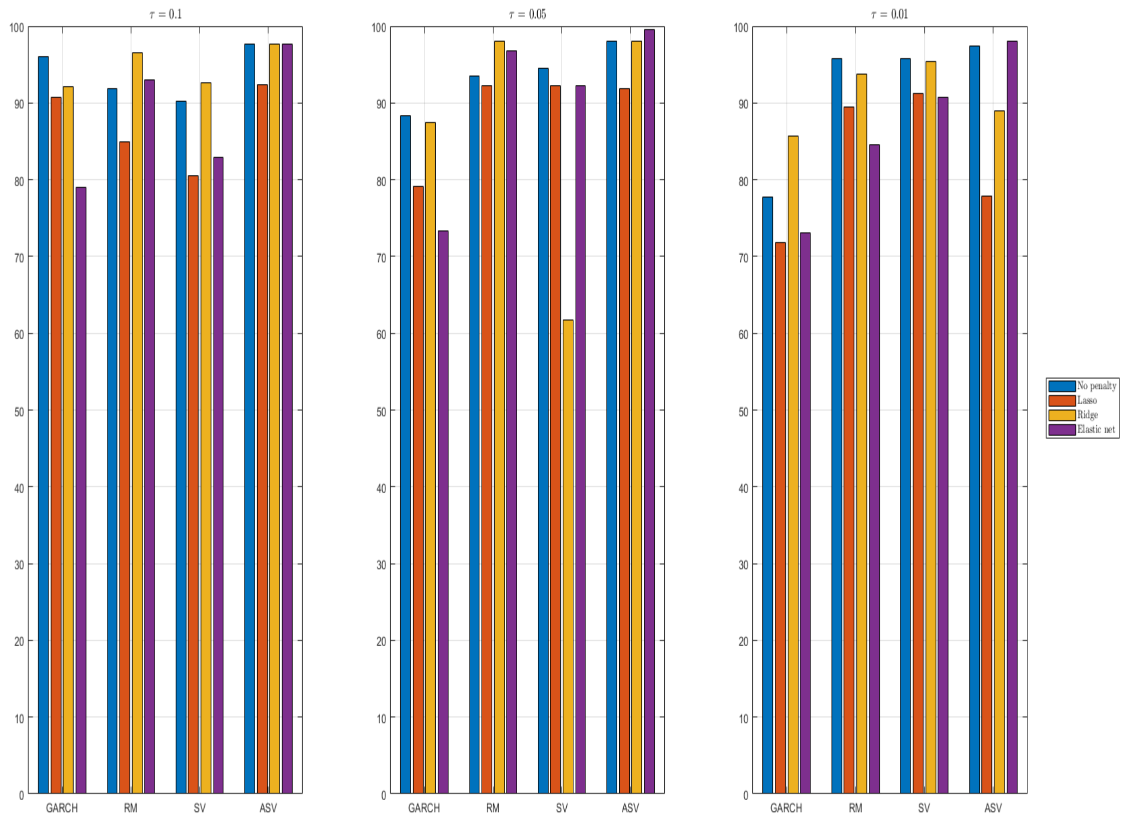
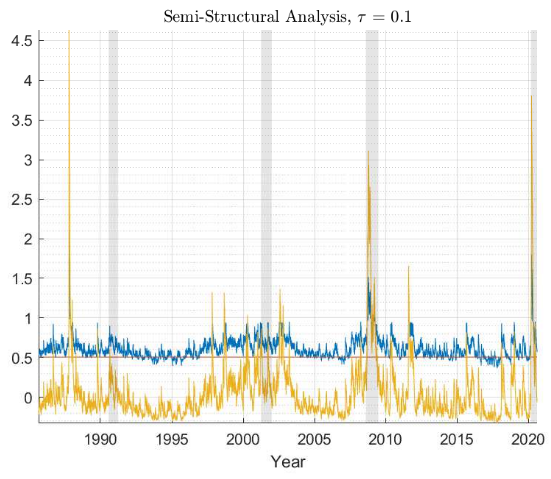
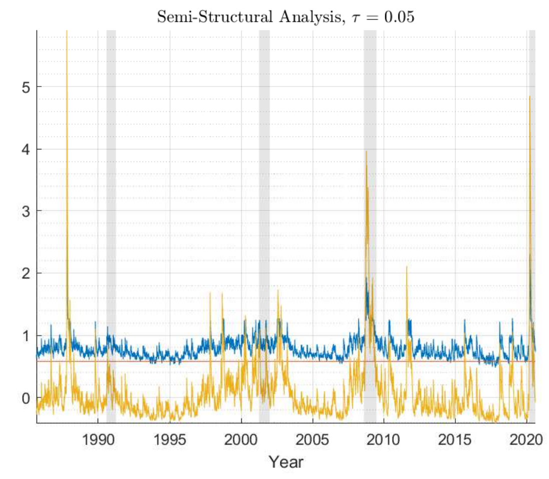


