Semi-Nonparametric Forecasting via ANN-Sieves

In Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.)

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
CV	Cross Validation
DCC	Dynamic Conditional Correlation
DGP	Data Generating Process
GAM	Generalized Additive Model
GARCH	Generalized Auto Regressive Conditional Heteroskedasticity
GLM	Generalized Linear Model
LL	Local Linear (Nonparametric Regression)
MC	Monte Carlo
MGARCH	Multivariate GARCH
MLP	Multilayer Perceptron
MSE	Mean Squared Error
$\mathcal{O}$	Big O (Limiting Behaviour)
OLS	Ordinary Least Squares
PLS	Partial Least Squares (Regression)
ReLU	Rectified Linear Unit (Activation Function)
RMSPE	Root Mean Squared Prediction Error
SANN	Semi-(non)parametric Artificial Neural Network
STN	Signal To Noise
SUR	Seemingly Unrelated Regression
SVD	Singular Value Decomposition
VAR	Vector Auto Regression
VaR	Value at Risk

1. Introduction

Providing accurate forecasts of economic and financial processes is one of the core tasks of economists, as businesses and policy makers rely on them. Yet, forecasting is also an area where failures are frequent. During the years of the great recession, many models failed to provide the forecasts needed, even a whole "class of models failed to see the crisis coming" (Solow, 2010). Having more accurate and well understood estimations of the underlying economic processes would provide a first step towards improving their credibility.

Often, economic theory does not suggest a specific functional form on, for example, time series. For the purpose of simplicity a (parametric) linear form is still the go-to standard in many applications. This seems contrary to the fact that the empirical evidence suggests that many economic time series seem to have nonlinear forms (see eg. Franses and van Dijk, 2000 for examples of financial phenomena). Nonlinear extensions of such parametric models exist, but they often suffer from misspecification, as a functional form must still be imposed. In recent years, due to the increased availability of data, nonparametric approaches have gained popularity. Their distribution-free techniques provide the possibility to estimate and forecast without as many assumptions as in standard parametric models. This does not come without cost though, nonparametric methods suffer from the curse of dimensionality[1], which makes them inaccurate in high dimensional problems. Semi-(non)parametric[2] approaches have provided promising results in recent years in terms of forecasting performance (see eg. Zhang, 2003) and accurate estimation of parameters (see eg. Härdle, Liang, and Gao, 2000).

A class of nonlinear models summarized under the term Artificial Neural Networks (ANN) began to be popularized throughout the statistical and econometric community in the early 1990s. As we will show in section 2.1, they are a versatile tool which nests many other estimators as special cases. Yet, they often involve a large number of free parameters, which makes them prone to overfit the data. Further, as will be discussed, the parameters of a standard ANN do not have a natural interpretation that statisticians and econometricians are used to, which means they are often treated as black-boxes. This

[1] Roughly, this means that nonparametric estimators tend to converge very slowly in higher dimensional problems. This is due to the fact that the volume of data space increases exponentially with a new column in the dataset, but the size of the dataset only increases linearly with each observation.

[2] A note on terminology: we will refer to a model as semi-parametric if we are interested in the values of the parametric part and semi-nonparametric if we are interested in the values of the parametric and nonparametric part.

goes against the rational, that a forecast should also make sense and that its results are interpretable. If this is not the case, Solow's critique from above would apply directly. A semiparametric modelling approach using ANNs provides a middle ground between their flexibility and black-box nature, by providing (at least partially) interpretable results.

The aim of this thesis is to review the existing statistical framework for ANNs and to analyse how it can be incorporated into a semi-(non)parametric model structure that allows inference from its estimates. Further, based on the recent success that ANNs had in the *M4 forecasting competition* in terms of forecasting performance, we intend to provide a framework of settings, when such models might lead to better forecasting results than conventional parametric and nonparametric methods.

To obtain asymptotic results for ANNs, we can follow the framework of sieve estimation as done by eg. Chen and Shen, 1998. This also has the advantage that it establishes ANNs as nonparametric estimators. Semi- and nonparametric modelling has by now been well studied, and we can use theory and identification techniques developed by eg. Li and Racine, 2006, to avoid treating ANNs as a separate estimator class. Incorporating ANNs in such a general nonparametric framework enables us to provide a method for inference on the parametric part of an ANN based semi-nonparametric model. From a heuristic perspective, combining ANNs and linear models into such a semi-nonparametric model also has a positive effect on the predictive performance of the model. As stated above, Zhang, 2003 obtained results from a semiparametric ARIMA-ANN that were superior in terms of forecasting performance than each model applied separately.

We begin with a motivating example that shows how linear and ANN models might complement each other and provide better forecasting results. The rest of the thesis is then organised as follows: section 2.1 provides an introduction to the relevant theoretical background of ANNs, which is then used for the statistical analysis of nonlinear sieve estimators in section 2.2. The results are then combined and a new semi-nonparametric estimator is proposed in section 2.3 and extended to the multivariate case in section 2.4. Finite sample simulations are considered in section 3 and finally the estimator is applied to real world data in section 4. Proofs are collected in appendix A and auxiliary non-technical results in appendix B.

1.1 Motivating Example

As stated, parametric models need to be correctly specified to perform both accurate forecasting and inference tasks. A special case of nonlinear models are *partially linear models*. As their name suggests, they combine a linear model augmented with a nonlinear part. An example is a data generating process (DGP) that follows model B.1. We generate a noise free time-series process and estimate both an ARIMA model and an

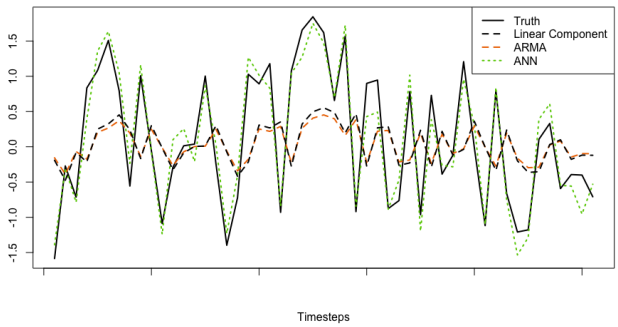

FIGURE 1.1: Fit of a linear ARIMA model and an ANN. The true series was decomposed and the linear part isolated (dashed line).

ANN and investigate the in-sample fit. The visual results of the example can be seen in figure 1.1. The (optimized) ARIMA model chooses the lag right order with only one autoregressive lag and its estimate of the AR coefficient is also close to the true parameter (0.3). It fits the parametric part of the series rather well, but the model is clearly unable to reflect the nonlinear part. In contrast to that, a very simple ANN is able to capture a significant part of the nonlinear dynamics. As this example did not yet include noise, the biggest disadvantage of ANNs, their frequent overfitting, did not yet pose a problem. By combining the two approaches, we hope to combine the simplicity and easy interpretability of linear models with the flexibility of ANNs to improve our forecasting results.

2. Methodology

2.1 Artificial Neural Networks

ANNs are often treated as a separate model class, but they bear close similarities to other known statistical estimators. Their traditional field of application is in image recognition or other fields where high signal-to-noise (STN) ratios are common. In fields where the STN ratio is lower, ANNs frequently overfit and are therefore less often used.

Although parameters are estimated given some criterion (for regression usually the mean squared error (MSE)), the rationale behind the estimation of ANNs differs from standard statistic procedures. From a statistical point of view, the criterion function differs from a likelihood function, as the parameters are not optimized to describe the best fitting random distribution given the data. Rather, the goal is to approximate some unknown (and from a modelling perspective uninteresting) function. In the present section, we will consider a simple version of the general ANN framework and point out

similarities to other statistical procedures.

2.1.1 Single Hidden Layer Artificial Neural Networks

To start with, we follow Kuan and White, 1994 and consider a model that takes the inputs $x_i, \in \mathbb{R},\ i = 1, ..., p$ and maps them to some output units $j, j = 1, ..., \nu$. Each input is then weighted a by factor $\gamma_{i,j} \in \mathbb{R}$. In the output unit j the weighted inputs are then combined by a simple additive rule, so each unit produces the following output:

$$\sum_{i=1}^{p} x_i \gamma_{i,j}, \quad j = 1, ..., \nu$$

For simplicity, we define $\tilde{x}$ as the x-vector with an additional "bias unit" $x_0 = 1$, which corresponds to the intercept in a linear equation. Rewriting the above we get the general form of the input combination:

$$f_j(x, \gamma) \equiv \tilde{x}' \gamma_j, \quad j = 1, ..., \nu \tag{2.1}$$

Note that (2.1) corresponds to the standard linear model when $\nu = 1$ (ie. only one output unit). With $\nu > 1$ and the layers staked according to$f(x, \gamma) \equiv (I \otimes \tilde{x})\gamma$ and $\gamma = (\gamma_1', ...\gamma_\nu')$ we get the linear SUR model. We note that this model contains the, for time series popular, VAR model as a special case. Having inputs that provide signals "in parallel" to units ahead is called *parallelism* in the ANN literature and is well known from the standard multiple linear regression (where the dependent variables are provided in parallel). A further feature of ANNs that provides their flexibility is the *nonlinear response unit*. In equation (2.1) the output unit simply linearly summed up the weighted inputs, but the sum may be transformed by a function (note that we keep the inner function equal to equation (2.1)). This yields the *single unit perceptron*:

$$f_j(x, \gamma) = G(\tilde{x}' \gamma_j), \quad j = 1, ..., \nu\ , \tag{2.2}$$

where G is called the *activation function*. Note again, that in the special case of $\nu = 1$ and the choice of the logistic function as activation, we get the standard logit model. Figure 2.1 (a) depicts the single unit perceptron, which can be interpreted as the ANN notation of a generalized linear model (GLM), the layers were therefore named accordingly.

What further differentiates ANNs from the standard linear models is their third characteristic, *multiple layer processing*. ANNs usually contain "hidden" layers, that is, arranged

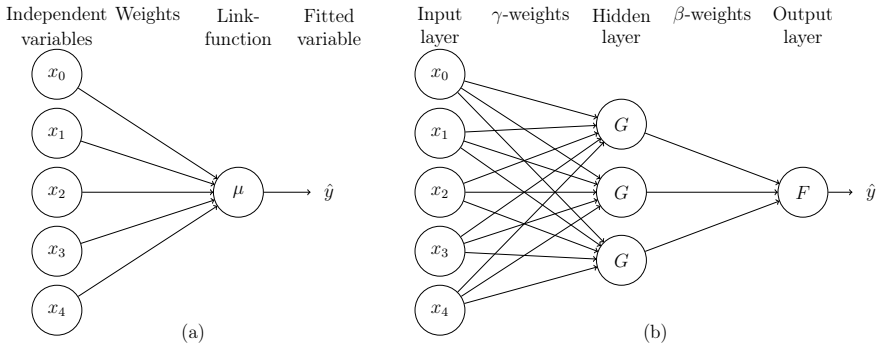

FIGURE 2.1: (a): A single unit perceptron (GLM). (b): A single hidden layer ANN

layers of response units between the input and output layers, that are not directly observable. These hidden layers treat the output of the previous layer as their inputs.[1] Such Multilayer Perceptrons (MLP) build the core of modern deep learning. For our theoretical considerations we treat the model with a single hidden layer and a single ouput unit (ie. $\nu = 1$), figure 2.1 (b) illustrates the architecture. We can then write this model as:

$$f(x, \theta) = F(\beta_0 + \sum_{k=1}^{m} G(\tilde{x}'\gamma_k)\beta_k), \tag{2.3}$$

where m is the number of the so called hidden units (processing units as the ν in (2.1)) in the hidden layer. $\beta_k,\ k = 0, ..., m$ are the connection weights from hidden unit k to the (in this case single) output unit and $\theta = (\beta_0, \beta_m, \gamma_1', ..., \gamma_m')'$ is the vector of all network weights. For ease of notation we will set F equal to the identity function (which is generally done when faced with a regression task), unless otherwise stated. This means it will be omitted from further notation. Note that this model again nests a well known model. By setting $m = p,\ \gamma_{0,k} = 0$ and $\gamma_{i,k} = 1$ if $i = k$ else 0, we obtain:

$$f(x, \theta) = \beta_0 + \sum_{k=1}^{p} G_k(x_k)$$

Which corresponds to a generalized additive model[2] (cf. Hastie and Tibshirani, 1990).

[1] There are ANNs which are not restricted to taking the input from the previous layer(s). In complex architectures they may also use input from output units that are further away. We will focus our analysis on the case where only the immediate preceding layer is used as input, these ANNs are therefore called the "feed-forward-networks".

[2] Note that in the ANN architecture, we generally choose a singe class of activation function per layer, which is why in equation (2.2) the function G is not indexed by the units k.

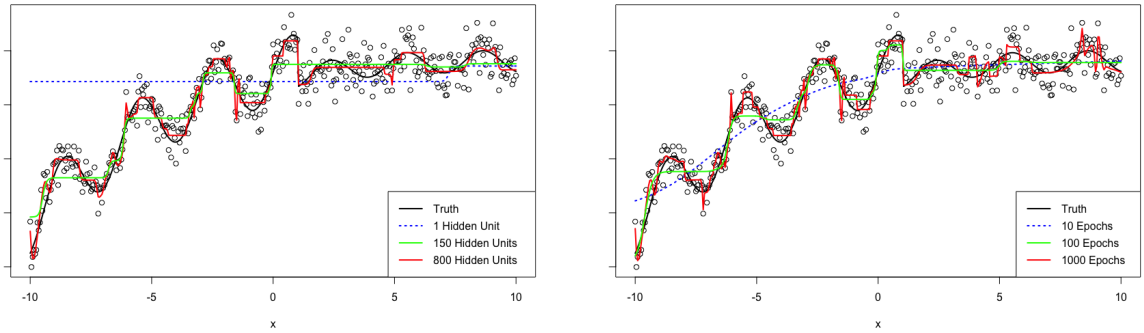

FIGURE 2.2: Different sources of overfitting for ANNs. Left, overfitting based on hidden units and right overfitting for different number of epochs (with 800 hidden units). Note that the dotted blue line in the right figure is still close to a linear solution due to early stopping. Data was generated from model B.2.

2.1.2 Estimation

The structure of an ANN permits the use of highly parallelizable gradient descent algorithm with error backpropagation. In contrast to the standard OLS optimization, they are inexact, but permit optimization on subsets of the data, which is often necessary for large datasets. The structure of the algorithm is useful for some properties that we will discuss, but is not directly needed to understand the subsequent analysis. Therefore, we leave the derivation to the appendix A.1.[3]

The flexibility of ANNs has the disadvantage that they will often optimize the parameters at the global minimum of the training data. If the observed data y contains a noise term ε (ie. $y = f(x) + \varepsilon$), this means the ANN will start to explain y rather than the true underlying process $f(x)$. There are multiple ways to prevent overfitting. One way is called *early stopping*. Because the dataset is passed to the algorithm multiple times[4] until it converges, stopping the iterative procedure before it reaches the global minimum can prevent overfitting. This has the effect of shrinking the model towards the linear solution (cf. Hastie, Tibshirani, and Friedman, 2009). We show the effect in the appendix A.1 and present the result visually in figure 2.2.

More akin to methods from the econometrics literature is explicit parameter shrinking, which bears similarities to methods such as ridge regression or the LASSO. It is achieved by adding a norm penalty term to all or specific parts of the loss function $L(\theta, x)$ such that the optimization problem effectively becomes (cf. Goodfellow, Bengio, and Courville, 2016):

$$\hat{\theta} = \arg\min_{\theta} \left(L(\theta, x) + \alpha\Omega(\theta) \right) ,$$

[3]Further, as the computational estimation and different algorithms are a huge field of research in the computer science literature, a detailed discussion would go beyond the scope of this thesis and is omitted.

[4]Where one complete pass of the dataset is called an "epoch".

where $\alpha \geq 0$ is a tuning parameter and Ω a norm penalty on the parameters θ. Finally, as will be discussed in section 2.2, it is important to have bounds on the weights of the ANN. In order for the ANN to adjust its parameters optimally, standardization is usually applied to the inputs before passing the data on to the algorithm.

2.1.3 Universal Approximation Property

The reason for the flexibility of ANNs is, that they possess the *universal approximation property*. From the large variety of activation functions available, we will restrict our analysis to the two most popular choices, squashing functions and the rectified linear unit (ReLU). A squashing function is nondecreasing such that $f(x) : \mathbb{R} \to [0, 1]$ with $\lim_{x\to-\infty} = 0$ and $\lim_{x\to\infty} = 1$, with the logistic function as a popular example. The ReLu activation is defined as $f(x) = \max\{0, x\}$. Among others, Hornik, Stinchcombe, and White, 1989 have established, that ANNs with suitably many hidden units and layers can approximate any function arbitrarily well. To properly define the universal approximation, we adopt their definition of denseness and convergence:

Definition 2.1.1 (Denseness and uniform convergence)**.** A subset S of a metric space (X, ρ), where ρ is the metric, is *ρ-dense* in a subset T if for every $\varepsilon > 0$ and for every $t \in T \; \exists \; s \in S$ such that $\rho(s, t) < \varepsilon$. To establish the convergence for functions, we define further: Let C^d be the set of continuous functions from $\mathbb{R}^d$ to $\mathbb{R}$ and $K \subset \mathbb{R}^d$ a compact subset. For $f, g \; \in C^d$ let $\rho_K(f, g) \equiv \sup_{x\in K} |f(x) - g(x)|$. A subset S in C^d is then said to be *uniformly dense on compacta* in C^d if for every K, S is ρ_K dense in C^d. Further, a sequence of functions $\{f_n\}$ converges to a function f uniformly on compacta if for all (compact) $K \subset \mathbb{R}^d$, $\rho_K(f_n, f) \to 0$ as $n \to \infty$.

Intuitively, this means that an element of S or the sequence of functions f_n can approximate to any degree of accuracy an element of T or a function $f \in C^d$ respectively. The following theorem then summarises the findings of Hornik, Stinchcombe, and White, 1989 for squashing functions and Leshno et al., 1993 for ReLU activations[5], that will be used throughout this thesis.

Theorem 2.1.1 (Universal approximation theorem)**.** Let $G(\cdot)$ be the activation function. An ANN of the form of (2.3) is, for every $d \in \mathbb{N}$, uniformly dense on compacta in C^d, if $G(\cdot)$ is a squashing function (Hornik, Stinchcombe, and White, 1989) or a ReLU activation (Leshno et al., 1993).

[5]We note that already in the earlier paper of Hornik, Stinchcombe, and White, 1989 the universal approximation theorem was established for ANNs with ReLU activation functions (cf. Theorem 2.2 of the paper). What is new in Leshno et al., 1993 is that the property is established for single hidden layer ReLU ANNs.

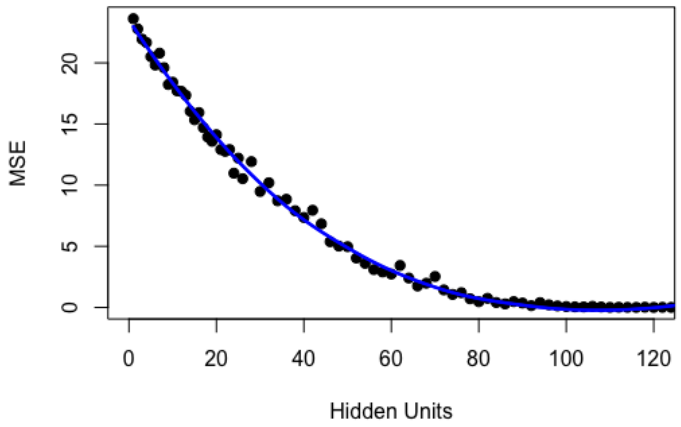

FIGURE 2.3: Shows the decreasing MSE with increasing number of hidden units and the LOESS smoothed decrease (blue) for model B.1. Note that the MSE does not decrease strictly with the number of hidden units. This is an indicator that the gradient descent algorithm got stuck in a local minimum.