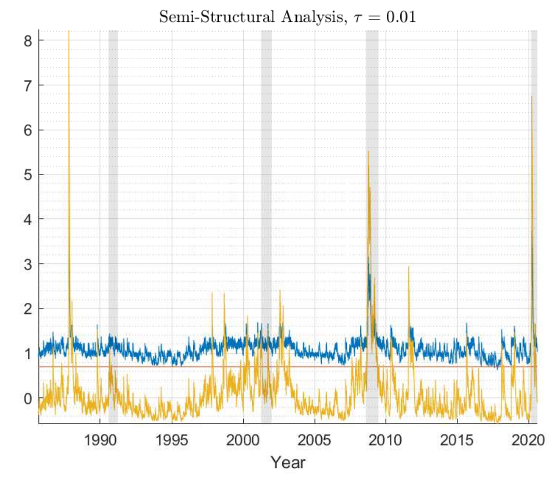
Figure 3: Partial Derivative, SHAP and $\hat{\beta}(\tau)$ for GARCH and RM.



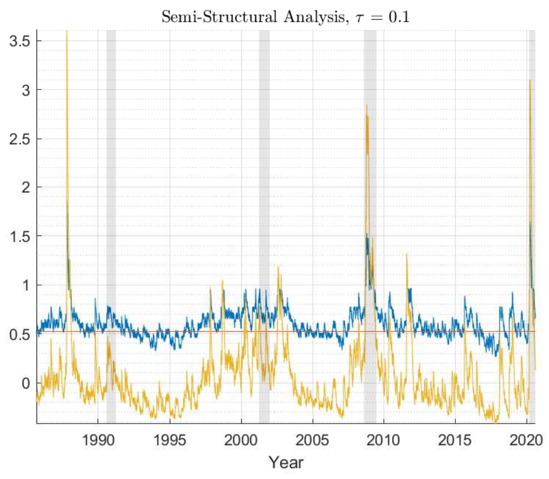
(a) GARCH without penalty



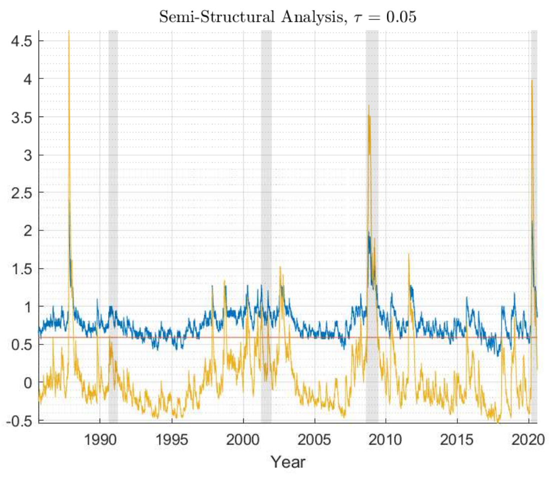
(b) GARCH without penalty



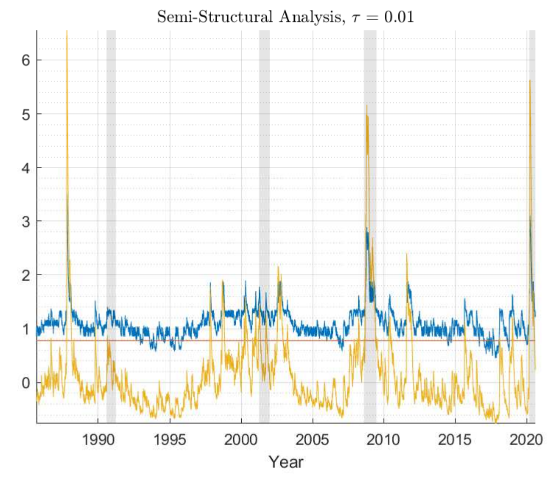
(c) GARCH with Ridge penalty



(d) RM with Ridge penalty



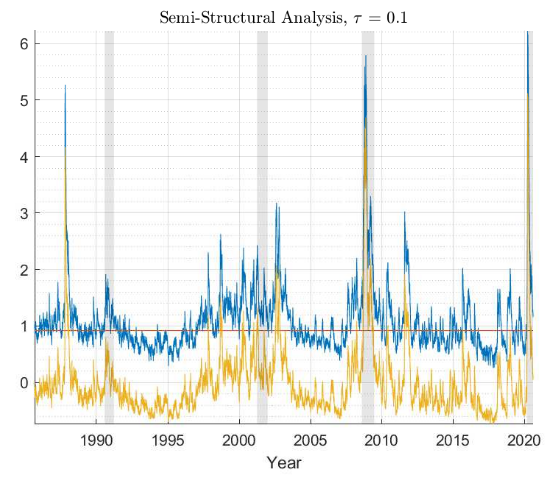
(e) RM with Ridge penalty



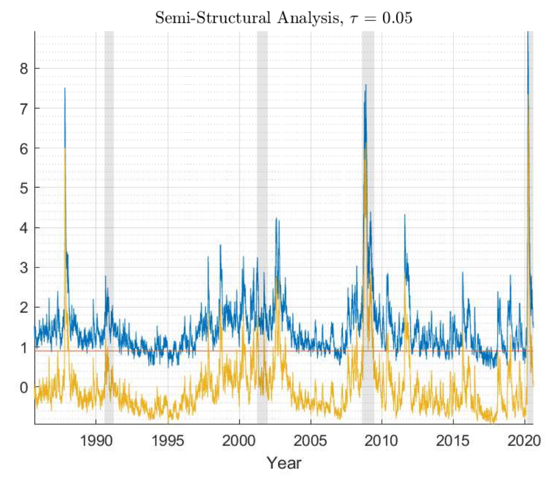
(f) RM without penalty

▬:Derivative, ▬: SHAP, ▬: $\hat{\beta}$, Shade area presents NBER recession indicators

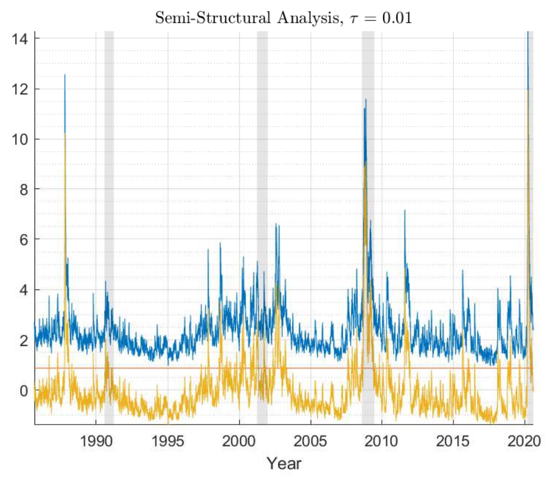
Figure 4: Partial Derivative, SHAP and $\hat{\beta}(\tau)$ for SV.