This establishes the asymptotic approximation properties, but still leaves open the choice of the number of hidden units for a concrete problem. Figure 2.3 illustrates the approximation property of the single-hidden layer ANN with an increasing number of hidden units, for the adapted autoregressive chaos model (B.1). Again, the model did not contain any noise, which means that there is no "downside" to adding more hidden units. In practice, with noisy data, there is a tradeoff that we will investigate in the next section.

2.2 Sieve Estimation

The method of sieves is a flexible estimation technique generally attributed to Grenander, 1981. In semi- or nonparametric regression settings, the set of parameters θ that describe a process, is situated in a not necessarily finite dimensional space, which makes the optimization problem no longer well posed (Chen, 2007). The method of sieves approximates the infinite parameter space with a sequence of approximating parameter spaces that will be dense in the original (possibly infinite) space in the limit. To illustrate the approach we consider a generic time series model of the form:

$$y_t = \psi(x_t) + u_t, \quad t = 1, ..., n, \tag{2.4}$$

where x_t is a $q \times 1$ vector (with possibly lagged values of y_t) and u_t is white noise with $\mathbb{E}[u_t|x_t] = 0$ and $\mathbb{E}[u_t^2|x_t] = \sigma^2$. If we take the least squares approach, the optimal

estimator can be written as:

$$\hat{\psi}(\cdot) = \arg \min_{\psi(\cdot)\in\Psi} \frac{1}{n} \sum_{t=1}^{n} (y_t - \psi(x_t))^2$$

In parametric settings the functional space of Ψ is typically strictly limited to an additive form with the parameter vector bound in some space $\mathbb{R}^q$. Grendander's idea consists of estimating ψ by a parametric class of increasingly complex sieve-functions:

$$\psi_n \in \Psi_n = \left\{ f : f(x;\theta) = \sum_{k=1}^{r_n} \theta_k G_k(x) \right\}, \tag{2.5}$$

where G_k is a sequence of basis functions and r_n denotes the number of terms in the sieve space when we have a sample of size n. With the right choice of G_k, f can then approximate the true function $\psi(\cdot)$ arbitrarily well with increasing r_n.[6] The idea is then, to let r_n increase with n. This feature distinguishes the estimation procedure from standard parametric techniques, which assume a fixed and finite dimensional parameter space. The function f is parametrized by $\theta \in$ (the sieve space) Θ_n. To properly describe Θ_n we define the infinite dimensional parameter space of the nonparametric problem as Θ with a metric d and its corresponding approximation spaces Θ_n. To have well defined properties, we let the approximation spaces be compact and nondecreasing in n and all $\Theta_n \subseteq \Theta$. The sieve estimators is then given by:

$$\hat{\psi}(\cdot) = \arg \min_{\theta\in\Theta_n} \frac{1}{n} \sum_{t=1}^{n} (y_t - \psi_n(x_t))^2 , \tag{2.6}$$

with the definition of ψ_n from above. This then becomes a feasible estimation problem because in the parameter space of $f(\cdot, \theta)$ the optimization is well defined. Note that in order to achieve consistency, we need the following assumption to hold (adapted from Newey, 1997 for series estimation):

$$\frac{1}{r_n} \to 0 \text{ and } \frac{r_n}{n} \to 0 \text{ as } n \to \infty$$

Note that the requirement on the left hand side ensures that the parameter space is infinite in the limit of n but grows slower than the sample size (right condition). The conditions can also be closely compared to those of nonparametric kernel regression.

The reason why sieve estimation is popular is its simplicity and also the ease with which restrictions such as additivity or previous knowledge on the distribution of errors, such as excess kurtosis, can be imposed (cf. Chen, 2007). Further, certain shape constraints (such as concavity) often arise in financial settings (Andersen et al., 2016) and could

[6] For example, with G_k a polynomial function of degree k we can employ the Stone-Weierstrass theorem.

be used to improve the resulting model. Finally, sieves are *global* estimators, as they approximate the function over the whole set of data values and not just locally, like for example, nonparametric kernel methods. As we will see later, the method of sieves also has desirable properties in higher dimensional settings.

As stated earlier in section 2.1, neural networks optimize a certain loss function, but usually that function cannot be interpreted as a likelihood that is maximized. The sieve framework described here gives us the possibility to do so nevertheless. More formally than above, we follow Chen, 2007 and define the estimation problem as follows:

Definition 2.2.1 (Approximate Sieve Extremum Estimate)**.** Define Q as some population criterion function $Q : \Theta \to \mathbb{R}$, which is uniquely maximised at $\theta_0 \in \Theta$. $\widehat{Q}_n : \Theta \to \mathbb{R}$ is its sample equivalent based on the data $\{X_t\}_{t=1}^n$, such that it converges to the true Q with $n \to \infty$. In a finite dimensional (ie. parametric) setting, we would maximise $\widehat{Q}_n$ over Θ which yields an extremum estimator. But in an infinite dimensional space the optimization is no longer well defined. Instead, we let $\hat{\theta}_n$ be the approximate maximizer of $\widehat{Q}_n(\theta)$ over the sieve space Θ_n that is:

$$\widehat{Q}_n(\hat{\theta}_n) \geq \sup_{\theta \in \Theta_n} \widehat{Q}_n(\theta) - \mathcal{O}(\eta_n), \text{ with } \eta_n \to 0 \text{ as } n \to \infty \,, \tag{2.7}$$

where η_n denotes an error margin that converges to zero. $\mathcal{O}$ denotes the standard big-O notation[7].

Put simply, this means that the estimated maximiser over the sieve space converges to the supremum of the sample criterion over the sieve space. The equation can be interpreted as a simplification of the original problem that was situated in the (possibly infinite) dimensional parameter space Θ and now belongs to the (finite but variable - sized) parameter space Θ_n. With the above defined property that the sieve spaces are nondecreasing, we use the projection π_n and define $\pi_n\theta_0 \in \Theta_n$ such that $\varphi_n \equiv d(\theta_0, \pi_n\theta_0) \to 0$ as $n \to \infty$, which is the *approximation error* due to the lower dimension of the sieve space compared to the original parameter space.

If we define the inverse of the MSE for a single observation from (2.6) as $l(\theta, (y_t, x_t')')$, we can rewrite the optimization problem as:

$$\sup_{\theta \in \Theta_n} \widehat{Q}_n(\theta) = \sup_{\theta \in \Theta_n} \frac{1}{n} \sum_{t=1}^{n} l(\theta, (y_t, x_t')') \tag{2.8}$$

This specifies the sieve estimator as an M-estimator, which allows us to give some interpretation to the loss function. Instead of just maximizing the objective function (which would mean minimizing the loss), we are maximising the empirical criterion function $\widehat{Q}_n$ over the (simpler) approximation space $\Theta_n \subset \Theta$.

[7] $f(n) = \mathcal{O}(a_n)$ means $\exists M < \infty$ s.t. $|f(n)| \leq Ma_n, \forall n \in \mathbb{N}$ (eg. the data set size).

2.2.1 Sieve Order

When we consider the functional sieve form of equation (2.5) it becomes clear that the ANN of equation (2.3) can be stated as a sieve estimator (if we keep $F(\cdot)$ the identity and let the number of hidden units grow with n[8]). Given the universal approximation property from theorem 2.1.1, we also know that in the limit, ANNs can approximate any function, which guarantees that the sieve problem is well defined. The choice of the sieve order r_n then becomes the choice of the number of hidden units in the ANN.[9] As it is common across all nonparametric methods, ANN sieves also have a bias-variance tradeoff, which can be controlled by the number of hidden units as already seen in the left pane of figure 2.2. Although sieve estimators are parametrized, they approximate a function of unknown form. We will therefore first illustrate the tradeoff using the mean integrated squared error (MISE) to make the relation to other nonparametric function approximations clear. The definition of the MISE is:

$$\text{MISE} = \mathbb{E}||f_n - f||_2^2 = \mathbb{E}\int \big(f_n(x) - f(x)\big)^2 dx \ ,$$

where $f_n(x)$ is the ANN approximation of the sieve order r_n and f the function of interest. To derive the following proposition we use the property of ANNs that they represent the solution of a linear equation in the output layer. This allows us to exploit some known properties of linear models. For example, we can assume that the solution to the optimization problem in the final hidden layer can be found with a standard linear technique. Therefore, we define $g_k(x)$ the (nonlinearly) transformed $(n \times 1)$ output produced by the k^{th} unit in the final hidden layer and $\mathcal{G}(x)$ the $(n \times r_n)$ corresponding matrix of all r_n units and $\mathcal{G}(x_t)$ the $1 \times r_n$ transformed feature vector for observation t. We can then follow the methodology of Hansen, 2014 who considers spline based sieves and estimate equation (2.4) with the corresponding (ANN)-sieve as:

$$\psi_n = f_n(x, \theta) = \beta_0 + \sum_{k=1}^{r_n} \beta_k g_k(x) = \mathcal{G}(x)\beta_n \ ,$$

$$\hat{\beta}_n = (\mathcal{G}(x)'\mathcal{G}(x))^{-1}\mathcal{G}(x)'Y \ ,$$

where we include the γ weights implicitly as defined above and allow the first column of $\mathcal{G}(x)$ to be a constant. Further, we indexed the parameter vector β_n because it will grow with an increasing sieve order. Note that contrary to the standard linear case, where

[8] To make the change to a variable number of hidden units explicit, we will adopt the term r_n to indicate the number of hidden units in line with our considerations about sieve estimation from above.

[9] In section 2.3.4 we will show that the number can be regularized as described in the previous section. In a regularized model, we consider the number of hidden unit r_n the number of non-zero hidden units.

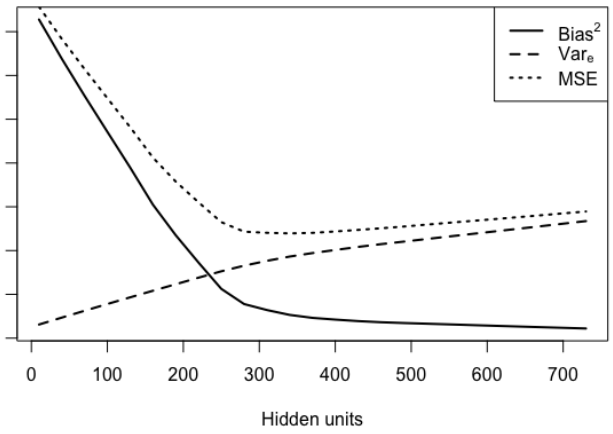

FIGURE 2.4: Pointwise MSE decomposition of an ANN Sieve estimator, with increasing number of hidden units, we can observe the standard bias-variance tradeoff for nonparametric methods. Data was generated from model B.2.

the only error is the projection error, we have an additional approximation error in $\hat{\beta}_n$. We define the approximation error and its squared expectation as:

$$\upsilon_n(x_t) = \psi(x_t) - \mathcal{G}(x_t)\beta_n \,, \tag{2.9}$$
$$\varphi_n^2 = \mathbb{E}[\upsilon_n(x_t)^2] = \int \upsilon_n(x)^2 f(x) dx$$

Taking this together and following the methodology of Hansen, 2014, we are then able to propose the following (proof in appendix A.2):

Proposition 2.2.1 (Bias-Variance Tradeoff for ANN Sieves)**.** For model (2.3) with the identity activation function for the output layer we have a bias-variance tradeoff which can be summarized by

$$\text{MISE}_n(x) = \mathbb{E}\int \big(\hat{\psi}_n(x) - \psi(x)\big)^2 dx = \varphi_n^2 + n^{-1} trace(\mathcal{E}_n^{-1}\Omega_n)$$

$$\mathcal{E}_n = \mathbb{E}[\mathcal{G}(x_t)'\mathcal{G}(x_t)], \;\; \Omega_n = \mathbb{E}[\mathcal{G}(x_t)'\mathcal{G}(x_t)\sigma_t^2], \;\; \sigma_t^2 = \mathbb{E}[u_t^2|X_t]$$

Given that we used a linear model, we note that for the single hidden layer ANN, models with more hidden units will nest smaller (in terms of hidden units) models. If we then consider $r_n' > r_n$ the optimization problem for sieve order r_n implicitly poses $\beta_k = 0$ for all $k = r_n + 1, ..., r_n'$ in the final hidden layer. Therefore, the expected approximation error will be at least weakly decreasing in the number of sieve terms r_n (assuming the maximum of $\widehat{Q}_n$ is found) and hence φ_n^2 will decrease with increasing n. This is a well known property when working with the standard linear model. The second

term corresponds to the asymptotic variance and the term in the trace will (weakly) increase in r_n, as the diagonal is weakly positive. Figure 2.4 depicts the decomposed mean squared error[10]. As mentioned above, sieve estimators are parametrized and we are able to derive an explicit bound on the terms in section 2.3.

With the bias-variance tradeoff dependent on the number of sieve terms, r_n becomes a hyperparameter that is similar to the choice of for example the bandwidth in kernel-type estimators. Similar to the optimal choice for the bandwidth, the optimal number of r_n is dependent on the true underlying function and hence we are not able to get an exact result with noisy data from an unknown distribution. Unfortunately and contrary to the case with kernel methods, there is no rule of thumb on how many sieve terms should be chosen. We will discuss some solutions to this issue in section 2.3.4.

2.2.2 Asymptotic Properties

We now want to study the asymptotic behaviour of ANNs, which can be done by employing the more general sieve theory. Due to the parametric nature of sieve estimation, we can give the convergence of the estimated parameters (in the finite dimensional space) to the true parameter (in the possibly infinite dimensional space), rather than the convergence to the underlying function itself (as was done in the last section). This has two uses, first, convergence rates will permit us to compare ANN sieves to other nonparametric methods. And second, the rates will provide guidance on how fast the estimator will converge to the true function even if we only consider the nonparametric part as a nuisance term.

Because much of economic and financial data consists of time series, we will provide rates of convergence for weakly dependent data that follow a so-called β-mixing process, the iid case is similar. We adopt the definition from Li and Racine, 2006:

Definition 2.2.2 (β-mixing process)**.** Let $\mathcal{M}_t^{t+\tau}$ denote the σ-field generated from the stochastic series $Z_{s\,s=t}^{t+\tau}$. Define:

$$b_\tau = \sup_{t\in\mathbb{N}} \sup_{A\in\mathcal{M}_{t+\tau}^{\infty}, B\in\mathcal{M}_{-\infty}^{t}} |\mathbb{E}[A|B] - \mathbb{E}[A]|$$

The sequence $Z_{t\,t=-\infty}^{\infty}$ is said to be β-mixing if $b_\tau \to 0$ as $\tau \to \infty$

We need impose the mixing condition to ensure that the statistical dependence between two data points in time goes to zero, as the temporal distance between them increases. The β-mixing condition is used throughout the literature because it permits to obtain

[10]Which can be expressed as $\text{MSE} = \mathbb{E}[(f(x) - \hat{f}(x))^2] = \text{Bias}\hat{f}(x)^2 + \text{Var}(\hat{f}(x))$. For a derivation see for example Geman, Bienenstock, and Doursat, 1992.

faster convergence rates than for strong (α)-mixing processes but still allows for a rich variety of economic time series (Chen, Racine, and Swanson, 2001).

For the convergence results we will use a theorem developed in Chen, 2007 who derives it from previous research. As seen in section 2.2.1, a bias-variance tradeoff exists for sieve estimates. Whereas the last layer was sufficient for the previous analysis, we will now analyse the whole sieve space Θ_n. This will also allow us to analyse "deeper"[11] ANNs later on. In order for the sieve-estimate $\hat{\theta}_n$ to converge to the true value θ_0 we need to minimize the approximation error $||\theta_0 - \pi_n\theta_0||$ to avoid asymptotic bias (because the sieve spaces are only in the limit equal to the true function space), but at the same time the sieve space should not be too complex to avoid overfitting. To measure the complexity of the sieve space we use the L_2 metric entropy with bracketing (see eg. Chen and Shen, 1998 for a definition) of a class $\mathcal{F}_n = \{g(\theta, \cdot) : \theta \in \Theta_n\}$. We denote this by $H_{[]}(\omega, \mathcal{F}_n, ||\cdot||_2)$. We can then state the same conditions for the convergence of sieve M-estimators as in Chen and Shen, 1998:

Condition 1. $Y_{t=1}^n$ is a stationary β-mixing sequence, with $\beta_\tau \leq \beta_0\tau^{-\xi}$ for some $\beta_0 > 0, \xi = \gamma - 2 > 0$ (see definition of γ below)

Condition 2. $\exists\ c_1 > 0$ such that for small $\varepsilon > 0$,

$$\sup_{\theta\in\Theta_n:||\theta_0-\theta||\leq\varepsilon} \mathrm{Var}\big(l(\theta, Y_t) - l(\theta_0, Y_t)\big) \leq c_1\varepsilon^2$$

Condition 3. Let $\mathcal{F}_n = \{l(\theta, Y_t) - l(\theta_0, Y_t) : ||\theta_0, \theta||_2 \leq \delta, \theta \in \Theta_n\}\ \exists\ \delta_n \in (0, 1)$ and constants c_2 and $b > 0$ such that:

$$\delta_n = \inf\left\{\delta \in (0,1) : \frac{1}{\sqrt{n}\delta^2}\int_{b\delta^2}^{\delta}\sqrt{H_{[]}(\omega, \mathcal{F}_n, ||\cdot||_2)}d\omega \leq c_2\right\}$$

Condition 4. For any $\delta > 0, \exists\ s \in (0, 2)$ such that

$$\sup_{\theta\in\Theta_n:||\theta_0-\theta||\leq\delta} |l(\theta, Y_t) - l(\theta_0, Y_t)| \leq \delta^s U(Y_t)$$

for some function U with $\mathbb{E}[(U(Y_t))^\gamma]$ for some $\gamma > 2$

The conditions ensure that within a neighbourhood of θ_0, $l(\theta, Y_t)$ is continuous at θ_0 and that $||\theta_0 - \theta||^2$ behaves locally as the variance (Chen and Shen, 1998). Chen, 2007 then gives the following theorem:

[11] There is no universal definition of what constitutes a deep and what a shallow neural network. In this thesis we will adopt the term "deep" if the ANN has more than one hidden layer.

Theorem 2.2.1 (Chen, 2007 Theorem 3.2)**.** Let $\hat{\theta}_n$ be the approximate sieve M-estimator defined by (2.7) and (2.8), then under conditions 1-4 we get:

$$||\theta - \hat{\theta}_n|| = \mathcal{O}(\max\{\delta_n, ||\theta_0 - \pi_n\theta_0||\})$$

The final convergence of the estimator then depends on the type of sieve used. At this point it is important to mention a comment in Chen and Shen, 1998, who argue that there is no kind of sieve that trumps all others in terms of convergence, because it depends on the parameter space of (the true parameter) θ_0. As this thesis treats sieves based on ANNs, we focus on the findings of Chen, Racine, and Swanson, 2001 who consider ANNs with sigmoid activation functions and summarise their findings in the following proposition:

Proposition 2.2.2 (Chen, Racine, and Swanson, 2001 Proposition 1)**.** Assume conditions 1-4 are fulfilled. Then, for conditional mean or quantile regression for a β-mixing time series under assumptions that $\mathbb{E}[\varepsilon x_t] = 0$ and $\mathbb{E}[\varepsilon^4 x_t] < \infty$, single hidden layer ANNs with smooth sigmoid activations functions have the following upper bound on their rate of convergence:

$$||\hat{\theta}_n - \theta_0|| = \mathcal{O}\left([n/\log(n)]^{\frac{-(1+\frac{2}{d+1})}{4(1+\frac{1}{1+d})}}\right)$$

The authors propose the same rate of convergence under slightly different assumptions for ANNs with gaussian radial basis activations and ridgelet based ANNs. It can also be shown, that under general regularity conditions, sieve extremum estimation will consistently estimate both finite and infinite dimensional parameters. Because we will focus on the rate of convergence rather than the consistency, we will skip a thorough discussion and instead refer the interested reader to Chen, 2007 chapter 3.1. Next, we will investigate properties of the activation functions.

2.2.3 Choice of the Activation Function and Properties

As stated in section 2.2, the basis function needs to satisfy the condition that a linear combination can approximate any function in the desired function space arbitrarily well as r_n increases. We already established the universal approximation property for the most common activations in theorem 2.1.1, but did not elaborate on the specific choice. In the financial literature, ANNs are mostly used with sigmoid functions (Andersen et al., 2016), which corresponds to the logistic function as a special case. Researchers often omit the discussion of the activation function, because almost all activation functions fulfil the approximation condition. Most of the research during the 1990s involved single hidden layer ANNs and frequently employed a sigmoid activation function (see eg. Kuan and White, 1994).

More recently though, and probably due to the easier access to computing power, some empirical applications of neural networks in the economics and finance literature have employed deeper ANNs than the discussed single hidden layer ANN. Some applications had considerable empirical success (see eg. Gu, Kelly, and Xiu, 2018 or Sirignano, Sadhwani, and Giesecke, 2016). Yet, deep networks complicate the estimation process. An issue that arises frequently in deep networks is the *vanishing gradient problem*. As discussed in section 2.1, modern deep learning frameworks employ gradient based fitting techniques. When considering the algorithm in appendix A.1, one can see that it also depends on the gradient of the activation function. To illustrate the issue, suppose an ANN with only one hidden unit per layer, but 3 hidden layers. For a single input, we can then write the gradient problem as:

$$\begin{aligned}\frac{\partial L_t(\theta)}{\partial \gamma_{1,1}} &= \frac{\partial}{\partial \gamma_{1,1}} \frac{1}{2}\left(y_i - \beta_0 + \beta_1\left(G_3\Big(\gamma_{1,3}G_2\big(\gamma_{1,2}G_1(\gamma_{1,1}x_{1,i} + \gamma_{0,1}) + \gamma_{0,2}\big) + \gamma_{0,3}\Big)\right)\right)^2 \\ &\overset{\text{using } \pi_i \equiv G_j'(x)}{=} \beta_1\pi_3 \cdot \gamma_{1,3}\pi_2 \cdot \gamma_{1,2}\pi_1 \cdot x_1 \cdot (y_i - f(x_i, \theta))(-1)\end{aligned}$$

When analysing the derivative of the standard logistic function which is $\text{logistic}(x)(1-\text{logistic}(x))$ and has a maximum of 0.25 at $x = 0$, the issue becomes clear. The derivative contains a term $\pi \leq 0.25$ with the number of hidden layers as exponent, which lets the gradient decay exponentially towards zero. The effect could be mitigated with suitable weights for $\beta_1, \gamma_{1,j}, j \neq 1$, but this might lead to the opposite case of *exploding gradient* case. As a remedy, the ReLU function discussed in section 2.1.3 can be used, as its derivative is either 0 or 1. In deep learning applications the ReLU transformation has therefore become the standard activation function.

2.2.4 Multiple Layers

Until now, our analysis has only considered single hidden layer ANNs. From a theoretical point of view this will also stay our main focus, but we want to introduce the reader to an (at least heuristically) interesting property of multilayer (deep) ANNs. We run an experiment where we consider two ANNs. One, with 100 sigmoid units in the hidden layer and a second with 100 units in the first layer and 10 ReLU units in the second layer. This *increases* the number of free parameters from 301 to 1221. From the standard notion of bias-variance, we would therefore expect the bias to decrease and variance to increase. To make the comparison complete, we also fit a single layer model with 1228 free parameters.[12]

[12] Which corresponds to a single hidden layer ANN with 408 hidden units.

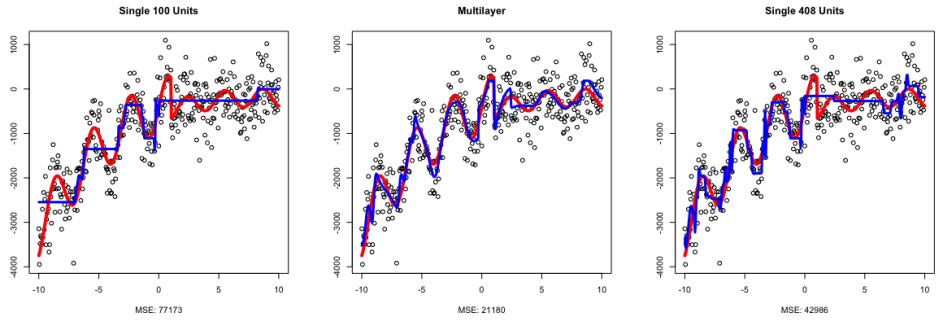

FIGURE 2.5: Data was generated from model B.2. The larger models were allowed more training epochs and all networks have sigmoid activations. Refer to appendix B.5.1 for the corresponding loss plots.

Figure 2.5 visualizes the results of this experiment. Because we did not use any regularization, all models start to overfit. Yet, the MSE of the multilayer network is considerably lower than that of the other two and visually seems to overfit the data less. A possible explanation can be found when analysing the output of each layer separately. Assuming we have a single input x_t and following our notation from above, we get (with k units in the first layer and k' units in the second):

$$x_t \xrightarrow[\text{k-units}]{\text{First Layer}} (g_1(x_t), ..., g_k(x_t)) \xrightarrow[\text{k'-units}]{\text{Second Layer}} \big(g_{2,1}(g_1(x_t) + \cdots + g_k(x_t)), \\ ..., g_{2,k'}(g_1(x_t) + \cdots + g_k(x_t))\big)$$

Looking at this from a matrix perspective for the output of the hidden layers

$$\xrightarrow[\text{k-units}]{\text{First Layer}} \underbrace{\begin{bmatrix} g_1(x_1) & g_2(x_1) & \dots \\ \vdots & \ddots & \\ g_1(x_n) & & g_k(x_n) \end{bmatrix}}_{\mathcal{G}_1;(n\times k)} \xrightarrow[\text{k'-units}]{\text{Second Layer}} \underbrace{\begin{bmatrix} g_{1'}(\cdot) & g_{2'}(\cdot) & \dots \\ \vdots & \ddots & \\ g_{1'}(\cdot) & & g_{k'}(\cdot) \end{bmatrix}}_{\mathcal{G}_2;(n\times k')}$$

For the sake of simplicity, assume that the activation functions in the second hidden layer are the identity (ie. $g_{k'}(\cdot) = \gamma_{2,k',1}g_1(x_t) + \cdots + \gamma_{2,k',k}g_k(x_t)$ where $\gamma_{2,k',j}$ are the weights of unit k' in the second hidden layer). If we have $k' < k$ the algorithm will then produce the linear combination of the $(n \times k)$ matrix, that preserves as much information as possible but fits in the smaller $(n \times k')$ matrix. A natural way to think of this is as a dimensionality reduction problem. Because we set the second activation function to the identity, this problem will be linear. We could achieve this using singular value decomposition (SVD) for the matrix $\mathcal{G}_1$ and selecting the k' left hand side eigenvectors corresponding to the k' largest eigenvalues. We can then reformulate the problem as:

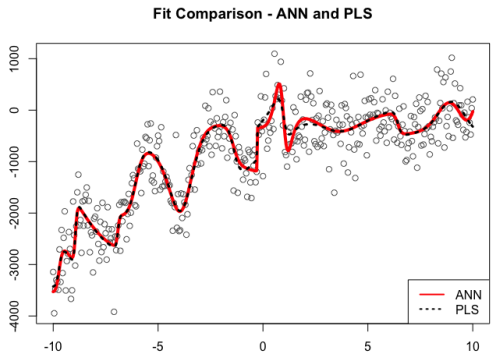

FIGURE 2.6: Comparison of the fit of a two hidden layer ANN with an identity activation function in the second hidden layer. The PLS model was calculated from the output of first hidden layer and the ten largest weighted principal components, the same number as hidden units in the second layer. Data was generated from model B.2

$$\mathcal{G}_1(x) \overset{SVD}{=} U_{k'} D_{k'} V'_{k'} + E_{\mathcal{G}_1(x)}, \quad \mathcal{G}_2(x) = U_{k'} D_{k'}$$

$$y_t = \mathcal{G}_2(x_t)\beta + u_t \ ,$$

where $U_{k'}$ corresponds to the k' principal left eigenvectors (and analogous for $D_{k'}$ and the eigenvalues $V_{k'}$). $E_{\mathcal{G}_1(x)}$ corresponds to the residual information. However, such an approach would ignore the relationship between $\mathcal{G}_1$ and y. A better way to consider this would be as a *partial least squares* regression (PLS) problem. PLS first reduces the input matrix using a "supervised SVD" which is achieved iteratively by taking the SVD of $X'y$ and deflating the resulting crossproduct by their left and right singular vectors (see eg. Mevik and Wehrens, 2007 for details). The resulting matrix $X_{\text{red.}}$ is then used for a linear regression of y on $X_{\text{red.}}$, before the estimated $\hat{\beta}$ is projected back into the space of the original data. This has the advantage, that it solves the optimization problem in a lower dimension and can hence reduce overfitting. In our case, we do not consider the original data as the X matrix, but its nonlinear transformations from the first hidden layer.

This means that we would project the k dimensional data from the first hidden layer to k' and then finding the best linear combination for the target y. Figure 2.6 provides visual evidence for a similarity between the two procedures. The PLS model was calculated with the output of the first hidden layer of a two hidden layer ANN. The first layer was estimated using sigmoid activations, the second used a linear activation to make the comparison to the PLS approach straightforward.[13] We will analyse this property more extensively in section 3 but leave out a thorough theoretical discussion as it would go beyond the scope of this thesis.

[13]Because with a nonlinear activation function, we would not be restricted to finding the best *linear* combination.

2.3 A Semiparametric Model

Semiparametric approaches contain both a parametric and nonparametric part. They have become popular because they usually provide parametric estimates that can be estimated at a $\sqrt{n}$ rate and also reduce the input dimension in the nonparametric part, which in turn alleviates the curse of dimensionality.

We have seen in the last sections, that ANNs provide us with a simple way to model data nonparametrically. This section develops an estimator for semi-nonparametric models, that is capable of capturing both linear and nonlinear relationships. We will take into account the properties of different ANN sieves seen in the last sections and provide a framework for inference. Finally, we will compare the theoretical convergence of ANNs to kernel-type nonparametric estimation.

2.3.1 Partially Linear Model

We will follow the motivating example from section 1.1 and assume a time-series partially linear form of the true process. We further assume that the data is stationary and β-mixing as defined in 2.2.2. Such a model can then be interpreted as a parametric model with time-varying intercepts driven by the nonparametric component (Ghysels and Marcellino, 2018). This specific form allows us to give a meaning to the nonparametric part of the equation as well. The formulation is:

$$Y_t = \mathbb{E}[Y_t|M_t] + e_t = m(M_t) + e_t = x_t'\beta + \phi(z_t) + e_t \quad t = 1, ..., n \tag{2.10}$$

where we partitioned the vector M_t into its linear and nonlinear components (x_t and z_t respectively). Restrictions can easily be imposed on the model. A popular choice is to impose additivity on the z_t terms in nonparametric component, in order to reduce the dimensionality of the problem (ie. set $\phi(z_t) = \phi(z_{1,t}) + \phi(z_{2,t}), ..., \phi(z_{k,t}))$. The nonparametric part of the equation then becomes the generalized additive model (Hastie and Tibshirani, 1990). The issue with such a restriction is that they require the model to be correctly specified in the case of (nonlinear) interactions between the input variables.

2.3.2 Augmented ANN

We have established in the previous section, that ANN-sieves can nonparametrically estimate β-mixing time series. To include a parametric part, we can slightly modify our ANN from equation (2.3) to obtain the *augmented* ANN (cf. Kuan and White, 1994):

$$f(x, \theta) = x'\alpha + \beta_0 + \sum_{j=1}^{r_n} \beta_j G(\tilde{x}'\gamma_j) \tag{2.11}$$

Note that the θ vector now contains the vector α as well. Further, note that we did not impose a special form on the parametric part. This allows us to include parametric nonlinear models as special cases. The model also shows a further advantage of using sieve estimators. Because we treat the nonparametric component as a parametric extension of an existing model, we can include it easily, even if the existing model is nonlinear (eg. in a logistic GLM - which could be used in a latent variable model).

We have seen in the previous section that multilayer networks might have desirable properties, which is why we propose to model the partially linear model from (2.10) with a multilayer ANN-sieve as the nonparametric component:

$$\phi_n(z_t;\theta) = \vartheta_0 + \sum_{r=1}^{k'} \vartheta_r G\big(\sum_{j=0}^{k} \beta_j G(\tilde{z}_t'\gamma_j) + \iota_r\big) \;, \tag{2.12}$$

where ϑ_r represents the weights from the second hidden layer and we set $G(\tilde{x}'\gamma_0) = 1$. The vector θ now contains $(d+1)k + (k'(k+1)) + (k'+1)$ parameters. The parameter d represents the dimension of z_t. The parametric part can then be expressed by a simple ARIMA model. The combined model then becomes:

$$Y_t = x_t'\zeta + \vartheta_0 + \sum_{r=1}^{k'} \vartheta_r G\big(\sum_{j=0}^{k} \beta_j G(\tilde{z}_t'\gamma_j) + \iota_r\big) + e_t \tag{2.13}$$

Note that we did not include a constant in the parametric vector ζ. Because the functional form of $\phi(z_t)$ is not specified, the model can incorporate the constant in the nonparametric part, which would make a parametric identification impossible (Li and Racine, 2006).[14]

This model is extendable to even more than two hidden layers if it is desired. We have seen that multilayer ANNs with sigmoid type activation functions suffer from the vanishing gradient problem. Therefore, we propose the use of the ReLU function as nonlinear transformation. We will henceforth call the proposed model the SANN (semi-nonparametric Artificial Neural Network). Figure 2.7 depicts the SANN in the network architecture as already seen before. The subsequent sections discuss the convergence properties of the proposed estimator.

2.3.3 Estimation and Convergence of the Parametric Part

First we will discuss the convergence of the parametric part of the regression equation in (2.13). We propose the use of a two-step procedure to estimate the model because it has several advantages. To begin with, it makes it easier to derive properties for the inference

[14] A closer inspection of (2.13) reveals that the constant will be absorbed by the bias unit in the final layer of the ANN.

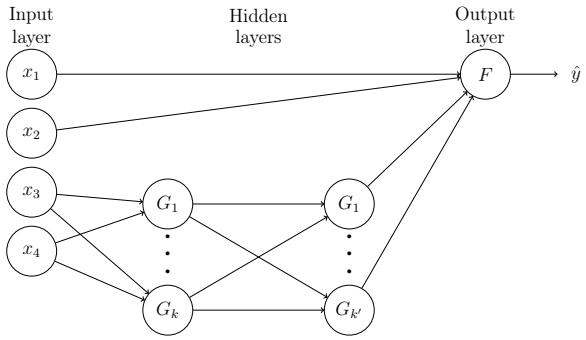

FIGURE 2.7: Representation of a semi-(non)parametric augmented ANN with two hidden layers and two input variables each. Note that we are not restricted to separate the linear and nonlinear inputs.

on the parametric part because we can use existing theory, and in addition, it makes use of the specific form of our ANN sieve. Note that we will follow the notation of (2.10) to keep the expressions concise (hence the derived β corresponds to ζ in equation (2.13)).

Although we have a linear and additive structure in the output layer, the preceding hidden layer ensures that the nonparametric part is nevertheless approximated well by the ANN terms.[15] The idea is then to isolate the parametric component and estimate it with standard procedures. The simplest form to isolate β would be by taking conditional expectations and differencing out from model 2.10:

$$Y_t - \mathbb{E}[Y_t|z_t] = (x_t - \mathbb{E}[x_t|z_t])'\beta + (e_t - \mathbb{E}[e_t|z_t])$$

Then we define:

$$\tilde{Y_t} = Y_t - \mathbb{E}[Y_t|z_t], \quad \tilde{x_t} = x_t - \mathbb{E}[x_t|z_t]$$

Which allows us to provide the (matrix) least squares solution as.

$$\hat{\beta}_{\text{infeasible}} = (\tilde{X}'\tilde{X})^{-1}\tilde{X}'\tilde{Y} \ , \tag{2.14}$$

where we used the standard matrix notation. We call this the infeasible estimator, as it would require the unknown quantities $\mathbb{E}[Y_t|z_t]$ and $\mathbb{E}[x_t|z_t]$. Because we can estimate the nonparametric part with our ANN, we are nonetheless able to derive a consistent estimate $\hat{\mathbb{E}}$. The specific form of the final layer makes the isolation of the parametric part convenient. By using partitioned regression (see appendix A.3 for details) we are able to obtain the estimator for the parametric part:

$$\hat{\beta} = (\tilde{X}^{*'}\tilde{X}^{*})\tilde{X}^{*'}\tilde{Y}^{*} \tag{2.15}$$

[15]The next subsection will treat the convergence of the nonparametric part. For the moment, just suppose the estimates in the output layer are sufficiently close to the true values.

$$\tilde{X}^* = X - \hat{\mathbb{E}}[X|Z], \quad \tilde{Y}^* = Y - \hat{\mathbb{E}}[Y|Z]$$

Further, if we impose the condition, that the convergence rate of the nonparametric part is *slower* than $\sqrt{n}$. Then under some fairly general restrictions, it can be shown, that the parametric part of a sieve M estimate is asymptotically normal and converging at the rate $\sqrt{n}$. As we have already established, that our ANN sieve is an M-estimator and that we estimate β with a two step procedure, we refer the interested reader to Chen, 2007 theorem 4.1 and only state the result:

Theorem 2.3.1 (Asymptotic normality for semiparametric two-step estimation)**.**

$$\sqrt{n}(\hat{\beta} - \beta_0) \xrightarrow{\text{dist}} \mathcal{N}(0, \Omega)$$

$$\Omega = (\Gamma_1' W \Gamma_1)^{-1} \Gamma_1' W V_1 W \Gamma_1 (\Gamma_1' W \Gamma_1)^{-1} \,,$$

where Γ_1 is the partial derivative of a function $M(\beta, \phi_0) = 0$ iff $\beta = \beta_0$ and W a weighting matrix such that a sieve estimation M_n weighted with W (ie $M_n(\beta, \phi_0)' W M_n(\beta, \phi_0)$) is close to $M(\beta, \phi_0)' W M(\beta, \phi_0)$. See Chen, 2007 for details. We also note that this is similar to the asymptotic distribution of standard series estimators (cf. Li and Racine, 2006, Ch 15). The fact that the parametric part of our model converges to a normal distribution allows us to employ statistical inference on the estimated coefficients.

2.3.4 Convergence of the Nonparametric Part

What is left is the discussion of the convergence of the nonparametric of the SANN. For simplicity we will derive the upper bound for the single hidden layer ReLU ANN rather than the proposed multilayer network. This has two reasons. First, it allows us to directly compare the rates to the other ANN sieves that were discussed by eg. Chen, Racine, and Swanson, 2001. And second, a closer examination of the derivation in appendix A.4 reveals that the variance term δ_n from theorem 2.2.2 still increases with additional parameters (even in new hidden layers), without taking into account any potential benefit from dimensionality reduction. This is due to the nature of the proof, where we derive a general *upper bound* on the convergence. Nevertheless, the first point made above already merits a thorough discussion of the convergence rate of ReLU sieves.