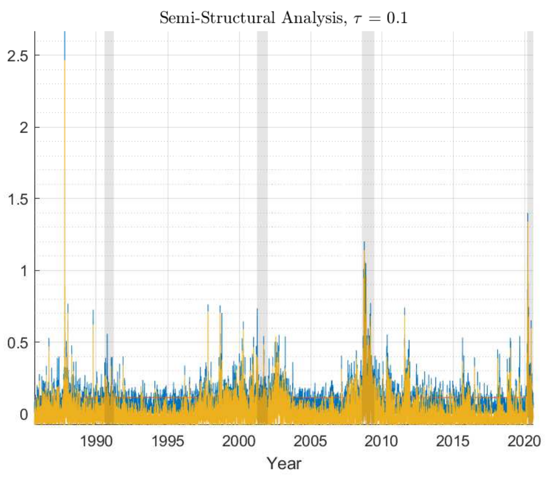
(a) SV with Ridge penalty



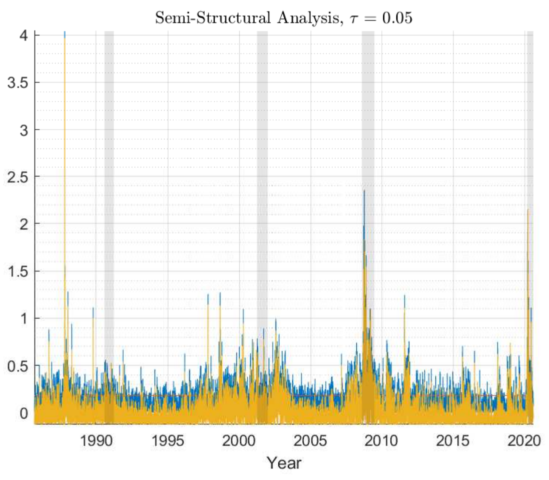
(b) SV without penalty



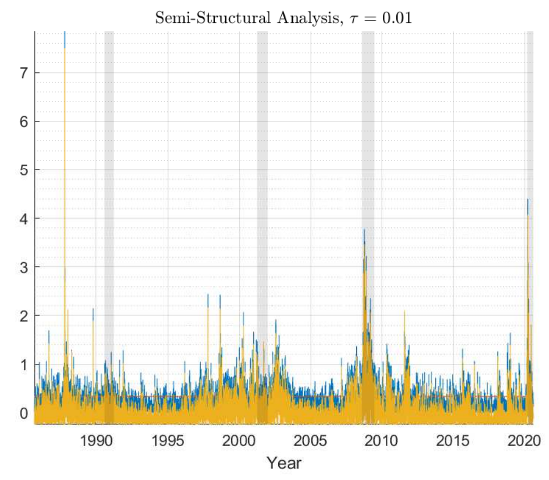
(c) SV without penalty



(d) VaR lagged values with Ridge penalty



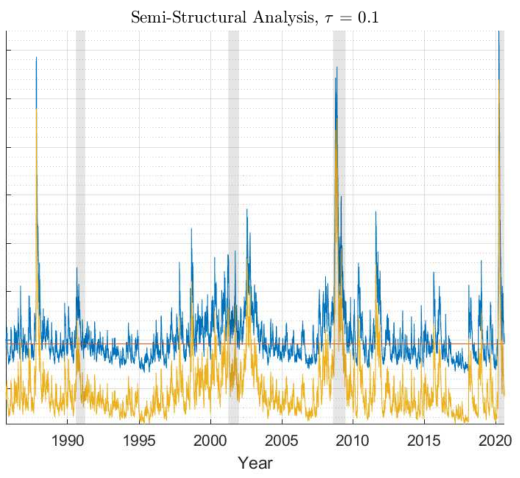
(e) VaR lagged values without penalty



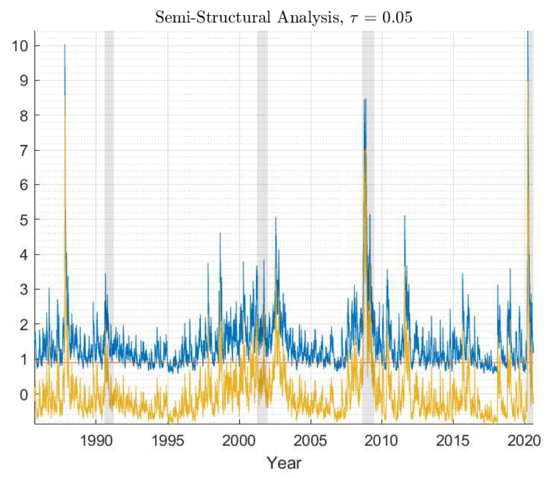
(f) VaR lagged values without penalty

—:Derivative, —: SHAP, —: $\hat{\beta}$, Shade area presents NBER recession indicators