To get our convergence results, we first need to impose some technical assumptions (analogous to the assumptions imposed by Chen, Racine, and Swanson, 2001):

Assumption 2.3.1 (Nature of the time series process)**.** The considered time-series is stationary and β-mixing with $b_\tau \leq b_0 \tau^{-\xi}, \xi > 2$

Assumption 2.3.2 (Relationship between the parametric and nonparametric part)**.** Δx does not enter $\phi(z)$ additively and has full column rank. z_t has compact support χ on $\mathbb{R}^d$

Assumption 2.3.3 (Bounds on weights)**.** We impose a limit on the weights of the β coefficients in the ANN in equation (2.13):

$$\sum_{j=0}^{r_n} |\beta_j| \leq c_n, \quad \sum_{i=0}^{p} |\gamma_{i,j}| \leq c_{n_2},$$

$$\sum_{i=0}^{p} |\gamma_{ij} x_i| \leq c_l, \implies \sum_{j=0}^{r_n} \beta_j \text{ReLU}(\gamma_{ij} x_i) \leq r_n c_n c_l$$

Which ensures that our the function only maps to values within $\mathbb{R}$ and not $\bar{\mathbb{R}}$. Following Hornik et al., 1994 and Chen, 2007 we define a possible function space for ϕ in equation (2.10): Suppose that the true $\phi_0 \in \mathcal{P} \equiv \left\{\phi \in L_2(\chi) : \int_{\mathbb{R}^d} |\omega||\tilde{\phi}(\omega)|d\omega < \infty\right\}$, where $\tilde{\phi}$ is the Fourier transform of ϕ. That is, $\phi \in \mathcal{P}$ iff. it is square integrable and $\tilde{\phi}$ has finite first moments.

Proposition 2.3.1 (Rate of convergence for ReLU-ANN sieves)**.** Let $\hat{\theta}_n$ be the sieve M-estimate described by equation (2.6), $z_t \in \mathbb{R}^d$ and $\phi \in \mathcal{P}$. Suppose that assumptions 2.3.1-2.3.3 hold. By letting the sieve order grow according to $r_n^{2(1+\frac{1}{d+1})} \log(r_n) = \mathcal{O}(n)$ we achieve the following convergence rate for an ANN a ReLU activation function:

$$||\hat{\theta}_n - \theta_0|| = \mathcal{O}\left([n/\log(n)]^{\frac{-(1+\frac{2}{d+1})}{4(1+\frac{1}{1+d})}} \right)$$

When following through the proof in appendix A.4, note the following: Given assumption 2.3.3, we can express the sieve order for multilayer networks with k units in the first layer and k' units in the second layer as $r_n = k * k'$ and the implication in the assumption would stay the same. In section 3.2 we will discuss how the heuristic findings from section 2.2.4 can still be possible although the derived bound does not treat single or multilayer networks differently. Also note that this rate is *slower* than $\sqrt{n}$, which ensures the convergence to a normal distribution for the parametric part.

We then see that the proposed rate is the same as was achieved by eg. Chen, Racine, and Swanson, 2001 for ANNs with activation functions that suffer from the vanishing gradient problem (cf. proposition 2.2.2). Given our findings from section 2.2.4, this motivates us to use the ReLU activation for our SANN. The established convergence results also allow us to compare the asymptotic performance of our ANN-sieve to that of other popular nonparametric estimators. For example, we take the optimal convergence for a kernel estimator (see eg. Hansen, 2009):

$$\text{AMISE}_{\text{opt}} = \mathcal{O}(n^{\frac{-2\text{order}}{(2\text{order}+\text{dimension})}})$$

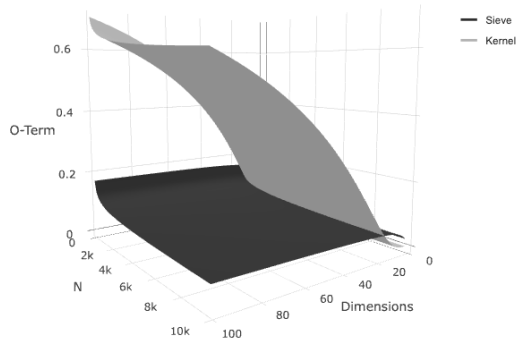

FIGURE 2.8: Comparison of the optimal rate of convergence between multiple second order kernel- and sieve-type Estimators (for a single regressand).

When ignoring the constant, it becomes clear that in higher dimensional settings, the sieve estimator discussed above might have better convergence properties. Figure 2.8 provides a graphical illustration of the optimal rate of convergence for the two type of estimators (where we assumed a second order kernel). The constant M of our earlier definition of $\mathcal{O}$ is *not* taken into account in this visualisation (although in the limit as $n \to \infty$ they will both converge to 0 no matter the value of the constant). Therefore, we cannot compare the rates directly for finite samples. The illustration serves only to show that sieve estimators are less sensitive on the input dimension than kernel type estimators. To take into account the constant, a finite sample study of the convergence property will be discussed in section 3.3.

2.3.5 Estimation

There are some aspects of estimation worth mentioning. If we were not interested in the parameters of our model, we can use a joint estimation. This is frequently done in the literature. For example, the highly cited article of Zhang, 2003 used an ARIMA model on the data and re-modelled the residuals with an ANN. A closer inspection of the estimation process of Kuan and White, 1994 reveals that they are actually estimating the same with an augmented ANN of the form of (2.11), but less explicitly than in the former paper[16]. The parametric structure of the ANN sieve also provides us with the possibility to regularize the network parameters with eg. L_1-penalization.

Further, regularization alleviates another problem for the inference of the parametric part. Because our convergence only provides us with an upper bound on the number of sieve terms for our theoretical considerations, we often need to provide a value for r_n. L_1-regularization will effectively set some weights and some hidden nodes to zero. Instead of

[16]They estimate their model with a two step approach, where they force the β parameters of the augmented ANN from (2.11) to be 0 in the first stage and then optimize them in a second step without changing the α from the first step

providing the network with the (unknown) correct number of sieve terms, we can simply provide a sufficiently large number of hidden units and let the algorithm choose the optimal non-zero number via eg. cross validation. This is often computationally less demanding than other procedures employed for finding the optimal terms for series estimation (such as the *jackknive* of eg. Hansen, 2014). Further, because we used the estimate $\hat{\phi}_n$ to eliminate the non-parametric component in our inference framework in section 2.3.3 we need a data-driven way to find r_n. If we just choose some random number as our sieve order, we cannot employ standard inference as our model would be (at least partially) random and misspecified. L_1-regularization also mitigates that issue.

2.4 Extension to the Multivariate Case

The extension to the multivariate case is straightforward. Because the ANN is not restricted to a single output and every input is passed on the subsequent layer, we can easily extend our model (2.13) to the multivariate case. For example, we can express the structure of the partially linear model (2.10) as a semiparametric vector autoregressive model with covariates, which we call SVARX. Suppose we have Y_t, a q-dimensional time series and Z_t an s-dimensional time series of covariates or lagged values. We can then specify a VAR-model that is augmented by a nonparametric transformation of observable covariates. The model can then be represented as:

$$Y_t = \boldsymbol{B_1} Y_{t-1} + \cdots + \boldsymbol{B_p} Y_{t-p} + \boldsymbol{\phi}(Z_t) + u_t \quad t = 1, ..., t \tag{2.16}$$

where p is the lag order that we select in the VAR model and

$$Y_t = \begin{bmatrix} y_{t,1} \\ \vdots \\ y_{t,q} \end{bmatrix}, \boldsymbol{B_i} = \begin{bmatrix} \beta_{1,1,i} & \dots & \beta_{1,q,i} \\ \vdots & \ddots & \\ \beta_{q,1,i} & & \beta_{q,q,i} \end{bmatrix}, i = 1, ..., p, \quad \boldsymbol{\phi} = \begin{bmatrix} \phi_1(Z_t) \\ \vdots \\ \phi_q(Z_t) \end{bmatrix}, u_t = \begin{bmatrix} u_{1,t} \\ \vdots \\ u_{1,q} \end{bmatrix}$$

Note that we are not restricted to the VAR case for the linear part because the ANN-sieve is very flexible. We presented this specific case because it is widely used and because we can easily transfer some established properties from the univariate case. Although, the (nonlinear) parameters themselves do not have a specific interpretation, the system as a whole can be analysed with a generalized impulse response function (analogous to most standard VAR applications).

3. Simulation

In the previous chapter, several theoretic aspects for the proposed sieve estimator were discussed. The present section will analyse them in detail and illustrates finite-sample performance and inference properties. For that, we will first investigate general properties of the nonparametric part of our proposed estimator, and then analyse the capabilities of the SANN (2.13) as a whole.

3.1 Setup

The simulations are carried out in the statistical software R with the following packages for the corresponding models (if not otherwise specified, we will use the `base` implementation for linear models):

ANNs and SANNs [`keras`]
The simulations involving an ANN were done using the keras package (with a tensorflow backend), which is an R wrapper around the python version. For ANN-type estimations, we chose enough epochs (passes of the dataset to the gradient descent algorithm) such that the loss levelled off. Appendix B.5.1 and B.5.2 contain the corresponding plots for the first simulations. The optimization algorithm, number of hidden units, batch size and the regularization parameter were always chosen to be the same when comparing models. Precise model specifications are summarised in appendix B.5.

Nonparametric Regression [`np`]
For the nonparametric regression, we chose the local linear (LL) model with a second order gaussian kernel. Its theoretical convergence was briefly discussed in the previous section, its *local* estimate has the form:

$$\beta^{LL}(x_0) = (X'W_0X)^{-1}X'W_0Y \ ,$$

where $W_{0,i,i} = K(x_i, x_0, h)$ is the diagonal weighting matrix from the density estimates. When the `np` package is used, a bandwidth parameter is needed. We chose Silvermans adapted rule of thumb. For the univariate data it can be expressed as (eg. Härdle et al., 2004):

$$\text{bw}_{\text{silverman}} = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{\frac{1}{5}} \ ,$$

where $\hat{\sigma}$ is the sample standard deviation. The `np` package calculates a slightly modified version for multivariate data: $\text{bw}_{\text{rule of thumb}} \approx 1.06\sigma_j n^{\frac{1}{2P+l}}$. Where the σ_j is calculated for every variable as $\min\{\text{std.}, \text{mae.}/1.4826, \text{iqr.}/1.346\}$, P is the order of the kernel and l the number of variables. We opted for the rule of thumb because it usually provides a good approximation to the optimal bandwidth and because other methods such as cross-validation (CV) often have impractically long calculation times for large datasets.

Partially Linear Model [`PLRModels`]
The package fits partially linear models with a kernel-type estimator for the nonparametric part. The package can only use a single variable as nonparametric component by default and does not provide a framework to implement out-of-sample predictions.

MGARCH Models [`rmgarch`]
The package allows to fit different MGARCH models to the data. It is useful for the last part of the simulations where we investigate the capabilities of the multivariate version of the SANN. We did not cover MGARCH processes as it would go beyond the scope and goal of this thesis. The interested reader is referred to eg. Martin, Hurn, and Harris, 2012 for details. We will use the Dynamic Conditional Correlation (DCC) model as a baseline, because it is widely used in practice but suffers from restrictions on the multivariate distribution of its components.

For the simulations themselves, we specify the following setup: For each experiment, we will use $B = 50$ MC iterations. Consider the general case of equation (2.4):

$$Y_t = \psi(x_t) + u_t, \quad t = 1, ..., n$$

The considered $\psi(\cdot)$ can be either linear, partially linear or completely nonlinear. We then analyse the specific features of our ANN-sieve and the proposed model in equation (2.13). For each simulation, we fix the DGP $\psi(\cdot)$ and leave it unchanged throughout the B iterations. The noise term u_t is drawn from a distribution and *changed* for every iteration $b \in \{1, ..., B\}$. Next, the relevant statistical metrics are calculated, which will change according to the specific issue analysed. Refer to appendix B.1 for details on those metrics. In general, we use pointwise metrics (calculated on the specific data-points) if we want to compare the performance of different estimators and integrated metrics when we compare the approximation properties. The integrated metrics are calculated by numerically integrating the pointwise metrics as suggested in Chen, 2007.

3.2 Single- and Multilayer Performance

We start our simulation study with an investigation into the properties seen in section 2.2.4. We compare the bias-variance decomposition of a single-layer and a multi-layer model for a fixed number of hidden units in the (first) hidden layer. For that, we simulate data according to model B.2. To ensure consistency with the earlier results from section 2.2.4, we use a sigmoid ANN with 250 hidden units and a corresponding ANN with an additional five identity units in the second layer. Because we are interested in the specific properties of multilayer networks, we did *not* use regularization for either network. The results are presented in table 3.1.

	Bias^2	Var_e	MSE
Multilayer	$1,745$	$38,466$	$40,211$
Singlelayer	$10,160$	$39,026$	$49,186$

TABLE 3.1: Bias-Variance decomposition for single- and multilayer ANN. MSE is the sum of Bias^2 and Var_e. The total number of free parameters for the single layer network were 751, for the multilayer 1761.

Both models were trained until the loss criterion was levelled off (refer to appendix B.5.1 for the loss plots). We see that for the multilayer ANN the squared bias decreases significantly, and the variance of the estimator also decreases slightly. Whereas the decrease in bias would be expected with an increasing number of parameters, the decrease in the estimator variance comes counter-intuitive. We have already explored a possible explanation for this in section 2.2.4.

The findings from this small simulation provide further evidence that multilayer networks might be preferable to single hidden layer ANNs in certain settings. We also note a limitation of the use of multiple layers. With increasing training time, the units in the first layer can just "explode" to achieve the same result as a single layer even though the second layer acts as dimensionality reduction. Still, the multilayer architecture reduces the impact on the output that each unit in the first hidden layer as and thus still prevents overfitting, as visible in the loss plots in the appendix (the multilayer network was trained until well *after* its loss levelled off). Further, by allowing more flexibility in the first layer, we can decrease the bias and stop the gradient descent algorithm earlier. This in turn, as already discussed in appendix A.1, reduces the variance of the estimation. From a theoretical view, we can also reconcile these findings with the structure of derivation of the convergence rate. Because even with a limit on the magnitude of the model parameters, we still use an upper bound of the metric entropy $\mathcal{H}_{[]}$, which grows with the sieve order. Hence a gain on the variance from the model architecture could still be possible, as $\mathcal{H}_{[]}$ defines an upper bound but not necessarily a supremum.

3.3 High Dimensional Estimation

The next property that we consider is the performance of ANN-sieves in a higher dimensional settings. We saw in section 2.3.4 that the $\mathcal{O}(\cdot)$ term for the ANN sieve is *less* responsive to an increasing number of input parameters than that of kernel type estimators. From figure 2.8 we can visually infer that in settings where the input dimension is small, the kernel-type estimator seems to have a faster convergence. This advantage diminishes as the input dimension increases, which would be in line with the curse of dimensionality. We investigate the performance of the two estimators by generating data according to:

$$y_i = \sum_{i=1}^{k} f_k(X_i) + \varepsilon_i, \quad i = 1, ..., 500, \ k = 1, ..., 15$$

$$\varepsilon_i \sim \mathcal{N}(0, 7^2) \ ,$$

where f_k are different nonlinear functions. Refer to table B.1 for details. We then let the number of inputs that generate the true function increase (2,5,15) and for each generated function we add 1,10 and 20 noise terms. We estimate a single hidden layer ANN with 200 ReLU units and a local linear regression (kernel-type) on the data. To avoid having to choose the optimal number of sieve terms r_n, we use L_1-regularization in the hidden layer. For this and all subsequent simulations we only optimize the regularization parameters once and leave them unchanged throughout the simulations. We split the generated data into a 80/20 training-test set and calculate the RMSPE on the test data. The results are summarized in table 3.2.

		Relevant Predictors		
	Noise	2	5	15
Nonparametric (LL)	1	6.11*	18.31*	59.13
	10	28.98	46.48	79.30
	20	44.02	66.65	97.99
ANN	1	22.33	33.99	41.86*
	10	27.10*	36.85*	47.03*
	20	34.87*	43.27*	51.72*

TABLE 3.2: Comparison of the RMSPE across different estimators and data generating processes. The * denotes the best performing (in terms of RMSPE) model for the corresponding problem. For the first column of the ANN simulation we increased the learning rate slightly to make sure the loss levelled off. We also allowed for more training epochs for larger models. The loss functions can be found in appendix B.5.2

As expected, the kernel-type estimation performs really well when the function is generated by few terms and when few noise-terms are present. As the number of relevant predictors increases the ANN starts to perform (relatively) better. The lower sensitivity of the ANN against higher dimensions can also be seen with the increasing noise terms, where the RMSPE increase is relatively lower for the ANN.

3.4 Inference and Prediction Accuracy

One of the goals of this thesis is to provide guidelines for the inference from ANNs. By using the two step estimation procedure for the parametric part that we developed in section 2.3.3, we are able to obtain standard errors and a convergence to a distribution for our parametric estimates which is not possible in standard ANN estimation.

We will consider data generated from two different time series settings. One with and one without temporal autoregression. For ease of interpretation and calculation, we will consider lower-dimensional series. We saw in the preceding section 3.3 that a kernel-type estimator would probably provide more accurate results for such a problem, nevertheless, the present section allows us to illustrate the theoretical findings. Further, to the best of our knowledge, no general inference framework was yet proposed for semi-nonparametric ANN sieve estimators. The present section provides motivation for a development of such theory. We note that we allow all models the correct number of lags, as the focus of the analysis is not model selection but rather model comparison.

The first simulation is essentially a replication of example 11.11 in Martin, Hurn, and Harris, 2012, who use it to illustrate the properties of a kernel estimator. We consider DGP of the form:

$$y_t = 2x_{1,t} + x_{2,t} + g(x_{3,t}) + u_t, \quad t = 1, ..., 1250,$$

$$g(x_{3,t}) = 0.3\exp(-4(x_{3,t}+1)^2) + 0.7\exp(-16(x_{3,t}-1)^2),$$

$$u_t \sim \mathcal{N}(0, 0.1), \quad x_{1,t}|x_{3,t} \sim \mathcal{N}(0.5x_{3,t}, 1), \quad x_{2,t} \sim \text{student}_{df=4}, \quad x_{3,t} \sim \text{Unif}[-2, 2]$$

The parameters are then estimated with our proposed SANN from equation (2.13), a kernel-based partially linear model, a benchmark linear model of the form $y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \beta_3 x_{3,t}$ and a correctly specified model. To avoid having to choose the right sieve order for the SANN, we use L_1-regularization in the first hidden layer of the ANN. The parameter estimates are then calculated according to the procedure in section 2.3.3. The prediction results, based on the RMSPE and the integrated bias and variance for the nonparametric part are reported in table 3.3 (Model 1) and the nonparametric approximation is visualized in figure 3.1. From the results we can see that the kernel-type estimator outperforms the nonparametric approximation of the SANN, which is to be

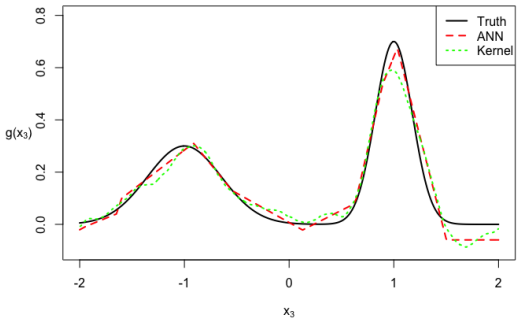

FIGURE 3.1: Nonparametric function approximation for x_3 in Model 1.

expected in such a low-dimensional problem. Nevertheless, the predictive performance of the SANN looks promising.

Next, we want to analyse the performance of the estimator in an autoregressive process time-series setting. For that, we consider the nonlinear multiple time series (which is closely related to the experiment of Gao and Tong, 2004):

$$y_t = \beta_1 v_{t-1} - \beta_2 v_{t-2} + \left(\frac{x_{t-1} + x_{t-2}}{1 + x_{t-1}^2 + x_{t-2}^2}\right)^2 + \varepsilon_t, \quad t = 1, ..., 1250$$

$$\beta_1 = 0.47, \beta_2 = -0.45$$

$$v_t = 0.55 v_{t-1} - 0.42 v_{t-1} + \delta_t, \quad x_t = 0.8 \sin(2\pi x_{t-1}) - 0.2 \cos(2\pi x_{t-2}) + \eta_t$$

$$\varepsilon_t \sim \mathcal{N}(0, 0.5^2), \quad \delta_t, \eta_t \sim \text{Unif}[-0.5, 0.5]$$

We then run four different models on the data. The benchmark linear model, a hybrid ANN model, a fully nonparametric ANN model and the correctly specified model. Again, we use L_1-regularization to shrink the sieve order r_n to the optimal level. Results are summarized in table 3.3 (Model 2). We also present the distribution of β_1 and its standard errors in figure 3.2 (β_2 yielded similar results). Because the two series v_t and x_t are not correlated, the OLS prediction should provide unbiased and consistent estimates. We can infer that the SANN also provides accurate parameter estimates in addition to the increased predictive performance. Further, we can see that the standard errors from the SANN are overall lower than for the OLS estimation. This probably arises due to the efficiency gain from removing the nonparametric variance before estimating the parameters.

Our simulation results also confirm what other empirical studies for time series have found before. The simple ANN provides worse results than the baseline linear estimate, even though we used regularization. On the other hand, our SANN which used the same

	Model 1	Model 2	Model 2a	$\int \text{Bias}^2$	$\int \text{Var}_e$
True	0.3174	0.4944			
Linear	0.3647	0.5199			
ANN	0.3434	0.5495			
SANN	0.3206	0.5061	0.5110	0.0012	0.0072
Kernel				0.0012	0.0061

TABLE 3.3: Performance statistics for semi-nonparametric models, based on the RMSPE. For Model 1 we also present the integrated squared bias and integrated variance of the SANN and Kernel-partially linear model. The performance is not reported for the Kernel-PLM because the package does not support predictions.

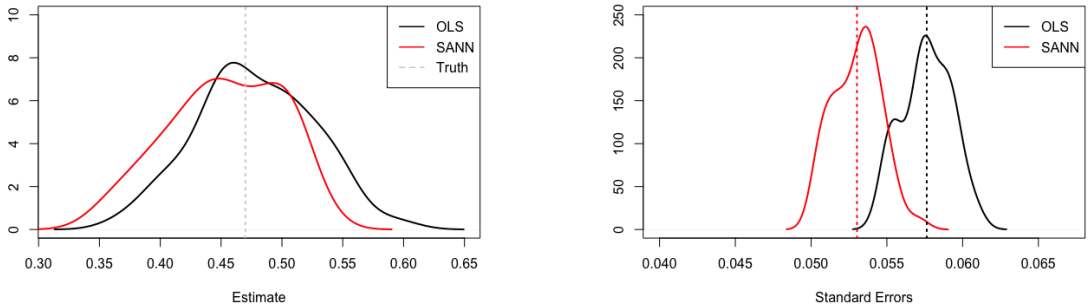

FIGURE 3.2: Distribution of estimates of the two parametric components in the autoregressive time series simulation. The dotted lines in the right pane denote the respective means of the distribution.

amount of regularization and free parameters, improved our forecasting result. The improvement is not as large as in the previous experiment, which is probably due to the lower SNT ratio (also compare this to the correctly specified model, which only provides a 10.7% improvement on the baseline linear model). Of course, it seems reasonable that a model which has the correct division of the parametric and nonparametric part, will outperform a purely linear or nonparametric model. Nevertheless, the results are motivating for the further use. We also estimated our SANN by providing all inputs to the linear and nonlinear part. Interestingly, the model still outperforms the baseline linear and ANN models (Model 2a) but, unsurprisingly, the performance is lower than that of the correctly specified SANN.

3.5 Multivariate Case

In this last section of simulations, we will consider an extension to the multivariate case. We have already seen that ANN-sieves can provide better results than standard linear

techniques, and that especially in higher dimensional settings ANN-sieves outperform kernel type estimation. In this section we will analyse whether these findings can heuristically be carried over to the multivariate case.

A possible application where such high dimensions arise, is in settings where dynamic variances and covariances are estimated. Commonly, multivariate generalized autoregressive conditional heteroskedasticity (MGARCH) models are used to model such processes. We follow Andersen et al., 2016 and define the process as:

$$y_t = \boldsymbol{H}_t^{1/2}\eta_t$$

where y_t is the $N \times 1$ vector of mean zero returns, $\boldsymbol{H}_t$ is the conditional variance covariance matrix and η_t an iid random vector with $\mathbb{E}[\eta_t] = 0$ and $\mathbb{E}[\eta_t\eta_t'] = I_N$. Misspecification in parametric models arises due to the assumptions imposed on η_t (Andersen et al., 2016) and hence y_t. Commonly it is assumed that y_t is multivariate normal, but this assumption is often violated, especially with financial data. Although standard parametric models are able to model for example financial returns with leptokurtic distributions, they are often not able to model all of it (Martin, Hurn, and Harris, 2012).

If the distributional assumption is wrong, the maximum likelihood will only be a quasi-maximum likelihood for MGARCH models and there is an efficiency loss in finite samples. Kernel-density estimates can be used to correct such misspecifications, but they suffer from the curse of dimensionality. We want to investigate how the SANN performs in such a setting. To adapt the SANN for this multivariate variance-covariance problem, we propose a slightly changed version of the proposed SVARX from equation (2.16) and transform it into a semi-parametric diagonal VECH model:

$$\text{vech}(H_t) = \text{diag}(B)\text{vech}(y_{t-1}y_{t-1}') + \phi(\text{vech}(y_{t-1}y_{t-1}')) \ ,$$

where $\text{vech}(\cdot)$ is the column stacking operator for the (co)-variances of the lower triangular argument matrix, and B the parametric $(N(N+1)/2 \times N(N+1)/2)$ weights matrix. The VECH model comes with many disadvantages, for example, the positive definiteness of H_t is not guaranteed with such a model. We still opt for the model because it is easy to implement and we can consider the results as a lower benchmark for the SANN performance. Because the ANN does not need a specific error distribution for its estimation, we can relax such conditions.[1] We consider two cases, the first follows closely Rossi and Spazzini, 2010 and specifies an autoregressive process for the variance and a

[1]The above estimation can also be generalized to a multivariate GARCH-type estimation, for the current section we refrain from such a model, as they are (in the used framework) computationally extremely demanding and will be used in the next section. Further, one can show (see B.2) that the GARCH model is merely a parsimonious representation for an ARCH_∞ specification.

dynamic correlation matrix with:

$$\sigma_{i,t}^2 = 0.3 + 0.55\varepsilon_{t-1}^2 + u_t, \quad t = 1, ..., 1100 \quad \text{(Model 1)}$$

$$\boldsymbol{H}_t = \Omega_t R_t \Omega_t, \quad R_{i,j} = 0.5 + 0.2 * \cos\big(\frac{2\pi t}{1000}\big) \forall i \neq j, \text{else } 1 \ ,$$

where Ω_t is diagonal and contains the square root of the volatilities $\sigma_{i,t}^2$ and R_t the time varying correlation matrix. We let $\eta_{i,t}$ for the $i = 1, ..., 10$ variables be generated by a multivariate normal distribution and a skewed generalized t-distribution. The correlation matrices for both processes are redrawn for each iteration. For additional details see appendix B.4.4. The M-ARCH model assumes conditional multivariate normality of the data but is otherwise correctly specified. Finally, to ease the computational burden, we separated the SANN and ANN estimation of the variance and covariance. We then calculate the RMSPE on a hold out set consisting of the last 100 observations. The results are reported in table 3.4. The plain ANN is not able to capture enough structure and performs worst in terms of RMSPE for both versions of Model 1. Further, the M-ARCH model performs better than the SANN when the distributional assumption is satisfied, but the SANN performs relatively better when that assumption is violated. Figure 3.3 illustrates the effect. Although the DCC estimator picks up the dynamics of the process, it provides worse results than the SANN for larger shocks that violate the normality assumption.

	Model 1			Model 2
$\boldsymbol{\eta_{i,t}}$	Normal Variance	Skew-t Variance	Skew-t Covariance	Normal Variance
M-ARCH	0.193	0.151	0.220	0.341
SANN	0.216	0.139	0.204	0.183
ANN	0.939	0.408	0.867	0.195

TABLE 3.4: RMSPE for the multivariate models. For Model 2, the forecasts were additionally corrected in mean with the last 100 observations of the test set.

The second case that we investigate is multivariate misspecification of the functional form of the conditional variance. We simulate the data according to:

$$\sigma_{i,t}^2 = 0.1 + \big(\frac{0.7\varepsilon_{t-1}^2 + 0.2\varepsilon_{t-2}^2}{1 + \varepsilon_{t-1}^2 + \varepsilon_{t-2}^2}\big) + u_t, \quad t = 1, ..., 1100 \quad \text{(Model 2)}$$

Which is the multivariate extension of the autoregressive time-series setting in section 3.4, slightly adapted to avoid numerical issues in the simulation process and u_t can be considered an error process. For Model 2 we assume that $\eta_{i,t}$ follows a multivariate

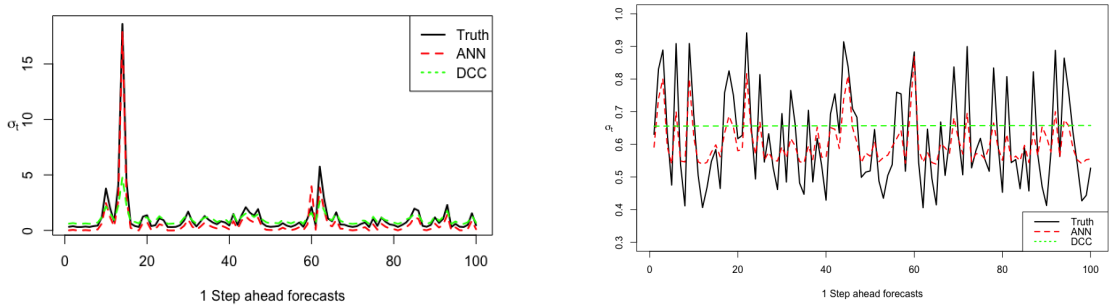

FIGURE 3.3: Out of sample one-step-ahead forecasts of Model 1 with a distributional misspecification in the MGARCH (left pane) and Model 2 (right pane).

normal distribution. We estimate the variance with the same models as above. Results are summarized in table 3.4. Because the model is completely nonlinear, the M-ARCH model does not pick up any dynamics and its forecast collapses to the unconditional variance. The effect can also be seen visually in figure 3.3. As no linear relation is present, the SANNs weights for the parametric part are mostly zero but it still performs slightly better than a plain ANN (which is probably due to the fact that we only optimized the regularization parameters of both models once). We infer from the results that the SANN is able to profit from structure if it is correctly imposed (Model 1) but only suffers a marginal efficiency loss in the case that the parametric part is misspecified.

We conclude this section with a brief summary of our three key findings. First, in higher dimensional settings, the ANN-sieve outperforms standard kernel regression. Second, when correctly specified, the SANN provides competitive results for function approximation and increases the efficiency of the parametric estimates compared to the linear baseline. Third, SANNs are also applicable in multivariate settings, and increase the model performance when the error distribution or the functional form of a variance-covariance model is misspecified.

4. Application to Financial Data

We began the introduction with the argument that economic and financial time series often seem to have nonlinear components. Further, we have found that ANNs seem to perform better than other nonparametric estimators in higher dimensional settings. Portfolio modelling is a case where such high dimensional settings arise naturally, as an effective diversification often involves many assets.

An important measure in statistical risk management is the Value at Risk (VaR), which exists for single assets or portfolios. For a confidence level $\alpha \in (0, 1)$ at horizon h,

the Value at Risk ($\text{VaR}_{\alpha,h}$) is defined as the smallest value p such that the probability of a loss (L) smaller than p is at least $(1-\alpha)$. The definition corresponds to the $(1-\alpha)$ quantile of the loss distribution at time t for the horizon h:

$$\text{VaR}_{h,\alpha} = \inf\{p : \mathbb{P}(L_{t,t+h} \leq p) \geq (1-\alpha)\}$$

For our application, we will set the forecast horizon h to 1, h is therefore omitted from further notation. The VaR_α is, for example, used to partially determine the reserves that a financial institution is required to hold. This opens up two aspects that are important from the modelling perspective. A well constructed estimate should reflect the risk of a loss greater than VaR_α accurately, but at the same time the estimate should also be as low as possible to avoid excessive reserves.

4.1 Semiparametric CAViaR

A variety of estimation procedures exist for the estimation of the VaR. A specially interesting basic estimate is the CAViaR model from Engle and Manganelli, 2004. Instead of modelling the whole distribution of the return (and hence the loss) and then estimating the $(1-\alpha)$-quantile, the CAViaR approach attempts to model the VaR directly as an autoregressive function. The rationale behind this is the empirical finding that variances are autoregressive conditionally heteroskedastic. The VaR which is linked to the standard deviation of returns should hence also be conditionally linked to its previous observations. To specify a CAViaR model, let y_t be a vector of returns, α the probability as described above, x_t a vector of observable variables such as the lagged variance and $f_t(\theta) \equiv f_t(x_{t-1}, \theta_\alpha)$ the α-quantile of the distribution of portfolio returns at time t, parametrized by θ. Following our notation from the last section, we propose to model the VaR as:

$$f_t(\theta) = \sum_{j=1}^{p} \beta_j f_{t-j}(\theta) + \phi(x_{t-1}) \;, \tag{4.1}$$

where $\phi(x_t)$ is a nonparametric function of lagged values, such as the variance, in the information set at $t-1$. Here, besides the predictive performance, we are interested in how much information can be drawn from past VaR observations. A significant coefficient would suggest that clustering of volatilities even has effects in the tails of the distribution (Engle and Manganelli, 2004).

The model is simply an augmented version of the original CAViaR-GARCH model proposed in the original paper. With our considerations from section 2.3, we can consider the VaR as an autoregressive process driven by past VaR, with a time-varying intercept

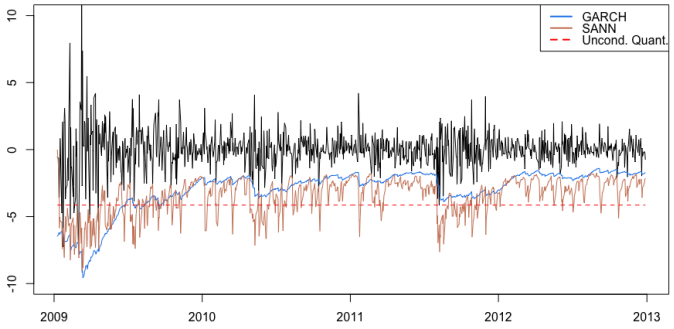

FIGURE 4.1: Out of sample prediction of GARCH(2,1) and SANN(2,1)

driven by an unknown process of the lagged variance. Because the aim of this section is the illustration of possible applications for the proposed estimator and not the discussion of statistical properties of the CAViaR model itself, we do not further engage in a theoretical discussion and refer the interested reader to Engle and Manganelli, 2004.

Because we are modelling a quantile, we need to adapt the optimization problem from equation (2.6) to:

$$\hat{\theta} = \arg\min_{\theta \in \Theta_n} \frac{1}{T} \sum_{t=1}^{T} [\alpha - \mathbb{1}(y_t < f_t(\theta))][y_t - f_t(\theta)],$$

where α is the quantile as defined before. Also note that we are no longer in a standard feed forward case, because the autoregressive VaR_α-term is itself an estimation and needs to be fed back into the model. The autoregressive term is initialized as the unconditional α-quantile of the training data. We let the lagged quantile enter linearly into our model, because even in such a case, the lagged value will contain the nonlinear transformations of the lagged variables in x_{t-1}.

The data considered is the daily closing price of the GE-stock from 01.01.2000 up to 31.12.2012, for a total of 3265 observations. We leave out the last 1000 observations for out of sample forecasting (corresponding to 08.01.2009-31.12.2012). We deliberately include the stress period of the great recession and the European debt crisis in the test set, because we expect stress periods to have return distributions which are not in line with the normality assumption. We estimate the VaR_α for $\alpha = 0.01$, because the lower quantiles are more difficult to model due to the scarceness of data in this range. We estimate the model in equation (4.1) for $p = \{0, 1\}$ and allow for up to two autoregressive lags. We then compare the performance of our SANN to that of a standard GARCH(2,1) model.

The forecasts are then evaluated with two statistical tests based on Kupiec, 1995 and

Christoffersen and Pelletier, 2004. First we check whether the number of exceedances of the VaR is in line with the expected number (which for $\alpha = 0.01$ is 10). Further, if the model is correctly specified, the exceedances should be independent (otherwise there is still information left that could be incorporated into the model). A correctly specified model would fail to reject the null, that the model is correctly specified, in both cases. To assess the economic perspective, we numerically integrate the VaR estimates over the forecasting period. We then report the percentage change of the value compared to the integrated value of the unconditional α-quantile over the same period. Ideally, a good model would decrease the integrated VaR vis-a-vis the unconditional quantile. Finally, we also report the estimate and standard errors for the autoregressive VaR_α-term to check whether conditional heteroskedasticity is also relevant in the tails. The results are summarised in table 4.1.

	GARCH(2,1)	SANN(2,0)	SANN(2,1)
Exceedances	14	11	12
Failures Test	0.23	0.75	0.54
Duration Test	0.15	0.01	0.17
Change $\int \text{VaR}_\alpha$	-27.7%	-13.8%	-15.8%
Coef. VaR_{t-1}			0.66
Std. Error			0.19

TABLE 4.1: Results of univariate $\text{VaR}_{0.01}$ forecasting models. For the failures and duration tests the p-values are reported. The change towards $\int \text{VaR}_\alpha$ is compared to the integrated unconditional quantile. The VaR_{t-1} coefficient and standard error are from the in-sample estimation.

The Failure test is not rejected for all models, suggesting that they provide an acceptable failure rate. The SANN(2,0) model rejects the duration test, suggesting there is still residual information in the data. Indeed, the SANN(2,1) which increases the dependence horizon, fails to reject the duration test and also further decreases the integrated VaR. The parameter estimate and the significance of the VaR_{t-1} term are also in line with what was found in Engle and Manganelli, 2004. This is further evidence that conditional heteroskedasticity is also relevant in the tails of a distribution. From the statistical evidence, we cannot conclude that the SANN provides a significant improvement over a standard GARCH model in the univariate case. Figure 4.1 depicts the out of sample forecasts and the true returns. It appears that the SANN forecast picks up similar patterns as the GARCH forecast, but is a lot more responsive to movements in the return series.

4.2 Multivariate Semiparametric GARCH

We have seen in section 3.5 that the proposed model also generalizes (at least heuristically) to the multivariate case. In the second part of this application, we will consider the VaR of a portfolio instead of a single stock. In the previous section it was also shown that the SANN can also be used to model conditional covariances nonparametrically and does well when modelling nonlinear relationships or non-normal distributions. To test the performance, we construct 50 randomly weighted portfolios of six assets, containing stocks, commodities and currencies. The portfolio construction and the assets are closer described in appendix B.6.