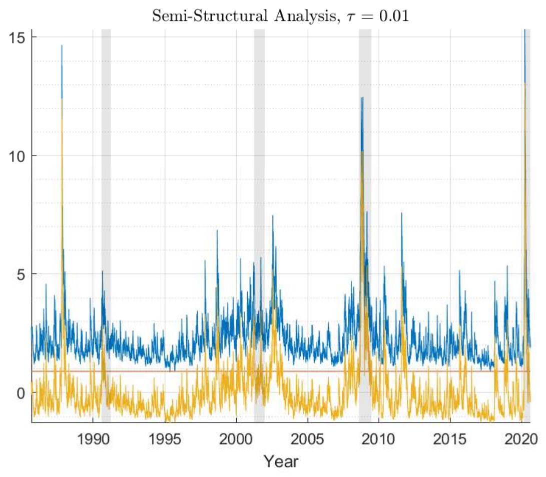
Figure 5: Partial Derivative, SHAP and $\hat{\beta}(\tau)$ for ASV.



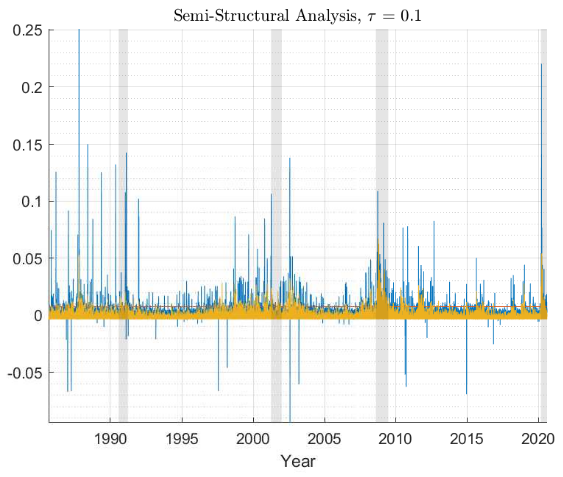
(a) ASV without penalty



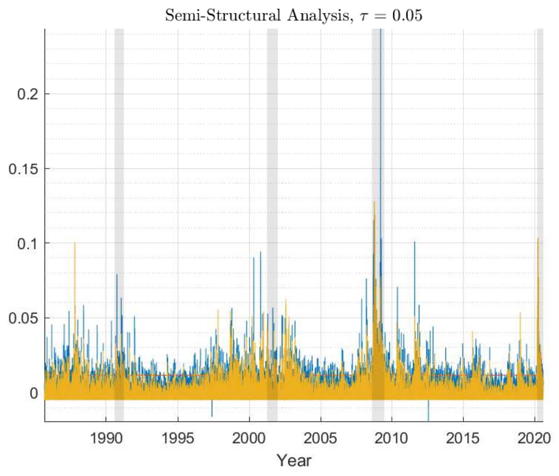
(b) ASV with Elastic Net penalty



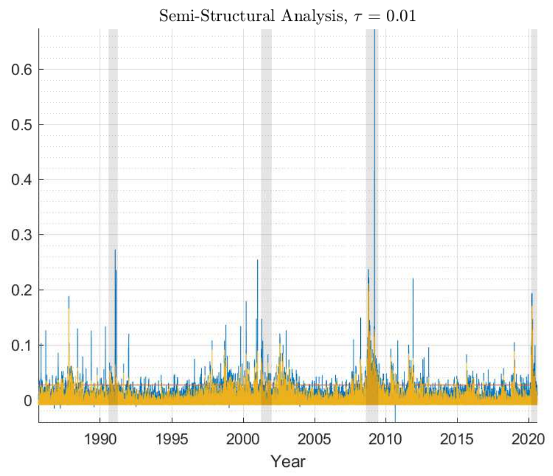
(c) ASV with Elastic Net penalty



(d) $S\&P500$ positive values without penalty



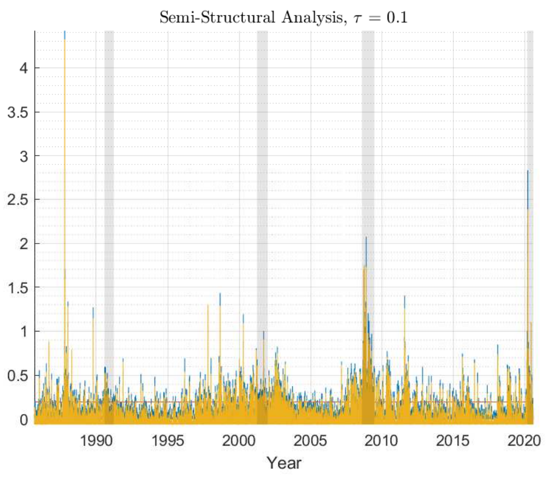
(e) $S\&P500$ positive values with Elastic Net penalty



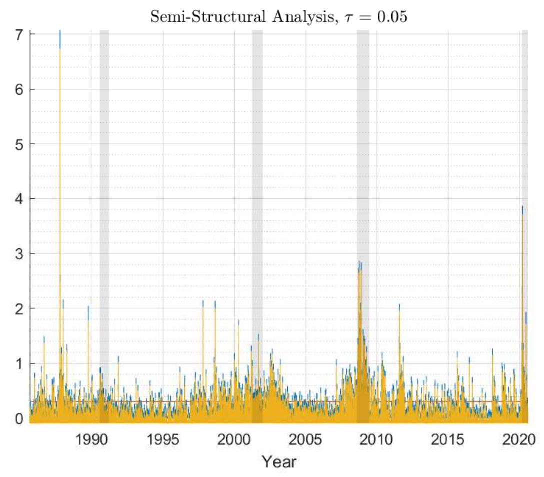
(f) $S\&P500$ positive values with Elastic Net penalty

▬:Derivative, ▬: SHAP, ▬: $\hat{\beta}$, Shade area presents NBER recession indicators

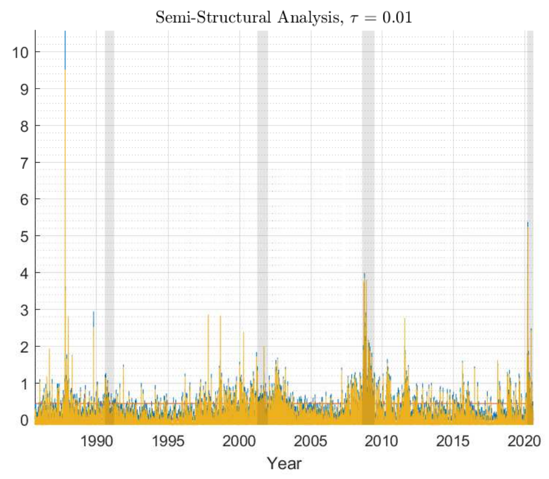
Figure 6: Partial Derivative, SHAP and $\hat{\beta}(\tau)$ for ASV.



(a) $S\&P500$ negative values without penalty



(b) $S\&P500$ negative values with Elastic Net penalty



(c) $S\&P500$ negative values with Elastic Net penalty

—:Derivative, —: SHAP, —: $\hat{\beta}$, Shade area presents NBER recession indicators

The Data Analytics for Finance and Macro (DAFM) Research Centre
Forecasting trends is more important now than it has ever been. Our quantitative financial and macroeconomic research helps central banks and statistical agencies better understand their markets. Having our base in central London means we can focus on issues that really matter to the City. Our emphasis on digitalisation and the use of big data analytics can help people understand economic behaviour and adapt to changes in the economy.

Find out more at kcl.ac.uk/dafm

KING'S BUSINESS SCHOOL

Data Analytics for Finance & Macro Research Centre