	MGARCH(1,1)	SANN(2,0)
Exceedances	23.02	10.56
Failures Test	0.52	0.02
Duration Test	0.12	0.06
Change $\int$ VaR$_\alpha$	-25.0%	-6.1%

TABLE 4.2: Results of VaR$_{0.01}$ out of sample forecasts on the portfolio level. The statistical tests were calculated for the 0.95 level and the figures represent the average rejection rate over the 50 constructed portfolios.

We compare the performance across the portfolios and summarize the results in table 4.2. Contrary to the univariate case, the MGARCH model fails to provide accurate VaR$_{0.01}$ forecasts for the portfolios in over half of the cases. The SANN on the other hand provides very accurate results for the portfolio VaR. Both models do not appear to have significant improvement possibilities. Although the MGARCH model fails the duration test for around 12% of the portfolios, we could not improve the results by allowing for more lags. The SANN seems correctly specified, which is why we refrain from estimating an SANN with a lagged VaR coefficient for the portfolio case. Similar to the univariate case, the average decrease in the integrated VaR compared to the unconditional baseline is smaller for the SANN than for the MGARCH. But this might just reflect the true risk better, especially in turbulent markets, where the MGARCH model frequently underestimated the VaR.

5. Concluding remarks

We began the thesis with an argument for more accurate and better understood forecasting methods. Our goal was to provide an overview on how Artificial Neural Networks can

be incorporated into a statistical framework and under what circumstances they might provide better results than other existing estimation methods.

To start with, we discussed the structure of ANNs and their universal approximation property. We then illustrated how ANNs can be incorporated into the framework of sieve estimation based on Chen, 2007. The sieve framework allowed us to derive convergence properties and explain the bias-variance tradeoff of ANNs with statistical, rather than heuristic arguments. Further, the sieve framework also explains why ANNs can be considered as nonparametric estimators. The theoretic considerations then permitted us to propose a semi-nonparametric model based on ANNs (SANN). We extended theoretical results for ANN-sieves of Chen, Racine, and Swanson, 2001 to include multi-layer ANNs with ReLU activation functions, to provide a starting point for incorporating modern deep learning. Further, given that ANNs represent nonparametric estimators, we were able to draw insights from theory developed by Li and Racine, 2006 for general semi- and nonparametric estimation techniques. This allowed us to propose a method that yields standard errors for the parametric part in a semi-nonparametric ANN-model and hence enables inference on the parameters. To the best of our knowledge, this not been explicitly done before.

With extensive monte-carlo simulations we provided evidence that ANN-based nonparametric estimates tend to converges better in higher dimensions than other nonparametric methods, such as kernel regression. Furthermore, inference on the parametric part of the SANN is also possible and provided promising results. Finally, the proposed SANN estimation performed better in out of sample forecasts than both the standard linear model or a fully nonparametric ANN by themselves in a variety of settings. These findings also held when we applied the model to real stock market data to estimate the Value at Risk of a portfolio.

In the statistical analysis of ANNs, much remains to be done. For example, our multivariate application, we used a simple and joint estimation for the parametric and nonparametric part. But this approach could be extended to include richer parametric structures and error corrections. Since ANNs seem to perform better than other nonparametric techniques in higher dimensional settings, they are particularly attractive for portfolio modelling. Future research can focus on the many potential scenarios have not yet been explored, for example how ANNs behave in the presence of cointegration. Moreover, long-term dependencies arise frequently in financial or economic settings. Throughout this thesis we only used the simplest form of ANNs, but there exists a variety of ANNs that allow for recurrence relations in the data and could be further investigated from a statistical perspective.

Bibliography

Andersen, TG et al. (2016). *Handbook of Financial Time Series*. Springer Publishing Company, Incorporated.

Chen, X (2007). "Large Sample Sieve Estimation of Semi-Nonparametric Models". In: *Handbook of Econometrics*. Ed. by J.J. Heckman and E.E. Leamer. 1st ed. Vol. 6B. Elsevier. Chap. 76.

Chen, X, J Racine, and NR Swanson (2001). "Semiparametric ARX neural-network models with an application to forecasting inflation". In: *IEEE Transactions on Neural Networks* 12.4, pp. 674–683.

Chen, X and X Shen (1998). "Sieve Extremum Estimates for Weakly Dependent Data". In: *Econometrica* 66.2, pp. 289–314.

Chen, X and H White (1999). "Improved rates and asymptotic normality for nonparametric neural network estimators". In: *IEEE Transactions on Information Theory* 45.2, pp. 682–691.

Cheng, B and DM. Titterington (1994). "Neural Networks: A Review from a Statistical Perspective". In: *Statistical Science* 9.1, 2–30.

Christoffersen, P and D Pelletier (2004). "Backtesting Value-at-Risk: A Duration-Based Approach". In: *Journal of Financial Econometrics* 2.1, pp. 84–108.

Engle, RF and S Manganelli (2004). "CAViaR: Conditional Autoregressive Value at Risk by Regression Quantiles". In: *Journal of Business & Economic Statistics* 22.4, pp. 367–381.

Franses, HP and D van Dijk (2000). *Non-linear time series models in empirical finance*. Cambridge University Press.

Gao, J and H Tong (2004). "Semiparametric Non-Linear Time Series Model Selection". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 66.2, pp. 321–336.

Geman, S, E Bienenstock, and R Doursat (1992). "Neural Networks and the Bias/Variance Dilemma". In: *Neural Computation* 4.1, pp. 1–58.

Ghysels, E and M Marcellino (2018). *Applied Economic Forecasting using Time Series Methods*. Oxford University Press.

Goodfellow, I, Y Bengio, and A Courville (2016). *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press.

Greene, WH (2018). *Econometric Analysis*. eighth. Pearson Education.

Grenander, U (1981). *Abstract inference*. English. Wiley New York.

Gu, S, B Kelly, and D Xiu (2018). *Empirical Asset Pricing via Machine Learning*. Working Paper. National Bureau of Economic Research.

Hansen, BE (2009). *Lecture Notes on Nonparametrics*. `https://www.ssc.wisc.edu/~bhansen/718/NonParametrics1.pdf`. Accessed: 2019-03-07.

— (2014). "Nonparametric Sieve Regression: Least Squares, Averaging Least Squares, and Cross-Validation". In: *The Oxford Handbook of Applied Nonparametric and Semiparametric Econometrics and Statistics*. Ed. by Racine JS, L Su, and A Ullah. Oxford University Press. Chap. 8, pp. 215–248.

Härdle, WK et al. (2004). *Nonparametric and Semiparametric Models*. Springer Series in Statistics. Springer Berlin Heidelberg.

Hastie, T and R Tibshirani (1990). *Generalized additive models*. Wiley Online Library.

Hastie, T., R. Tibshirani, and J.H. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

Hornik, K, M Stinchcombe, and H White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359 –366.

Hornik, K et al. (1994). "Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives". In: *Neural Computation* 6.6, pp. 1262–1275.

Härdle, W, H Liang, and J Gao (2000). *Partially Linear Models*. Physica-Verlag Heidelberg.

Huk, M (2012). "Backpropagation generalized delta rule for the selective attention Sigma-if artificial neural network". In: *Applied Mathematics and Computer Science* 22.2, pp. 449–459.

Kuan, C-M and H White (1994). "Artificial neural networks: an econometric perspective". In: *Econometric Reviews* 13.1, pp. 1–91.

Kupiec, P (1995). "Techniques for Verifying the Accuracy of Risk Measurement Models". In: *The Journal of Derivatives* 3.2, pp. 73–84.

Leshno, M et al. (1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural Networks* 6.6, pp. 861 –867.

Li, Q and JS Racine (2006). *Nonparametric Econometrics: Theory and Practice*. Economics Books 8355. Princeton University Press.

M4 forecasting competition. `https://eng.uber.com/m4-forecasting-competition/`. Accessed: 2019-01-03.

Martin, V, S Hurn, and D Harris (2012). *Econometric Modelling with Time Series: Specification, Estimation and Testing*. Themes in Modern Econometrics. Cambridge University Press.

Mevik, BH and R Wehrens (2007). "The pls Package: Principal Component and Partial Least Squares Regression in R". In: *Journal of Statistical Software, Articles* 18.2, pp. 1–23.

Newey, WK (1997). "Convergence rates and asymptotic normality for series estimators". In: *Journal of Econometrics* 79.1, pp. 147 –168.

Ossiander, M (July 1987). "A Central Limit Theorem Under Metric Entropy with L_2 Bracketing". In: *The Annals of Probability* 15.3, pp. 897–919.

Rossi, E and F Spazzini (2010). "Model and distribution uncertainty in multivariate GARCH estimation: A Monte Carlo analysis". In: *Computational Statistics Data Analysis* 54.11, pp. 2786 –2800.

Shen, X and WH Wong (1994). "Convergence Rate of Sieve Estimates". In: *Ann. Statist.* 22.2, pp. 580–615.

Sirignano, J, A Sadhwani, and K Giesecke (2016). "Deep Learning for Mortgage Risk". In: *arXiv e-prints*.

Solow, RM (2010). "Building a Science of Economics for the Real World". `https://www.govinfo.gov/content/pkg/CHRG-111hhrg57604/pdf/CHRG-111hhrg57604.pdf`. Subcommittee on Investigations and Oversight, Accessed: 2019-14-01.

Stinchcombe, M and H White (1990). "Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights". In: *1990 IJCNN International Joint Conference on Neural Networks*, 7–16 vol.3.

White, H (1990). "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings". In: *Neural Networks* 3.5, pp. 535 –549.

Yao, Y, L Rosasco, and A Caponnetto (2007). "On Early Stopping in Gradient Descent Learning". In: *Constructive Approximation* 26.2, pp. 289–315.

Zhang, GP (2003). "Time series forecasting using a hybrid ARIMA and neural network model". In: *Neurocomputing* 50, pp. 159–175.

Technical Appendix

A.1 Backpropagation Algorithm

Instead of just presenting the specific form of the algorithm for the equation 2.3, we will derive the general case and show the equivalence for our example. The notation is slightly simplified. We consider the observations $t = 1, ..., n$, $\phi_{j,l}$ the standard weighted aggregation function (from equation (2.1)) and $G_{j,l}$ an activation function, where j, l denotes the j^{th} hidden unit in the the l^{th} consecutive layer in the neural network. Note that the input layer is denoted with $l = 0$. We will follow Huk, 2012 and write the least square criterion in the case of a single output unit as:

$$L_t(\theta) = \frac{1}{2}(y_t - f(x_t, \theta))^2 \ ,$$

where $f(x_t, \theta)$ is the fitted value of the ANN for observation t, note that we added the $\frac{1}{2}$ for notational convenience, as it is a linear transformation it will not alter the result. We then define the total error created by hidden unit j, m as the negative partial derivative of the squared error function wrt. the hidden unit, ie:

$$e_{j,l} = -\frac{\partial L_t(\theta)}{\partial \phi_{j,l}}$$

Rewrite $e_{j,l}$:

$$e_{j,l} = -\frac{\partial L_t(\theta)}{\partial \phi_{j,l}} = -\frac{\partial L_t(\theta)}{\partial G_{j,l}(\phi_{j,l})} \frac{\partial G_{j,l}(\phi_{j,l})}{\partial \phi_{j,l}} = -\frac{\partial L_t(\theta)}{\partial G_{j,l}(\phi_{j,l})} G'(\phi_{j,l}) \ .$$

For the output layer, we assumed the liner (identity) activation function (in which case $G_{j,l}(\cdot) = F(\cdot)$) and it in our specific example, it has only one output unit. It is easy to see then, that $e_{t,\text{output unit}} = -(y_t - f(x_t, \theta))$. For the other layers in the network, the first term of the RHS must be examined closer. Because the output from a hidden unit will be used in all units from the subsequent layers, the error induced will have further

effects on subsequent functions. Using the chain rule we obtain:

$$\frac{\partial L_t(\theta)}{\partial G_{j,l}(\phi_{j,l})} = \sum_{q=1}^{m_{l+1}} \frac{\partial L_t(\theta)}{\partial \phi_{q,l+1}} \frac{\partial \phi_{q,l+1}}{\partial G_{j,l}(\phi_j)}$$

Where m_{l+1} denotes the number of hidden units in the subsequent layer. Note that the reason for the name "backpropagation" originates in this formula. To calculate the value of the loss function $L_t(\theta)$, the errors must be calculated through the whole network (passed forward). As the equation shows, the derivatives of the functions must then be passed back along the network hidden unit to calculate the gradient and update the weights as is shown below. The gradient descent algorithm, then updates the parameter estimates $\hat{\theta}^r$ with the following recursion:

$$\hat{\theta}^{(r+1)} := \hat{\theta}^r + \lambda I_n \nabla L(\hat{\theta}^r) \,, \tag{A.1}$$

where r denotes the r^{th} iteration and λ is a parameter that regulates the step sizes of the descent algorithm and referred to as the *learning rate*.[1]

Up until now we have only calculated the error that was induced by the hidden unit itself, which does not yet enable us to calculate the specific weights that are needed to update the optimization function. To get the formula for the individual weights i in each hidden unit from the hidden layers let $\omega_{j,i,l}$ denote value of the i^{th} parameter in the j^{th} unit in the l^{th} layer of the network. For the simple single hidden layer network in equation (2.2) this would correspond to the weights $\gamma_{i,j}$ and β_i which we summarised in θ. We can use the formula:

$$\frac{\partial L_t(\theta)}{\partial \omega_{j,i,l}} = \frac{\partial L_t(\theta)}{\partial \phi_{j,l}} \frac{\partial \phi_{j,l}}{\partial \omega_{j,i,l}}$$

Where we can see that $\frac{\partial \phi_{j,l}}{\partial \omega_{j,i,l}}$ is equal to the output from the i^{th} neuron in the preceding layer. Furthermore, we can rewrite

$$\frac{\partial L_t(\theta)}{\partial \phi_{j,l}} = \frac{\partial L_t(\theta)}{\partial G_{j,l}(\phi_{j,l})} \frac{\partial G_{j,l}(\phi_{j,l})}{\partial \phi_{j,l}} = -e_{j,l}$$

Summarized for equation 2.3 we get:

$$\begin{aligned} \frac{\partial L_t(\theta)}{\partial \beta_k} &= e_{1,2} \frac{\partial \phi_{1,2}}{\partial \omega_{1,2,k}} \\ &= (y_t - f(x_t, \theta)) F'(\phi_k) G(x_t, \beta_k) \\ &= (y_t - f(x_t, \theta)) G(x_t, \beta_k) \end{aligned} \tag{A.2}$$

[1] We will discuss the choice of the learning rate further as it will not directly be needed throughout this thesis. The interested reader is instead referred to eg. Goodfellow, Bengio, and Courville, 2016.

and

$$\begin{aligned}\frac{\partial L_t(\theta)}{\partial \gamma_{i,k}} &= e_j \frac{\partial \phi_q}{\partial \omega_j} \\ &= G'(x_t'\gamma_j)\frac{\partial L_t(\theta)}{\partial \phi_\xi}\frac{\partial \phi_\xi}{\partial G_j(\phi_j)}x_{t,i} \\ &= G'(\phi_k)(y_t - f(x_t,\theta))\beta_j x_{t,i}\end{aligned} \tag{A.3}$$

Note that we only optimized over one observation t. This illustrates the use of the algorithm for massive datasets as not the whole dataset must be used to find an optimum. Commonly only a batch of the data is used in the estimation process, and given the algorithm this can be highly parallelized on modern multi-core machines, which helps to explain the recent surge in popularity of neural networks as Cheng and Titterington, 1994 already pointed out years earlier.

The derivation above allows us to illustrate how penalization and early stopping can prevent overfitting. We will discuss the simplest case of L_2-regularization on the network parameter vector θ, because for that specific case an analytic solution exists. Assuming a regularization parameter $\alpha \in [0, 1)$ that is added to the objective function for the weights of the hidden layer l, we modify the gradient descent iteration from equation (A.1) and obtain (see eg. Goodfellow, Bengio, and Courville, 2016):

$$\hat{\theta}^{(r+1)} := (1 - \alpha\lambda)\hat{\theta}^r + \lambda I_n \nabla L(\hat{\theta}^r)$$

Which leads to parameter shrinkage of the parameter vector in each iteration. The authors then go on and show that if the loss function is the MSE, the regularized solution will rescale the weights along the axes defined by the Hessian matrix of the objective function wrt. the network weights, evaluated at the value $\theta^* = \arg\min_\theta L(\theta)$. Specifically, the shrinkage towards zero will be largest along the directions of the Hessian where the eigenvalues are smallest (ie. where the objective function does not increase much). L_1-Regularization has a similar effect, with the difference that it can shrink non-zero weights to exactly zero (whereas the L_2 solution only shrinks non-zero weights *towards* zero), which results in sparse solutions. Especially with the ReLU activation this has the effect of shrinking the whole output of the unit towards zero.

Early stopping has a slightly different mechanism. We will only sketch out the rationale, as the detailed solution is long and technically involved and refer to Yao, Rosasco, and Caponnetto, 2007 for further details. From our update criterion in equation (A.1), we see that it is based on the sample data (by plugging in the earlier defined loss function). Where the parameter vector $\hat{\theta}$ is the approximation to the true underlying function f_0.

Let $\hat{\theta}^r$ and f_r be the sample and population solutions to the recursion at iteration r. Yao, Rosasco, and Caponnetto, 2007 show that the population iteration f_r converges to the true value f_0 with increasing r but that $\hat{\theta}^r$ converges to an overfitting solution. They show a simple bias-variance tradeoff based on the number of iterations r by using the triangular inequality:

$$||\hat{\theta}^r - f_0|| \leq ||\hat{\theta}^r - f_r|| + ||f_r - f_0|| \,,$$

where the second term on the RHS will converge to zero with increasing r (and represents the bias) but the first term will increase (representing the variance). Early stopping can hence decrease the variance, but increases the bias of the estimation.

The reason why this shrinks the estimation towards a linear solution is the initialization of the network weights themselves. Usually, the weights are initialized randomly around zero. If we assume the network uses the logistic function as an activation with weights near zero, we can show that it is almost linear in parameters. For that consider some small $x > 0$ such that in the range of $[-x, x]$ the logistic function is almost linear, which means the slope is almost constant. We know that the derivative of the logistic function is symmetric and maximises at the origin at 0.25. We can then find suitable a x by restricting the slope to be almost constant. Choosing some $\varepsilon > 0$ for example $\varepsilon = 0.01$ we can write:

$$\frac{d}{dx}\text{logistic}(x) = \frac{\exp(x)}{(1+\exp(x))^2} > 0.25 - \varepsilon$$

$$\log(\frac{2}{3}) < x < \log(\frac{3}{2}) \,,$$

which means that the logistic function is approximately linear in x on the range $(\log(\frac{2}{3}), \log(\frac{3}{2}))$. By using $\log(\frac{2}{3}) = \log((\frac{3}{2})^{-1}) = -1\log(\frac{3}{2})$ we also know that the range is symmetric. By adding a weight ω to x we get:

$$\omega x < \log(\frac{3}{2}) \,,$$

which increases the range $\forall\omega \in (-1, 1)$. Because the standard least squares is scale-invariant, a small value ω will produce a solution which is approximately linear. Stopping the gradient descent algorithm early then results in a shrinkage towards that solution as suggested in Hastie, Tibshirani, and Friedman, 2009.

A.2 Derivation Bias-Variance Tradeoff

We will derive the bias-variance tradeoff for ANNs with the methodology of Hansen, 2014.

Proof of proposition 2.2.1. Without loss of generality we will derive the mean integrated squared error for the sieve order r_n and data set size n ($\text{MISE}_n(r_n)$). The generalisation for stationary time series is then straightforward. Define the MISE as:

$$MISE_n(x) = \mathbb{E}\int \big(\hat{\psi}_m(x) - \psi(x)\big)^2 dx$$

Contrary to the standard OLS case, we need to account for two sources of error: the approximation error and the regression error. From the definition of the approximation error in 2.9, we know that $\psi(x) = \upsilon_n + \mathcal{G}(x)\beta_n$. We can then rewrite the RHS of the equation as[2]:

$$\begin{aligned}\int \big(\hat{\psi}_n(x) - \psi(x)\big)^2 f(x)dx &= \int \Big(\mathcal{G}(x)\hat{\beta}_n - (\mathcal{G}(x)\beta_n + \upsilon_n(x))\Big)^2 \\ &= \int \upsilon_n(x)^2 f(x)dx - 2(\hat{\beta}_n - \beta_n)' \int \mathcal{G}(x)'\upsilon_n f(x)dx \\ &+ (\hat{\beta}_n - \beta_n)' \int \mathcal{G}(x)'\mathcal{G}(x)f(x)dx(\hat{\beta}_n - \beta_n)\end{aligned}$$

We then use $\mathbb{E}[\mathcal{G}(x)'\upsilon_n] = 0$ (from the linear regression assumption) to let the second term be equal to 0. We use the definition of φ_n^2 and set $\mathbb{E}[\mathcal{G}(x)'\mathcal{G}(x)] \equiv \mathcal{E}_n$. Further by the property of $\mathbb{E}[x] = \mathbb{E}[trace(x)]$ for x a scalar and the cyclical property of the trace we get:

$$IMSE_n(x) = \varphi_n^2 + trace\left(\mathcal{E}_n \mathbb{E}\Big[(\hat{\beta}_n - \beta_n)(\hat{\beta}_n - \beta_n)'\Big]\right)$$

Finally, define $\mathbb{E}[\mathcal{G}(x)'\mathcal{G}(x)\sigma_t^2] \equiv \Omega_n$ where $\sigma_t^2 = \mathbb{E}[u_t^2|x_t]$. Using the formula of the least squares estimate for $\hat{\beta}_n$ and the true model of the sieve estimate of order n, $y_t = \mathcal{G}(x_t)\beta_n + u_t$ we get the following (asymptotic) result:

$$\begin{aligned}\mathbb{E}\Big[(\hat{\beta}_n - \beta_n)(\hat{\beta}_n - \beta_n)'\Big] &= \mathbb{E}\Big[\big((\mathcal{G}(x)'\mathcal{G}(x))^{-1}(\mathcal{G}(x)'\mathcal{G}(x)\beta_n) + (\mathcal{G}(x)'\mathcal{G}(x))^{-1}(\mathcal{G}(x)'u) - \beta_n\big) \\ &\qquad \big((\mathcal{G}(x)'\mathcal{G}(x))^{-1}(\mathcal{G}(x)'\mathcal{G}(x)\beta_n) + (\mathcal{G}(x)'\mathcal{G}(x))^{-1}(\mathcal{G}(x)'u) - \beta_n\big)'\Big] \\ &= \mathbb{E}\Big[(\mathcal{G}(x)'\mathcal{G}(x))^{-1}\mathcal{G}(x)'uu'\mathcal{G}(x)(\mathcal{G}(x)'\mathcal{G}(x))^{-1}\Big] \\ &= n^{-1}\mathcal{E}_n^{-1}\Omega_n\mathcal{E}_n^{-1}\end{aligned}$$

[2]For the ease of notation we omit the time index t from the calculations. In this case assume that $\mathcal{G}(x)$ is the $(1 \times r_n)$ vector of transformed variables for x_t in the first hidden layer and u the corresponding noise term.

Note that we used the sample equivalent in the last step. Together this yields:

$$MISE_n(x) = \varphi_n^2 + n^{-1} trace(\mathcal{E}_n^{-1}\Omega_n)$$

Which is proposition 2.2.1 for sieve order r_n.

□

A.3 Parametric Part of a Partially Linear Model

Proof. Following our notation from above we denote $\mathcal{G}(Z)$ nonlinear transformed, $(n \times r_n)$ output matrix in the last layer and X the $(n \times k)$ linear inputs. The optimization problem in the output layer is then similar to the standard OLS problem and we can then write it the equation as:

$$A = \begin{bmatrix} X & \mathcal{G}(Z) \end{bmatrix}, \Lambda = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

$$Y = X\beta_1 + \mathcal{G}(Z)\beta_2 + \varepsilon = A\Lambda + \varepsilon \tag{A.4}$$

Given the OLS solution to equation A.4 we then transform the problem into the normal equations:

$$\hat{\Lambda}_{OLS} = (A'A)^{-1}A'Y \implies (A'A)\hat{\Lambda}_{OLS} = A'Y$$

Which by the above definition can be rewritten as:

$$\begin{bmatrix} X'X & X'\mathcal{G}(Z) \\ \mathcal{G}(Z)'X & \mathcal{G}(Z)'\mathcal{G}(Z) \end{bmatrix} \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} X'Y \\ \mathcal{G}(Z)'Y \end{bmatrix}$$

Note that the matrix A needs to be invertible, which means that both X and $\mathcal{G}(Z)$ need to have full column rank. ANNs (especially with ReLU activation functions) tend to have at least some columns equal to 0 in the final layer, therefore they need to be removed prior to the estimation.We can then follow Greene, 2018 theorem 3.2 to obtain the estimate for β_1[3]

$$\hat{\beta}_1 = (X'M_{\mathcal{G}}X)^{-1}(X'M_{\mathcal{G}}Y)$$

[3]In the case where X and $\mathcal{G}(Z)$ are orthogonal, we could directly regress Y on the corresponding matrices to get the estimate of β_1 and β_2. However, we treat the general case, where we would for example allow the ANN to contain all variables or the case where the sieve order was chosen without data driven methods.

Where $M_{\mathcal{G}}$ is the "residual maker matrix" that contains the residuals from a regression of Y on $\mathcal{G}(Z)$:

$$\begin{aligned} Y - \mathcal{G}(Z)\hat{\beta}_2 &= Y - \mathcal{G}(Z)(\mathcal{G}(Z)'\mathcal{G}(Z))^{-1}\mathcal{G}(Z)'Y \\ &= [I_n - \mathcal{G}(Z)(\mathcal{G}(Z)'\mathcal{G}(Z))^{-1}\mathcal{G}(Z)']Y \\ &= M_{\mathcal{G}}Y \end{aligned}$$

Last, note that with the theory developed about ANN-Sieves we know that $\mathcal{G}(Z)$ will converge to the true function of $g(Z)$ and hence we can calculate the approximation of $\tilde{Y} = Y - \mathbb{E}[X|Z]$ and $\tilde{X} = X - \mathbb{E}[X|Z]$. As we only approximate the true function g and hence $\tilde{X}$ and $\tilde{Y}$ we denote the approximations with $\tilde{X}^* = X - \hat{\mathbb{E}}[X|Z]$ and $\tilde{Y}^* = Y - \hat{\mathbb{E}}[Y|Z]$ respectively. Hence we get:

$$\hat{\beta} = (X'M_{\mathcal{G}}X)^{-1}(X'M_{\mathcal{G}}Y) = (\tilde{X}^{*'}\tilde{X}^*)^{-1}\tilde{X}^{*'}\tilde{X}^*$$

By using the fact that $M_{\mathcal{G}}$ is symmetric and idempotent.

□

A.4 Proof of Proposition 2.3.1

We follow a similar way as Chen, Racine, and Swanson, 2001 and Chen, 2007 who prove the proposition for ANNs with smooth sigmoid and gaussian radial basis activation functions.

Proof of proposition 2.3.1. First, in order to use existing results, we note that by Stinchcombe and White, 1990 the ReLU function can be made to comply with the assumption of compact support of the activation function. For the theoretical considerations it is sufficient to think about a constant $c_l > 0$ such that we consider $f(x) = \min\{\max\{0, x, \}, c_l\}$. Which can be achieved by restricting the weights $\gamma_{i,j}$ suitably (note that under assumption 2.3.3 and for the single hidden layer model this is guaranteed). Under these assumptions, the ReLU-ANN sieve is (in the limit) dense in $\mathcal{P}$ by theorem 2.1.1.

Throughout the proof assume that assumptions 2.3.1, 2.3.2 and 2.3.3 hold. This implies that condition 1 is directly satisfied.

Further, conditions 2 and 4 can be verified the same way as in Chen, 2007 example 3.2.2.

To obtain the deterministic approximation error rate, note that we restricted the parameters and the maximum of our ReLU function, which means that the Hölder assumption in Chen and White, 1999 is fulfilled. Further it is clear that the ReLU function

is *not* homogenous (ie. $\text{ReLU}(\lambda x) \neq \lambda \text{ReLU}(x)$ in general). Then, by theorem 2.1, and lemmas A1, A2 from Chen and White, 1999 and the adjustment to our problem we obtain the deterministic approximation error rate :

$$||\theta - \pi_n \theta|| \leq \text{constant} * r_n^{\frac{1}{2}} \varepsilon_n(\mathcal{A}) = \mathcal{O}(r_n^{-\frac{1}{2}-\frac{1}{d+1}})$$

We note that our *constant* in the equation above differs slightly from the one in Chen and White, 1999, as we do not restrict the domain of our activation function to $(0, 1)$ but to $[0, c_l]$. Nevertheless we obtain a constant, in this case assume that it is multiplied by c_l.

What is left is to verify condition 3. To obtain the metric entropy with bracketing we employ Shen and Wong, 1994 lemma 5 and the methodology of White, 1990 lemma 4.3. We summarise the results in the following claim:

Claim (Bound on metric entropy with bracketing)**.** An upper bound on the metric entropy with bracketing for sieve order r_n and $\omega > 0$ can be expressed as:

$$\mathcal{H}_B(\omega, \mathcal{F}_n, ||\cdot||_2) \leq \text{constant} * r_n \log(\frac{\text{constant} * r_n}{\omega})$$

To prove the claim, we first note that by Ossiander, 1987 and Chen, 2007 (p. 5595) an upper bound on the bracketing metric entropy is sufficient and together with condition 4, we can write this upper bound as:

$$\mathcal{H}_{[]}(\epsilon, \mathcal{F}_n, ||\cdot||) \leq \log N(\epsilon^{\frac{1}{s}}, \Theta_n, ||\cdot||) \leq \log N(\epsilon^{\frac{1}{2}}, \Theta_n, ||\cdot||) \ ,$$

where $\mathcal{F}_n$ and Θ_n are as defined in condition 3. Note that we adapt the notation slightly by changing ω to ϵ, to separate the claim from the rest of the proof. $N(\epsilon, \Theta_n, \rho)$ is the *covering number* which is the cardinality of the smallest set of open balls with radius ϵ that is needed to cover (the set) Θ_n. We denote the set of centers of these balls T_ϵ (which is also called an ϵ-net). ρ is the corresponding metric. This also illustrates the rationale behind condition 3. As the sieve space increases, the variance of the estimation increases. By controlling for δ_n in theorem 2.2.2, we take that into account.

For T_ϵ to be a valid ϵ-net of Θ_n, we must have that $\forall\ \theta \in \Theta_n\ \exists\ t_k \in T_\epsilon$ such that $\rho(\theta, t_k) < \epsilon$. We can then follow White, 1990 (proof of lemma 4.3), set $\eta > 0$ and let $B_\eta \equiv \{b_k \in B, k = 1, ..., l\}$ and $G_\eta \equiv \{g_k \in G, k = 1, ..., q\}$ be η-nets for $B = \{\beta : ||\beta|| \leq c_n\} \subset \mathbb{R}^{r_n}$ and $G = \{\gamma : ||\gamma|| \leq r_n c_n\} \subset \mathbb{R}^{r_n(1+d)}$ respectively (which we imposed with assumption 2.3.3). We will also follow White, 1990 and employ the L_1-norm. Because we are in a finite dimensional space, it will hence also be an upper bound for, for example, the L_2-norm.

Then, we denote $M_\eta = B_\eta \times G_\eta$ and let $\tilde{T}_\eta$ be the η-net for $\{\theta \text{ such that } \theta \in M_\eta\}$. We can then choose η such that $\tilde{T}_\eta$ will be an ϵ-net. By using our definition of θ from equation (2.3) ($\theta = (\beta', \gamma')'$) we have that for an arbitrary θ there will exist $\mu = (b', g')' \in M_\eta$ such that $||\beta - b|| < \eta$ and $||\gamma - g|| < \eta$. Denote $t(\theta)$ the element of $\tilde{T}_\eta$ corresponding to μ. We can then solve (again, analogous to White, 1990):

$$
\begin{aligned}
|f_n(\tilde{x}_t, \theta) - f_n(\tilde{x}_t, t(\theta))| &= |\sum_{i=1}^{r_n} \beta_i G(\tilde{x}_t \gamma_i) - \sum_{i=1}^{r_n} b_i G(\tilde{x}_t g_i) + \sum_{i=1}^{r_n} b_i G(\tilde{x}_t \gamma_i) - \sum_{i=1}^{r_n} b_i G(\tilde{x}_t \gamma_i)| \\
&\leq |\sum_{i=1}^{r_n} (\beta_i - b_i) G(\tilde{x}_t \gamma_i)| + |\sum_{i=1}^{r_n} b_i (G(\tilde{x}_t \gamma_i) - G(\tilde{x}_t g_i))| \\
&\leq ||\beta - b|| c_l + c_n (d+1) L ||\gamma - g|| \\
&\text{where we note that our ReLU function is lipschitz continuous} \\
&\text{with lipschitz constant L=1 and we use standardized inputs x.} \\
&\leq \eta (c_l + c_n (d+1)) = \epsilon
\end{aligned}
$$

Which means we have to set $\eta = \epsilon / (c_l + c_n(d+1))$. We then have that $T_\epsilon \equiv \tilde{T}_{\epsilon/(c_l + c_n(d+1))}$ is an ϵ-net for our ANN parameters. To calculate the covering number we can then use that the covering number of $\tilde{T}_\eta$ is bounded by the product of the covering number of its components B_η and G_η. We can then employ lemma 5 from Shen and Wong, 1994 for the L_2-norm, with our calculated η. Note that we adapt the δ to c_n for B and $c_n r_n$ for G respectively (and transform n term to r_n and $r_n(d+1)$). For the L_2 norm, N_1 from lemma 5 Shen and Wong, 1994 for G can then be expressed as:

$$
N_1 \leq \frac{(\pi^{\frac{1}{2}} c_n r_n)^{r_n(d+1)} / \Gamma(\frac{c_n r_n}{2} + 1)}{(\eta / \sqrt{r_n(d+1)})^{r_n(d+1)}}
$$

Which can then be solved by plugging in the calculated ϵ and approximating the solution with Stirling's formula. Finally, we note that only the sieve term r_n and the radius of the n-ball ϵ are not fixed a priori. We can therefore summarize the other terms in the *constant* and obtain:

$$
\mathcal{H}_{[]}(\epsilon, \mathcal{F}_n, ||\cdot||_2) \leq \text{const.} * r_n(d+1) \log(\frac{\text{const.} * r_n c_l (d+1) c_n}{\epsilon}) = \text{const.} * r_n \log(\frac{\text{const.} * r_n}{\epsilon})
$$

Which is the same as claimed. The constant is positive and contains the $\frac{1}{s}$ term from Chen, 2007 (p. 5595). Note that this also corresponds to the bound on the entropy in Chen and White, 1999, but the constant is multiplied additionally by c_l.

Then in order to verify condition 3, we simplify the equation (where we leave out the

constant terms for ease of notation because they are positive and can hence be taken to the RHS of the inequality in condition 3):

$$\begin{aligned}
\frac{1}{\sqrt{n}\delta_n^2}\int_{b\delta_n^2}^{\delta_n}\sqrt{r_n\log(\frac{r_n}{\omega})}d\omega &\leq \frac{1}{\sqrt{n}\delta_n^2}\sqrt{r_n}\Big[\big(\int_{b\delta_n^2}^{\delta_n}\log(\frac{r_n}{\omega})d\omega\big)^{\frac{1}{2}}(\delta_n-\delta_n^2 b)^{\frac{1}{2}}\Big] \\
&= \frac{1}{\sqrt{n}\delta_n^2}\sqrt{r_n}(\delta_n-\delta_n^2 b)^{\frac{1}{2}}\Big[(\delta_n-\delta_n^2 b)\log(r_n)+\log(\delta_n^2 b)\delta_n^2 b \\
&\quad -\log(\delta_n)\delta_n+(\delta_n-\delta_n^2 b)\Big]^{\frac{1}{2}} \\
&\leq \frac{1}{\sqrt{n}\delta_n^2}\sqrt{r_n}(\delta_n-\delta_n^2 b)^{\frac{1}{2}}\Big[(\delta_n-\delta_n^2 b)\log(r_n)-(\delta_n-\delta_n^2 b)\log(\delta_n) \\
&\quad +(\delta_n-\delta_n^2 b)\Big]^{\frac{1}{2}} \\
&\quad \left(\text{By setting eg: } b=\frac{\log(\delta_n)-1}{\delta_n(\log(\delta_n)-1-r_n)}\geq 0, \forall r_n\geq 1\right) \\
&\leq \frac{1}{\sqrt{n}\delta_n^2}\sqrt{r_n}\sqrt{\log(r_n)}\delta_n
\end{aligned}$$

Where we used the Cauchy-Schwarz theorem in the first inequality and the fact that $\delta_n \in (0,1)$. Note that we can choose the value for the constant c_2 to satisfy the condition 3 (ie. $\delta_n \in (0,1)$). Rewriting the condition:

$$\delta_n \geq c_2\sqrt{\frac{r_n\log(r_n)}{n}}$$

Finally, if we set $\delta_n = ||\theta_0 - \pi_n\theta_0||$ to balance bias and variance as suggested in Chen and Shen, 1998, we can establish a rate for the sieve order r_n:

$$c_2\sqrt{\frac{r_n\log(r_n)}{n}} \leq \delta_n = ||\theta_0-\pi_n\theta_0|| \leq \text{constant} * r_n^{-(\frac{1}{2}+\frac{1}{d+1})}$$

$$\implies r_n^{2(1+\frac{1}{d+1})}\log(r_n) \leq \left(\frac{\text{constant}}{c_2}\right)^2 n = \mathcal{O}(n)$$

The last part of the proof then consists of establishing the convergence of the sieve estimate based on n. It is sufficient to calculate the convergence of the dominant term in the relation above. Hence we can proceed in two steps:

Claim (Dominant term)**.** The dominant term of $r_n^{2(1+\frac{1}{d+1})}\log(r_n)$ is $r_n^{2(1+\frac{1}{d+1})}$

To prove the claim we note the following, our dataset of size n and the dimension of the input $(d+1)$ are both assumed to be ≥ 1 (which otherwise would render the proof useless). Note then that both terms in the equation are continuous and strictly increasing in r_n. Further, $2(1+\frac{1}{d+1}) = \text{constant} > 2 \;\forall\; d < \infty$, define that constant as c for ease of

notation. To show that the dominant term is indeed r_n^c we use the following:

$$\lim_{r_n \to \infty} \frac{\log(r_n)}{r_n^c} = \lim_{r_n \to \infty} \frac{\frac{1}{r_n}}{c r_n^{c-1}} = \frac{0}{\infty} = 0$$

Where we used l'Hôpitals Rule in the first step. We then claim the following:

Claim (Behaviour of r_n)**.**

$$r_n = \mathcal{O}(\Big(\frac{n}{\log(n)}\Big)^{\frac{1}{c}})$$

To prove the claim we proceed in two steps, first consider the case where:

$$r_n^c \leq \sqrt{n} \text{ then : } \frac{r_n^c}{\frac{n}{\log(n)}} \leq \frac{\sqrt{n}}{\frac{n}{\log(n)}} = \frac{\log(n)}{\sqrt{n}}$$

By using the the same procedure as in the previous claim twice and the fact that $n \geq 1$, we have $\frac{\log(n)}{\sqrt{n}}$ is continuous and $\to 0$, as $n \to \infty$ and hence it is bounded by some constant $const._1$ which results in:

$$r_n \leq \Big(const._1 \frac{n}{\log(n)}\Big)^{\frac{1}{c}}$$

In the second case consider $r_n^c > \sqrt{n}$ then we have:

$$\frac{r_n^c \log(n)}{2} = r_n^c \log(\sqrt{n}) < \text{constant} * r_n^c \log(r_n) \text{ which we saw above is set as } = \mathcal{O}(n)$$

Noting that by the definition of $\mathcal{O}(n)$ this implies: $\frac{r_n^c \log(n)}{2} \leq \text{const.}_2 n$ and hence:

$$r_n \leq \Big(2\text{const.}_2 \frac{n}{\log(n)}\Big)^{\frac{1}{c}}$$

Taking these conditions together yields:

$$r_n \leq \max\{const._1^{\frac{1}{c}}, (2const._2)^{\frac{1}{c}}\}\Big(\frac{n}{\log(n)}\Big)^{\frac{1}{c}} = \mathcal{O}\bigg(\Big(\frac{n}{\log(n)}\Big)^{\frac{1}{c}}\bigg)$$

Then, because we set $\delta_n = ||\theta_0 - \pi_n \theta_0||$, we can combine the approximation error rate with the above and get the final convergence rate of:

$$||\hat{\theta}_n - \theta_0|| = \mathcal{O}\bigg([n/\log(n)]^{\frac{-(1+\frac{2}{d+1})}{4(1+\frac{1}{1+d})}}\bigg)$$

Which completes the proof.

□

Non-Technical Appendix

B.1 Metrics

Where not else defined, the following metrics apply (B denotes the total number of MC iterations):

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y_t})^2$$

Note that when we run MC simulations, we observe the true function and the MSE is calculated on the true y_t. For applications where the MSE is difficult to calculate, we report the RMSPE as described below. In cases where we use the RMSPE, where we only train the model with data up to T (ie. observations $T+1, ..., n$ are not used in the modelling phase)

Root Mean Squared Prediction Error (RMSPE)

$$\text{RMSPE} = \sqrt{\frac{1}{(n-T)}\sum_{t=(T+1)}^{n}(y_t - \hat{y}_t)^2}$$

Squared Bias of Estimator (Bias^2)

$$\text{Bias}^2 = \big(\frac{1}{B}\sum_{i=1}^{B}(\hat{y}_{t,i}) - y_t\big)^2, \quad t = 1, ..., n$$

Where B is the number of simulations and n the total number of observations. To get a single value, we then take the mean of the MSE, the squared bias and the variance (blelow) across all observation points.

Variance of the estimator (Var_e)

$$\text{Var}_e = \frac{1}{B}\sum_{j=1}^{B}\big(\frac{1}{B}\sum_{i=1}^{B}(\hat{y}_{t,i}) - \hat{y}_{t,j}\big)^2, \quad t = 1, ..., n$$

B.2 ARCH∞ to GARCH(1,1) transformation

To transform the ARCH_∞ model to a GARCH(1,1) model we use the lag operator L such that $Ly_t = y_{t-1},\ L^2y_t = y_{t-2}$ and so on. Then, writing a GARCH(1,1) process as:

$$\sigma_t^2 = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \beta_1 L\sigma_t^2$$

$$\implies (1 - \beta_1 L)\sigma_t^2 = \alpha_0 + \alpha_1\varepsilon_{t-1}^2$$

and assuming that β_1 is smaller than 1 in modulus, we can rewrite the GARCH(1,1) equation as:

$$\begin{aligned}\sigma_t^2 &= (1 - L\beta_1)^{-1}\alpha_0 + (1 - L\beta_1)\alpha_1\varepsilon_{t-1}^2 \\ &= \frac{\alpha_0}{(1-\beta_1)} + \alpha_1\sum_{i=1}^{\infty}\beta_1^i\varepsilon_{t-1-i}^2 = \text{ARCH}\infty\ ,\end{aligned}$$

where we note that the lag operator on a constant is the constant, as described in, for example, Martin, Hurn, and Harris, 2012.

B.3 Code

All code used in this thesis can be found on the GitHub Repository[1]

We also want to note the following: all performed simulations can be rerun in your local environment (provided you are using the same platform). Wherever possible, the simulations were made reproducible. Yet, as parallelization makes an exact reproduction impossible, you might find some numerical differences in the monte carlo (MC) simulations involving artificial neural networks. As a rule, all artificial data and non-ANN models are reproducible. `keras` models are only reproducible when the command `use_session_with_seed()` command was used, which suppresses multi-core calculations. This was *not* done in the MC simulations to save compute time. However, because all MC simulations are averaged over multiple estimations, the differences should be rather small. Further, for figure 2.8, we used the `plotly` package. A bug prevents that the

[1] https://github.com/chtonios/sieve_forecasting

visualisation can be stored directly from the R-Environment. The figure is reproducible but needs to be manually stored (unlike the other visualisations).

B.4 Models

This section contains the description of the models that were used throughout the thesis. Note that we index time-series models with t and iid models with i.

B.4.1 Chaos Model

The autoregressive chaos model is adapted from Kuan and White, 1994, where we changed the linear autoregressive term slightly to avoid a unit root. It is generated according to:

$$y_t = 0.3y_{t-1} + \frac{22}{\pi}\sin(2\pi y_{t-1} + 0.\overline{33}), \quad t = 1, ..., 600 \tag{B.1}$$

B.4.2 Irregular IID Model

This model contains a highly nonlinear relationship with between the x and y variables. Additionally, white noise has been added to the process to investigate the bias-variance tradeoff. The model is generated according to:

$$y_i = 0.4(x_i - 10)^3 + 0.1(\frac{x_i}{7})^7 + 600\sin(2x_i) + \mathbb{1}_{x_i>1}(-800\sin(2x_i) - 200) + \varepsilon_i \tag{B.2}$$

$$i = 1, ..., 400 \quad x_i \sim \text{Unif}([-10, 10]), \quad \varepsilon_i \sim \mathcal{N}(0, 441^2)$$

B.4.3 High Dimensional IID Model

The high dimensional model 1 is generated according to the process:

$$y_i = \sum_{i=1}^{k} f_i(X_i) + \varepsilon_i \tag{B.3}$$

$$k = 2, 5, 15, \quad X_i = \begin{cases} \sim \text{Unif}([0, 3]), & \text{relevant} \\ \sim \text{Unif}([-3, 3]), & \text{otherwise} \end{cases}, \quad \varepsilon_i \sim \mathcal{N}(0, 7^2)$$

Estimation is then based on all relevant variables plus the specified number of irrelevant components. For an overview of the used functions in the high dimensional model, refer to table B.1.

	Function
f_1	$\sin(7x)^4$
f_2	$\lvert 2x\rvert^3$
f_3	$2x^4$
f_4	$-0.4(x^2 + x^3 + 0.1\log(\max(\lvert x\rvert, 0.5)))$
f_5	$-4x^3$
f_6	$7x^2$
f_7	$2\log(\lvert x\rvert + 0.9)^3$
f_8	$-\lvert 2x\rvert^3$
f_9	$-(0.9x^2 + x^3)/(1 + \sin(x) + 2x^5))$
f_{10}	$-4\cos(\pi 22x)$
$f_{n>10}$	$-2\sin(nx)$

TABLE B.1: The functions generating the high dimensional model. If the number of relevant predictors exceeds 10, the function becomes a shifted sinus curve. Note that *all* generated models contain the number of nuisance parameters indicated in addition to the gaussian error term.

B.4.4 Multivariate ARCH Model

We simulate the error distribution from a skewed generalized t-distribution. The distribution depicts the fat tails and skewed distribution frequently arising in financial data. To draw the errors we set the skew parameters to -1.5 for all series that we draw. The covariance matrix is then specified by drawing randomly from a uniform distribution (Unif$[-1, 1]$) and creating a positive definite matrix from there. Figure B.1 visualizes the results from such a draw.

B.5 Model Specifications

Table	Layers	Units	Activation	L_1-Regularization	Total Steps
3.1	1;2	250;250-5	S; S-L	-	4e4
3.2	1	200	R	1e-5;1e-6;1e-6	3e3/4e3/5e3
3.3	2	150-5	R-R	1e-6	2e4
3.3	2	150-5	R-R	5e-6	1e4
3.4	2	150-20	R-R	2e-8;2e-3	1e4

TABLE B.2: Model specification for the simulations. A note on terminology: S denotes a sigmoid activation function, R a ReLU and L denotes the linear (identity) activation. The total steps are calculated by multiplying the number of batches by the number of epochs. The regularization was calculated on the specified amount of steps.

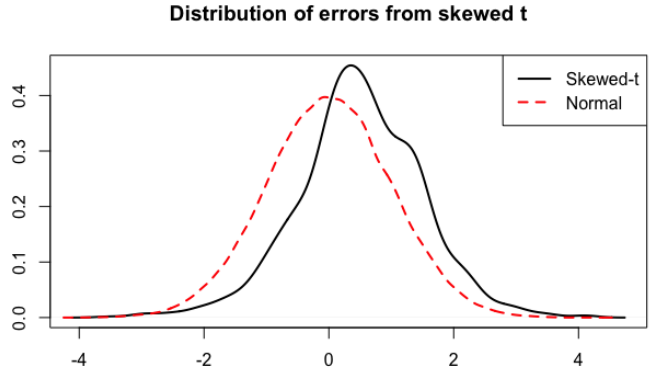

FIGURE B.1: Density of errors drawn from the skewed generalized t-distribution. A standard normal distribution is also depicted as a reference. The skew and fat tails can be seen.

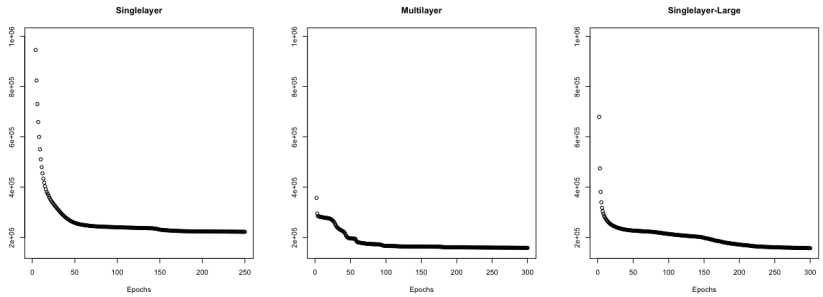

FIGURE B.2: Convergence of the loss (MSE) for the three discussed models. Note that the multilayer network with 408 hidden nodes does not seem to have converged. As the purpose of model was to show that it starts to overfit the data more than the multilayer network, this is sufficient, because we would expect the model to overfit the data even more with increasing training epochs.

B.5.1 Training loss for single and multilayer ANNs

Figure B.2 depicts the loss functions for the simulations in 2.2.4. Figure B.3 depicts the same for the results in table 3.1.

B.5.2 Training loss for ANNs in 3.3

Figure B.4 shows the loss with increasing number of epochs for the high dimensional model estimation.

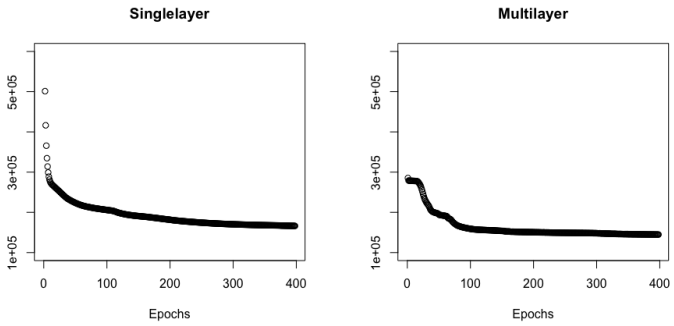

FIGURE B.3: Single- and multilayer loss functions for the simulation. The loss function of the multilayer network levels off faster and could hence be stopped earlier than the single layer network, further decreasing the variance.

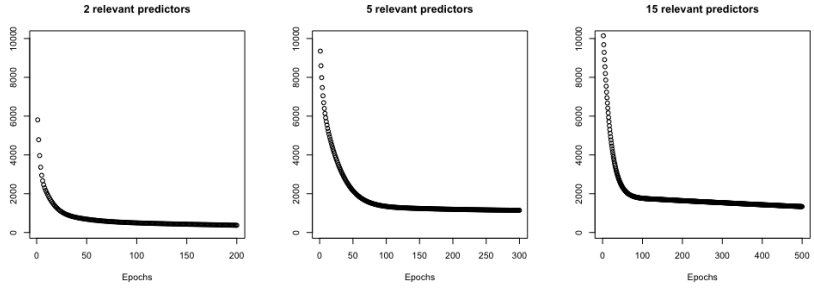

FIGURE B.4: Loss functions for ANNs trained in the high dimensional case (with 2,5 and 15 predictors and 1 noise variables). Note that the differences in the x-scale that makes the decrease for the 15-predictor network seem steeper towards the end.

B.6 Additional Results from Application

B.6.1 Portfolio Construction and Testing

The portfolio consists of three stocks and three other assets which are summarised in table B.3 (for their log-returns). Each asset is then weighted by a random weight under full investment constraint and no short selling (ie. for for every weight w_i we have $w_i \geq 0$ and $\sum_{i=1}^{6} w_i = 1$) to construct a portfolio. The weights are drawn from an exponential distribution with rate $\lambda = 1$ and adjusted to satisfy the above constraints. A total of 50 sets of weights are drawn and the returns calculated.

Figure B.5, depicts the train test split and the log-returns of a portfolio with equal weighting ($w_i = w_1 = 1/6$) of the assets. The beginning of the test data still has a higher volatility from the great recession and the European debt crisis in late 2011 (with the explosion of the long-term interest rate for Greece) is also visible.

The SANN estimates the quantile directly and assumes symmetric returns for simplicity. The portfolio VaR can be retrieved from the MGARCH model by using the forecasts of the covariance and correlation matrices and calculating the variance. From there we can

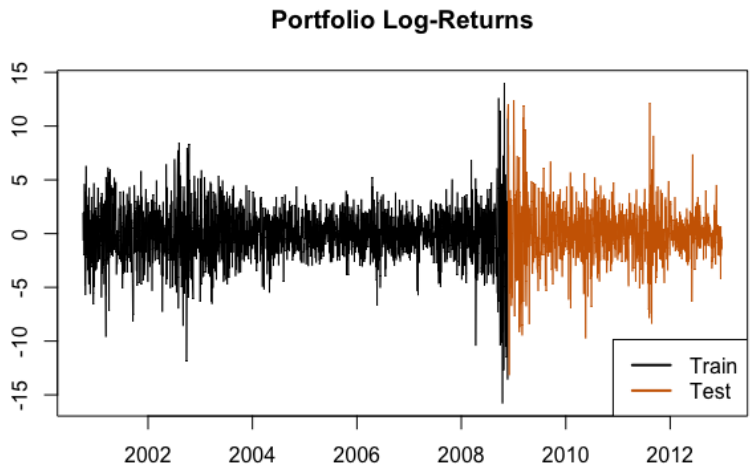

FIGURE B.5: Train-Test split for portfolio VaR estimation. The portfolio consists of all equally weighted assets.

estimate the VaR by using the given quantile. To get the portfolio variance we calculate:

$$\begin{bmatrix} w_1 & \dots & w_6 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \rho_{1,2}\sigma_{1,2} & \dots & \rho_{1,6}\sigma_{1,6} \\ \vdots & \ddots & & \vdots \\ \rho_{1,6}\sigma_{1,6} & & & \sigma_6^2 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_6 \end{bmatrix}$$

The stock market data (GE, Walmart and Altria) were retrieved from Kaggle (`www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs`) and the rest from the federal reserve bank of St. Louis.
(Gold: `fred.stlouisfed.org/series/GOLDAMGBD228NLBM`,
Brent `fred.stlouisfed.org/series/DCOILBRENTEU`
USD/CHF exchange rate `fred.stlouisfed.org/series/DEXSZUS`).

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
GE	2,992	−0.032	1.312	−8.434	−0.566	0.548	11.052
Walmart	2,992	−0.017	0.435	−2.483	−0.242	0.195	3.056
Altria	2,992	−0.010	0.659	−6.315	−0.298	0.306	4.126
Brent	2,992	−0.002	0.993	−8.485	−0.543	0.562	7.696
Gold	2,992	−0.0001	1.006	−7.177	−0.492	0.538	7.906
USD/CHF	2,992	0.004	1.003	−6.928	−0.565	0.564	12.452

TABLE B.3: Summary statistics for the used assets over the time period 01.01.2000-31.12.2012.

	GE	Walmart	Altria	Brent	Gold	USD_CHF
GE	1.722	0.242	0.238	0.101	-0.021	0.033
Walmart	0.242	0.190	0.065	-0.020	-0.021	0.034
Altria	0.238	0.065	0.435	0.022	-0.015	0.027
Brent	0.101	-0.020	0.022	0.987	0.145	-0.177
Gold	-0.021	-0.021	-0.015	0.145	1.012	-0.215
USD/CHF	0.033	0.034	0.027	-0.177	-0.215	1.006

TABLE B.4: Unconditional variance-covariance matrix for the log-retruns of the assets from the portfolio.

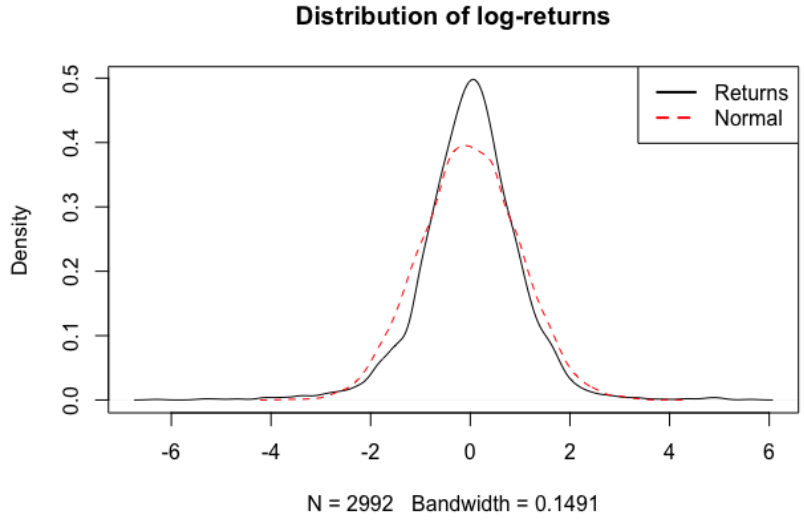

FIGURE B.6: Return distribution of (scaled) log returns with a reference $\mathcal{N}(0,1)$ distribution. Clearly visible is the excess kurtosis of the returns.

Declaration of Authorship

I hereby confirm that the work presented has been performed and interpreted solely by myself except for where I explicitly identified the contrary. I assure that this work has not been presented in any other form for the fulfillment of any other degree or qualification. Ideas taken from other works in letter and in spirit are identified in every single case.

Signed:

Date:
